International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2015): 78.96 | Impact Factor (2015): 6.391

Multi Rates and Resolutions SPIHT with Low Memory

Yahya Ali Lafta AL Husseini

University of Kufa, Faculty of Engineering, Department Electronics & Communication, Iraq

Yahyaa.alhusseini[at]uokufa.edu.iq

Abstract: ***For improvement a highly scalable image compression system in order at sizes (resolutions) and qualities (bit rate) are obtained to obtain simply decoding that is achieved by a compressed image. Since, there are many users when the Internet is used and into required that is various user's request. Thus, need how can able to induce that without complexity and much data can be has quality and preserve their size. Therefore, a novel algorithm dominated Highly Scalable-Single List-SPIHT (HS-SLS) is proposed to introduce those purpose. The adaptable bit-stream which is created from HS-SLS algorithm when that is both rate scalable and resolution are gotten as a robustly scalable bit-stream. This can readily its encoder, at any bit-rate involving, can be modified to different resolution requirements, without the required to decode the bit-stream, a very straightforward scaling process that is implemented on-the-fly. An unique feature of the novel algorithm is that due to its easy memory management it has low requirement and low memory complexity. As well as, the size can be established which avoid the dynamic memory allocation problem due to the size of the applicable memory is stationary. For hardware enforcement, the HS-SLS algorithm very convenient due to its merits. As compared to the original SPIHT algorithm that, for these worthy features is very petty shorthand in the algorithm's performance, the price is paid.***

Keywords: Highly Scalable Image Compression, Low Memory Image Compression, Set Partitioning image Compression Rate scalability, Resolution scalability, Wavelet image compression SPIHT, Zero-tree coding

1. Introduction

Nowadays have a witnessed a wide utilize of video topics and image on the Internet and also for many applications that Image security becomes increasingly important, video surveillance, confidential transmission, medical and military applications [1, 2].Users have been provided the option to reduce significant amount and the size of image through image compression[3]. However, for a particular scenario, there are many attributes may be even more important. For example, the primary concern may be observed for decoding time in medical diagnosis. Besides that, low power consumption and small memory are essential for mobile devices. Over packet networks for broadcasting, scalability in resolution or/and bit rate may take seniority [1]. Moreover, Scalable image compression may attain packages with more number of notions related to quality preservation with image compression. At the meantime, the web server can rapidly and easily regulate the bitrate to be transmitted in order to occupy the wide range of network bandwidth alteration when a good scalability is used as a key feature. Moreover, a good codec, hand-held and portable devices request capability suitable for those applications should also be need minimal power consumption and easy to implement [4].

Researchers have developed many approaches to achieve scalable image compression [2]. Wavelet Transform has received a considerable interest to develop new image compression algorithms. The Wavelet means when a signal is represented that the high pass coefficients represent localized change in short data segment, and the low pass coefficients represent slow changing long data segment. It approaches a wonderful framework in which both long term trend and short-term anomaly can be analyzed on equal footing. So that, the eventual wavelet coefficient can be easily scaled in resolution whereby the original one is larger than an image should be reconstructed with 2^M times, where M is the level of wavelet coefficients [6]. The whole original image is induced due to the DWT apply Image compression methodologies utilizing. In the different regions of interest should be the wavelet coefficient relationship when the different compression ratios are applied, in which either each wavelet domain band of the image itself is clustered or the transformed image, respectively [5]. When the coefficients into subbands are arranged belonging to different resolutions or scales that makes possible decoding or coding different scales. Due to the local nature of the transform that makes possibility selected regions of interest in the image for random access, so at different scales to decode a particular region of an image when one can select regions of subbands [2].

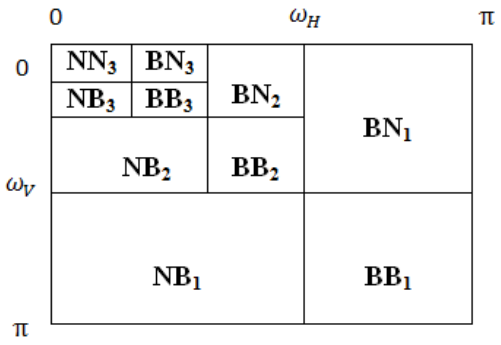

Figure 1: dyadic wavelet transform for Subbands of a 3-level

Since the invers discrete wavelet transform (IDWT) has in its construction bank filter that make us to do subband arrangement so by three stages of a swap high and low pass vertical and horizontal filtering of outcome low vertical and low horizontal subbands followed by 2:1 downsampling. Figure 1 illustrated the analysis and

synthesis for two stages. The source image itself just can be considered by the first stage and the first stage input low horizontal, low vertical, 2nd stage is called the LL2 subband that is merely a coarse, where it reduced by factor 22 of the original image in both dimensions. The NN1 subband yields from filtering and upsampling of, BN2, BB2, NN2, and LH2 subbands which it is two times the scale of NN2 so it is, by a factor of 21, still scaled down from the original. For getting the full gauge reconstruction of the original input that for one more stage repeats on the LL1 subband. Thus, we can get a reduced gauge version of the original image at each synthesis stage [2]. However, converting the image samples into a more compressible form when the two-dimensional Discrete Wavelet Transform (2D-DWT) is used with JPEG2000 standard where providing it the resolution scalability which it considers one of the outstanding features for it [5]. Whatever a base stream providing scalable (refinement) streams as well as a low (base) image quality is generated by Scalable image coding that successively improves the image quality [7].

2. Related Work

The core of the JBEG2000 standard is called the embedded block coding with optimized truncation (EBCOT) algorithm, among all wavelet-based scalable image compression it is the most efficient. Due to using arithmetic coding the algorithm has high computational complexity unfortunately. Each coefficient in each bit-plane coding pass when three times is scanned Therefore to get the intended highly scalable bit-stream a time consuming rate distortion optimization process is required to the compressed bit stream. Attaining in general Digital Signal Processor (DSP) or Purpose Processor (GPP) chips are made EBCOT very difficult and slow with those factors [12, 17].

Besides this such as Partitioning Embedded block (SPECK)[9] and Set Partitioning in Hierarchical Trees (SPIHT)[8] for setting portioning image compression algorithms when at all in performance compared to JPEG2000 [10] the coders run by about (5-7) times faster and possess all the desirable features and sacrifice very little. The maximum magnitude in it is above is determined a certain threshold when some kind of significance testing for sets of pixels employs by these schemes. The set is significant (SIG) when it is partitioned and it is represented by one symbol when it is insignificant (ISIG). In any case the SIG sets until all pixels are encoded, the same process should be recursively repeated. So these schemes achieve advantages such as they generate a rate scalable compressed bit-stream, good performance and they have relatively low computational complexity. Keeping track of which pixels/sets need to be tested these algorithms is used linked lists for the high memory requirements that is the price to be paid for these advantages. The image size may be less than the lists sizes at high rates. In addition, due to the continual process of removing and adding nodes to and from these lists a variable data dependent the need for complex memory management and memory requirement is caused with the use of lists [11]. Lately, the generated bit-stream do not provide a bit-stream and does not explicitly support resolution scalability that can be simply rearranged according to the rate and resolution desired by a decoder [12, 13].

Obtaining highly scalable bit stream such as the set partitioning schemes added to the resolution scalability has been presented, in the literature, via several works. The Highly Scalable-SPIHT (HS-SPIHT) algorithm is offered by A. Mertins and H. Danyali [13]. Here, producing highly scalable bit-stream due to the original rate scalable SPHIT is upgraded. The resolution scalability is appended via The HS-SPIHT through the resolution-dependent coding passes and introduction of multiple resolution-dependent lists. No explicit rate allocation mechanism considers the simplicity of bit-stream scaling process that is the main advantage of the HS-SPIHT coder. Yet, the terms of high memory requirements are still the drawbacks for both the HS-SPIHT and original SPIHT. Furthermore, in order to store the resolution level of each set in the corresponding resolution dependent list so the HS-SPIHT needs a mechanism to distinguish it. Obviously, the algorithm complexity compared to the original SPIHT is increased for these requirements [13].

K. KhanandA. monauwer offered the Listless HS-SPIHT (LHS-SPIHT) [14]. Here, the resolution attached list is replaced when by state markers with an average of 4 bits/coefficient should be using with HS-SPIHT to hold trajectory of set significance information and pixels. There is similarity when the set partitioning rules is used LHS-SPIHT and HS-SPIHT too so that identical bit-stream should be produced. Since the coder considers having high computational unfortunately due to, in each coding pass, the entire image pixels twice time must be processed via the coder, though in memory usage the coder is very efficient. the newly significant pixel is coded when Insignificant Pixel Pass at the first time and finding significant in the previous coding passes when the pixels is refined in the refinement pass at second time. Two main aims are worthily noted when the SPIHT uses the list: the first aim as the lists are processed in a specific order whereby preserving information ordering should obtain performance. The second goal, in each coding pass, in the lists as only the elements stored are processed in order complexity reduction[15, 16]. Thus eliminating the lists will obviously may decrease its performance and increase the complexity of the algorithm. Here the LHS-SPIHT reduces memory usage at the cost of decreasing performance and increasing complexity so that it swaps memory for efficiency and complexity i.e.

In stance of employ the coder to satisfy its requirement such as low memory that has been proposed in [17] where is depicted Single List SPIHT (SLS) as a novel coder. This coder has memory about six times less than the original SPIHT. Actually a rate scalable bit-stream is produced via the SLS coder as SPIHT. The experimental results and theoretical analysis indicated the proposed SLS algorithm has nearly the same complexity as the original SPIHT and has better performance. Indeed thus give enhance the performance of SPIHT without paying any additional overhead cost and without affecting its simplicity and also

by about six times the memory requirements is reducing. In order in this paper the SLS algorithm to achieve a highly scalable bit-stream that should be upgraded. Highly Scalable-Single List SPIHT (HL-SLS) is depicted with the proposed algorithm. Solving the scalability problem with the new HL-SLS coder is induced through the introduction of the resolution dependent bit-plane coding passes and the resolution dependent parts list. Moreover, as the original SLS encoder, the proposed HL-SLS has nearly the same memory requirements and the same complexity and management. The remainder of the paper is organized as follows: section III gives brief review for the SPIHT and the SLS algorithms. Section IV introduces the new HS-SLS algorithm. Section V covers the simulation results of the algorithm. Finally, Section VI gives the concluding remarks of the paper and presents new directions for future work.

3. The SPIHT and the Single List SPIHT (SLS) algorithms

3.1 SPIHT

A procedure that uses recursions splitting of sets of samples **or** of partitioning guided by a threshold test is the Set partition coding, so it is the principles of a data compression method [8]. This technique is based on some very simple principles; it considers one of the most interesting aspects, however is also very surpassing, effective other coding methods that may seem much more advanced theoretically. It is also the primary entropy coding method in the well-known Set Partitioning in Hierarchical Trees (SPIHT) [9]. For direct coding of natural images should not be chosen like the set partition coding due to it not be efficient. Therefore**,** with small maximum values the source must contain primarily sets of samples to be efficient. Thus, here a source transformation is requested such as discrete wavelet (DWT) [8]**.** The DWT as a number of nonoverlapping spatial orientation trees (SOT) can be viewed. Due to at the same spatial direction these trees branch successively to higher frequency subbands, they so called. Figure 2 shows SOT in the wavelet transform domain [2].

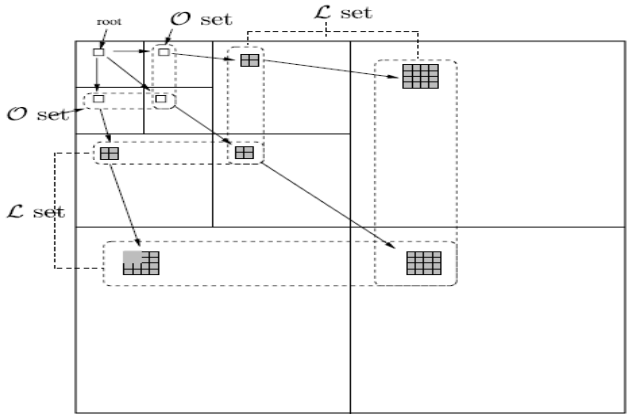

Figure 2: A SOT of DWT L denotes grand-descendant set and O denotes offspring the full descendant set D =O ∪ L.

As well a way that each node has either four offsprings or no offspring is the tree defined where always form a group of 2X2 neighboring pixels. From Figure 2 also can be shown the tree divide it's the set of all coefficients descending (grand) that means the set of from nods the all descendants and immediate offspring (child) coefficients from the child coefficients that means the set of from a single node the three or four nodes branching [2].

.

A comparison of two values is involved of the SPIHT algorithm a time which results in no/yes condition. From Figure 3, Namely for stored insignificant information require three ordered lists: the List of Insignificant Sets (LIS)**,** which the set of coefficients are having their magnitude smaller than the threshold considered, is one of the three lists categorizes sorting pass coefficients, the second list is List of Insignificant Pixels (LIP) that the threshold considered is larger than their magnitude pixel, last one is List of Significant Pixels (LSP) whose magnitude is greater than that of considered threshold. The ability of this coding method can be to generate codes with fractional bits due to obtaining by optimal performance. Therefore, widely it is used in different image compression technique [10].

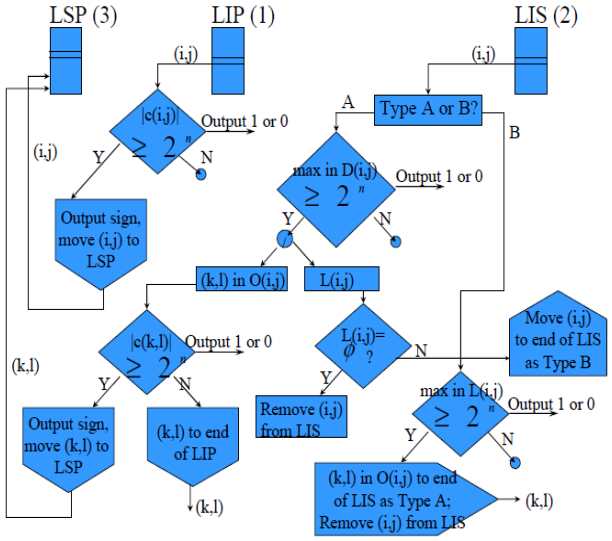

Figure 3: Flow chart for Initialization of lists can be used to code the significance of trees rooted in groups of pixels of size 2X2

For one each bit-plan level, the SPIHT algorithm consists of several bit-plan and initialization coding passes. Based on the maximum value of the DWT firstly it computes m_{max} which presents the maximum value of the DWT image that it given by:

$$m_{max} = \lfloor \log_2 \left(\max_{\forall (c,r) \in I} |F_{c,r}| \right) \rfloor \quad (1)$$

where at location (c, r) $F_{c,r}$ is a wavelet coefficient, I is the DWT image and in order to start decoding at m_{max}, $\lfloor z \rfloor$ is the nearest integer $\leq z$. m_{max} as a side information is sent to the decoder within the bit-stream. Afterwards with the coordinates of all the pixels in the LL_K subband, the algorithm is initialized the LIP, the LSP as an empty list, and the LIS have offspring as type A sets with all the coordinate of pixels in LL_K subband. The refinement and sorting are two passes that is encoded for each bit-plane. Here, for the first bit-plan is made for the sorting pass

only. The refinement pass codes the pixels and the new SIG pixels are identified the sorting pass that at previous passes are found SIG. when

$$2^m = |F_{c,r}| < 2^{m+1} \quad (2)$$

A pixels $F_{c,r}$ is considered SIG for given bit-plane with respect to m. similarly, the set contains one or more SIG pixels when with respect to m a set of pixels is considered SIG.

For significance with respect to m each entry in LIP is tested during the sorting pass. a (0) is transmitted to the compressed bit-stream when the pixel is ISIG. Moreover, a sign bit and a(1) acting the sign of that pixel (e.g., 1 for negative and 0for positive pixel) are transmitted to the bit-stream when the pixel is SIG. Later, the testing for each set in LIS that symbolize the root of an SOT as follows: the SOT of type A is tested when this type is the set for significance. a(0) is transmitted when the SOT is ISG(i.e., by a single bit the whole SOT is coded). in other hand, a(1) is transmitted when the SOT is SIG, so that each one of the set's four children is tested. If a(1), its sign bit are transmitted to the bit-stream, and the child is SIG, so to the end of LSP its coordinates are added. Furthermore, the coordinates of a(0)are added to LIP when the child is SIG and a(0) is transmitted, in the next bit-plan passes, to be tested. Lastly, the set is moved as a type B set to end of LIS when the set has grandchildren. Otherwise, the set of four children are added to the end of LIS as type A sets to be tested in the current pass, the set is removed from LIS, and a (1) is transmitted when the set is of type **B** and has at least one SIG child. Moreover, the set remains in LIS to be tested in next bit-plane passes, and a (0) is transmitted (all the set's children are ISIG).

All pixels in the LSP except those which have been added to the refinement pass during the current pass that by outputting its nth MSB are refined to the bit-stream. For bit-plane m after finishing refinement passes and sorting, m is decremented by 1 to begin coding the next bit-plane.

3.1.1 Drawbacks of the SPIHT algorithm [11, 13]

1. It has complex memory management due to adding nodes and the removing to the LIS and LIP lists.
2. Due to using the lists it needs a huge amount of memory. More precisely the LSP and LIS lists which store the (c, r) coordinates of individual pixels dominate the total storage. The number of image pixels is equal to the maximum number of entries in each list. Otherwise, due to its stores the (c, r) coordinates of the roots of the SOTs, the LIS has less memory requirement. As well, the roots in the highest level (BB1, BN1, and NB1subbands) are not stored due to they don't have descendants. The maximum number of entries of LIS is one-fourth of image pixels due to the size of these subbands is 3/4 the image size.. Thus, the working memory of SPIHT is at least 2.25 times the image size.
3. It does not explicitly support resolution scalability due to it produces a rate scalable bit-stream and thus cannot be easily reordered according to resolution and the rate desired by the decoder.
4. Due to the number of list entries can't be determined in advance as the lists sizes cannot be pre-allocated it depends on the amount of image details and on the compression bit-rate. This problem can solve by using pre-allocating the lists to the maximum size or either dynamic memory allocation. Evidently, and the latter one increases its memory requirements due to the former solution slows the algorithm.

3.2 SLS

The SLS algorithm [17] overcomes the huge memory requirements and the complex memory management drawbacks of the original SPIHT algorithm (the first three drawbacks). In addition, the SLS coder has slightly better performance than SPIHT. The basic idea of the SLS is based on the fact that the LIP and LSP are constructed from the offspring (the four children) of a root that belongs to a SIG SOT. So, instead of storing the offspring in these lists, they can be recomputed in each pass. Evidently, this will increase the complexity of the algorithm. However, this complexity increment is compensated by the reduced memory management overhead (as shown shortly). The SLS algorithm replaces the LIP and LSP lists (which dominate the total memory usage) by two bits state marker. In other words, each pixel is provided by two status bits referred to as σ to determine the type of the pixel as follows:

- An ISIG pixel that is not yet tested is New ISIG Pixel (NIP) when $\sigma = 0$.
- A pixel that is tested ISIG in the previous passes is Visited ISIG Pixel (VIP) when $\sigma = 1$.
- A pixel that just becomes SIG in the current bit-plane pass is: New SIG Pixel (NSP) when $\sigma = 2$.
- A pixel that is tested SIG in the previous passes is Visited SIG Pixel (VIP) when $\sigma = 3$.

In addition, using of a single list is made the SLS encoder described the List of Root Sets (LRS). Like the LIS, the LRS also saves the (c, r) coordinates of the roots of the SOTs. So one-fourth the image size is for the LRS exactly that is the same memory size as the LIS. Yet, implementing the LRS as a simple 1-D array is made possible with the adopted coding method that is accessed sequentially in First In First Out (FIFO) manner, it will be never removed due to once a set is added to the LRS. In contrast, the SPIHT coder used with the LIS must be implemented as a linked list that is accessed randomly due to the continual process of removing nodes and adding to from it. Evidently, implementing removing the LSP and LIP with the LRS as a 1-D array together will greatly simplify the memory management problem. Further, the type field used with, differently, the LRS is used. In the LRS a set that has no SIG SOT or A set of type A is untested set and a set of type B has one or whereas in the LIS set of type B represents the root of all descendants and a set of type A represents the root of all descendants and excluding the four direct offspring.

The SLS algorithm consists of several bit-planes coding passes and an initialization stage. At initialization, it first outputs the maximum bit-plane mmax and computes to the

bit-stream, and set m to mmax. Then, in LLK subband, it saves the (c, r) coordinates of the pixels that have offspring in the LRS as type A sets. By coding NNK subband, the first bit-plane pass starts which consists only of New ISIG pixels (NIPs). Each NIP, with respect to m, is tested for significance. If it is still ISIG, a(0) is updated to VIP and it is outputted to the bit-stream. If it becomes SIG, a(1) is updated to NSP, it and its sign bit are outputted to the bit-stream. Flowing, the sets of the LRS (which are sets of type A) are sequentially processed. Its SOT is computed for each set and then, with respect to m, tested for significance. a (0) is outputted (i.e., the whole SOT is represented by a single bit) when the SOT is still ISIG. Otherwise, a (1) is outputted when the SOT is SIG, each one of the set's offspring (which is an NIP) is coded the set is updated to type B set, and updated as given above. After, the offspring are added to the end of LRS as type A sets when the set has grandchildren (the set's offspring aren't in the highest resolution level) to be coded in the current bit-plane coding pass. Finally m is decremented by 1 to start a new bit-plane coding pass.

In NNK also, each one of the next bit-plane coding passes starts by coding the pixels. At these passes NNK may contain NSPs or/and VIPs only. In the same way of coding the NIP, a VIP is coded exactly, except it is still ISIG when that there is no need to update it due to it is already a VIP. Otherwise, a NSP is refined by outputting its mth MSB to the bit-stream when it after is updated to VSP. Flowing, the LRS list is scanned two times. Only the sets of type B are inspected in the first scan. Its four offspring is computed for each set, which may be of type NSPs or/and VIPs. Whereas a NSP is updated to VSP when a VIP is coded as given above to be refined later in the second scan pass. All the sets in LRS are inspected in the second scan. It is processed exactly when the set is of type A in the same way as was done in the first coding pass. Otherwise, only the VSPs of the set's four offspring are refined, when the set is of type B, by outputting its mth MSB bit to the bit-stream. Lastly m is decremented to start a new bit-plane coding pass.

3.3 The Proposed Algorithm for Highly Scalable-SLS (HS-SLS)

3.3.1 Description of HS-SLS Description

Generally speaking, a rate scalable bit-stream can be upgraded to a highly scalable one, i.e., a bit-stream that is resolution scalable and both rate if the coding information in each bit-plane coding pass are arranged in increasing order of resolution levels. As well, each level, in the output bit-stream, must be recognizable, permitting simple access to the data needed to rebuild the image during the scaling process, at the desired spatial resolution. Here, we the Highly Scalable-SLS (HS-SLS) algorithm is presented. The proposed algorithm is explicit in the sense that the transition to full scalability is done, within each bit-plane coding pass, by just identifying and separating the resolution levels. A individual feature of HS-SLS is that it adds, without the need of using the multiple resolution, the resolution scalability dependents lists as was done in HS-SPIHT [14]. Further, it doesn't need to process all the image pixels twice, as was done in [17], in each bit-plane pass. This means that HS-SLS doesn't involve any increase, memory requirements, in complexity and management as compared to the original SLS.

The HS-SLS coder has three facts that its work is based on. The first fact is that the once a set will be never removed when it is added to LRS, Thus if the sets are added to LRS according to a specific order. This means that the order will be preserved during the entire coding process when the sets are stored in increasing order of resolution levels, this order will be always preserved. The second fact is, in each resolution level, that the number of pixels is fixed. Let Ds denote the number of pixels in resolution level Hs. since A0 consists of NNF subband only, then D0 = (NNF×NNF). For s= 1, 2 … F, Ds is given by:

$$D_s = 3(NN_{f+1-s} \times NN_{f+1-s}) \quad (3)$$

For example, for F = 3, the number of pixels in H0 is (NN3 × NN3), and the number of pixels in H1 is 3(NN3×NN3) and so on. Thus, except the highest since it have no descendants that the sets of each resolution level, in a specific fixed size part in the LRS, can be stored. Therefore, the LRS can be thought to consist of F parts which are referred to as Resolution Part (RP). HPs has size Ds and stores the sets that belong to level s. Table(1) depicts the adopted Resolution Parts within the LRS for F= 3.

Table 1: for the LRS for F=3 is the adopted resolution parts.

HP_0	HP_1	HP_2
$(NN_3 \times NN_3)$	$3(NN_3 \times LL_3)$	$3(NN_3 \times NN_3)$

The third fact except the highest is that for a set at any resolution level, at the next level, its offspring are located. Thus, these offspring, in its assigned resolution part, can be easily stored in straight forward manner without the need to use any additional markers and without the need of extra processing to characterize the level to which the set is belongs as is done in LHS-SPIHT and HS-SPIHT algorithms. for example, a set at level H0, its offspring are located at H1 and are stored in HP1 and so on. so as to hold trajectory of the sets that are added to the different resolution parts, HS-SLS appoints two small Resolution Index Tables denominated the Start the End Index Table and (EIT) Index Table (SIT) of size F each. EIT[0] and SIT[0] are both initialized to 0, and EIT[s] and S EIT[s], 0 <s < F, are initialized to Ds-1 which is the maximum number of sets in level Hs-1. Each time a set is added to part HPs is then incremented with EIT[s]. Remember that the LRS is initialized by the (c, r) coordinates of the pixels of H0 (the NNF subband) that have offspring as type A sets. Evidently these sets are stored in RP0 and hence EIT[0] is updated to ¾ N0 because the top-left pixel of every (2×2) pixels has no descendent.

As SLS exactly the HS-SLS works except that it makes use, with the associated LRS resolution parts, of resolution dependent coding passes. That is, the HS-SLS codes all the information for each bit-plane pass that belongs to R0 and

proceeds to H1, H2 and so on. This very easily can be attained by dealing with the resolution parts of the LRS in state the whole LRS and set sufficient markers within the bit-stream to distinguish, that belongs to the different resolution levels, the information. So as to explain the adopted coding method, remember that each bit-plane coding pass processes the LRS after it has been started by coding the NNF subband. Since NNFsymbolizesH0, hence each pass starts coding the pixels in H0. Following, the resolution parts of the LRS are remedied sequentially. Here, two times is scanned every resolution part PHs. All the sets in PRs, in each scan, that are determined by SIT[s] through EIT[s] are inspected. Only the sets of type B, in the first scan, are coded in the same way as was done with SLS. All the sets in PHsare inspected in the second scan. Only the VSPs of the set's four offspring are refined when the set is of type B as given before. Otherwise, the set is tested for significance when it is of type A. (0) is outputted and nothing is done when it is still ISIG. The set's type is updated to type B set when the set is tested SIG, and status of the set's four offspring is updated as given before and each one of it is coded. Then, each one of its four offspring is added to its location in HPs+1 when the set has grandchildren (its offspring isn't in the highest resolution level) which is determined by EIT[s+1] as type A sets and EIT[s+1]correspondingly is incremented. In this way, consequently every bit-plane coding pass will be coded in increasing order of resolution levels and LRS will also stores the sets in increasing order of resolution levels.

As shown, by arranging the data that the proposed HS-SLS produced the highly scalable bit-stream without any need to additional memory management overhead and purely in increasing order of resolution levels. In contrast, due to dealing with multiple linked lists that the HS-SPIHT coders [12] has complex memory management and, as the nodes are added, each list is accessed randomly and removed to and from it continuously. That is, for each resolution level s (LISs, LSPs, and LIPs) that HS-SPIHT deals with a set of LIS, LSP and LIP lists. Further, HS-SLS exhibits a slight complexity increment; due to the two coding methods differ only, with respect to SLS by the way the sets of the LRS are ordered. Since a very slight complexity increment that the SLS coder has with respect to SPIHT [17].Thus, with respect to SPIHT that also the HS-SLS algorithm has very slight complexity increment. In contrast, the SPIHT algorithm [13] has much less complexity than LHS-SPIHT due to, all the image pixels two times in each coding pass that LHS-SPIHT must process as stated before. These lineaments are individual to the proposed HS-SLS algorithm. Unfortunately, due to the output bits within each bit-plane layer are ordered according to the resolution levels and not according to their Rate Distortion (R-D) properties, the HS-SLS coder exhibits some performance degradation with respect to SLS. More specifically, all the pixels in resolution part PHs, it processes before proceeding to the next part PHs+1. This may lead to code pixels before pixels that have higher R-D properties leading performance deterioration that have lower R-D properties. It should be noted also that the LHS-SPIHT and HS-SPIHT suffer from this limitation.

3.3.2 Scaling and Formation of Bit-stream

By the HS-SLS encoder is shown in Figure (4) represents the structure of the bit-stream generated. It consists of body of the bit-stream and main header (Hm). The main header Hm contains the number of decomposition levels (F) and the maximum bit-plane (mmax), and data (D) field such as image dimension, the image name, etc. The bit-stream consists of (mmax+ 1) layers. Every bit-plane level corresponds to each layer. Each bit-plane layer consists, by the layer body, of the layer length tag (Lm) is followed. Lm identifies, in bytes of layer m, the contribution to the total bit-stream. One for each resolution level that the body of the layer consists of (F+1) parts. The indicating the number of bytes that belong to resolution level s in each resolution part consists of a length tag (Ps). It should be noted, at the beginning of each bit-plane part that all header information is used solely by the image parser and does not need to be sent to the decoder. However, these markers may be kept within the bit-stream so as to improve the error resilience.

The HS-SLS bit-stream, at any desirable bit rate, can easily be reordered for multi-resolution decoding. An image bit-rate B bit per pixel (bpp) and at resolution s can be reconstructed from the bit-stream discard all other levels and by keeping the resolution levels 0, 1…s, i.e. levels s+1, s+2…F, in each bit-plane layer. Until the bit-rate equals to the requested bit-rate, B so the procedure is continued for each layer.

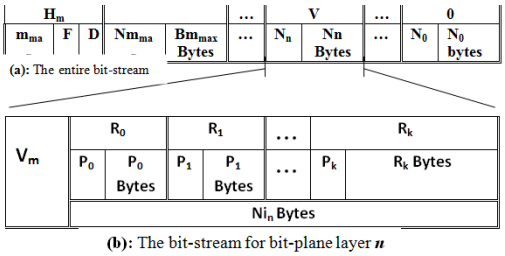

Figure 4: General structure ofbit-stream HS-SLS

4. Discussion for Experimental Results

The proposed HS-SLS algorithm is implemented by MATLAB programming language. The test is performed to the popular gray-scale (512×512) pixels 'Trees', 'Pout', 'Camera man', and 'Moon' test images shown in Figure (5). Using the bi-orthogonal 9/7 Daubechies (2-D) DWT with 5 dyadic decomposition levels is transformed for each image and the transform coefficients are rounded to the nearest integer prior to coding. The results of HS-SLS are compared with the SLS, SPIHT, LHS-SPIHT and HS-SPIHT algorithms [12]. The results are represented by the algorithm's computational complexity, its, its memory requirements, and its performance.

Figure 5: pixels test images for Gray-scale 8 bpp (512×512)

4.1. Performance

From figures 6a, 6b, 6c, and 6d that indicate the performance is measured by the Peak Signal to Noise Ratio (PSNR) versus the average number of bits per pixel (bpp) as the compression bit-rate. PSNR is given by:

$$\text{PSNR} = 10\log_{10}\frac{\text{MAX}^2}{\text{MES}}\text{dB} \quad (4)$$

where MSE is Mean-Squared Error and MAX is the maximum value of the pixel between the reconstructed and the original images defined as:

$$\text{MSE} = \frac{1}{\text{M}\times\text{N}}\sum_{c=1}^{M}\sum_{r=1}^{N}[I_0(c,r) - I_s(c,r)]^2 (5)$$

where I_s is the rebuild image, I_0 is the original image and number of pixels that is image size M × N. Clearly, larger PSNR and smaller MSE values correspond to lower of distortion.

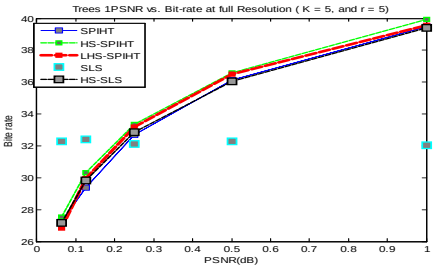

Figure 6a: PSNR (dB) vs. Bit-rate (bpp) at full Resolution (F = 5, and s= 5) for Trees

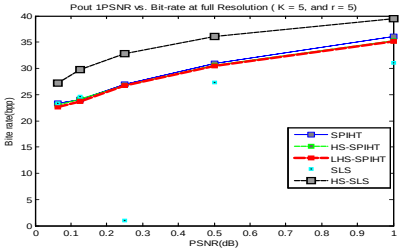

Figure 6a: PSNR (dB) vs. Bit-rate (bpp) at full Resolution (F = 5, and s= 5) for Trees

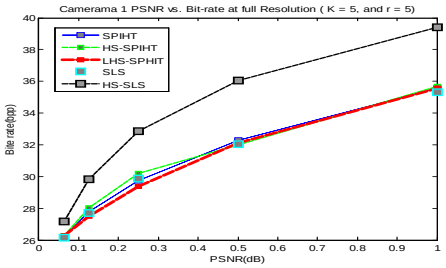

Figure 6c: PSNR (dB) vs. Bit-rate (bpp) at full Resolution (F = 5, and s= 5) for Camera man

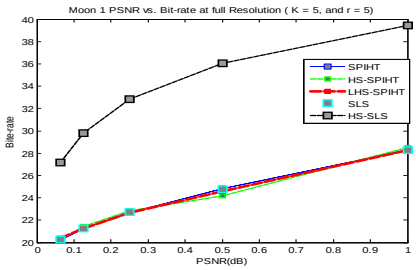

Figure 6d: PSNR (dB) vs. Bit-rate (bpp) at full Resolution (F = 5, and s= 5) for Moon

So that, returning for figures 6a, 6b, 6c and 6d can depicts the following results:

- ✓ SPIHT for 'Trees' and 'Pout' images has slightly bad PSNR than the SLS and it has comparable PSNR for 'Moon' and 'Camera man' images. This points that the state-of-art rate scalable image compression algorithms is less competitive with SLS.
- ✓ The SLS has slightly higher PSNR than The HS-SLS. This is expected in increasing order of resolution levels and not according to their rate distortion (R-D) properties due to HS-SLS doesn't preserve the information ordering as the pixels in each bit-plane coding pass are coded. Notice that this also chains is found in the LHS-SPIHT and HS-SPIHT algorithms.
- ✓ The PSNR SPIHT for 'Trees' of are lower than that of LHS-SPIHT and HS-SPIHT. Due to the above reasons, this is not possible. Moreover, due to using another version of the image, this may be occurred.

The PSNR of the proposed LHS-SLS and the HS-SPIHT are very close for 'Trees' and 'Moon images' while HS-SLS gives better PSNR for 'Pout' and 'Camera man' images especially at low bit-rates. This shows that the state-of-art highly scalable image compression algorithms has also less performance comparable PSNR with the proposed HS-SLS algorithm.

The size of the original image is $(1/2^{F-s}) \times (1/2^{F-s})$ for an image at resolution s, $0 \leq s \leq F$. Thus, due to the recovered and original images don't have the same size that a direct application of PSNR equation is not possible for s < F. Further, if the original image has higher than the recovered image resolution, so the numerical results provide of the highly scalable algorithms and too in [13] the same method adopted is used. An image at resolution s corresponds to the NNF-s subband that the fact this method is based on. So, reconstructed NNF-s subbands and the original can be compared instead of reconstructed

images and the original. Here, (M×N) and MAX symbolize the size and the maximum pixel value for the given NNF-s subbands respectively. For an 8 bpp gray-scale that the MAX value image is equal to $(255\times2^{F-s})$. This is after applying (F-s) levels of 2-D wavelet decomposition due to the fact that resolution level s is obtained from the original image, with filters having a DC amplification of $\sqrt{2}$ [4]. For example, a reconstructed image at s = 5 has the full size and MAX = 255 if a 512×512 image is decomposed with K = 5. Otherwise, at s = 4, the rebuild image has 256×256 pixels and MAX = 510. According to the number of pixels in the original full size image that the bit-rates for all levels are calculated. This enables not only to compare, for a given resolution at different bit rates, the results obtained but also to compare the results, at a given coding budget, related to different spatial resolutions. Figures (7a-9d) indicate the results of the HS-SPIHT, LHS-SPIHT, and the proposed HS-SLS algorithms at 1/4, 1/2, and 1/8 resolutions respectively. These figures pretend the PSNR notability of the proposed HS-SLS algorithm over the LHS-SPIHT algorithm for bit-rates and all scales. The only exception is for the image „Camera man" at 1/8 resolution. Further, HS-SLS is better than HS-SPIHT for „Pout" and „Moon" images which are more complex than „Trees" and „Pout"" despite of the complex memory management of the HS-SPIHT algorithm and the huge memory requirements.

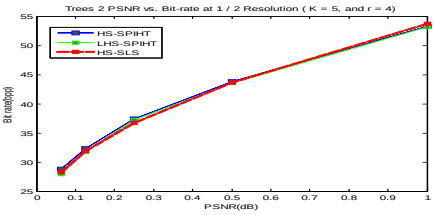

Figure 7a: PSNR vs. Bit-rate at 1 / 2 Resolution (F = 5, and s = 4) for Trees

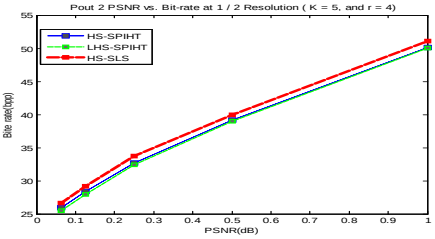

Figure 7b: PSNR vs. Bit-rate at 1 / 2 Resolution (F = 5, and s = 4) for Pout

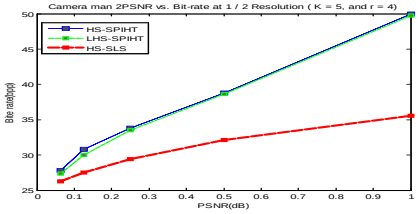

Figure 7c: PSNR vs. Bit-rate at 1 / 2 Resolution (F = 5, and s = 4) for Camera man

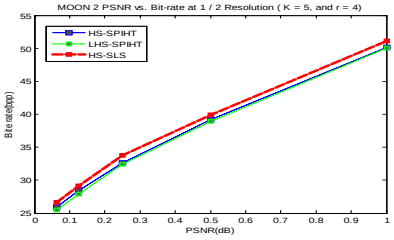

Figure 7d: PSNR vs. Bit-rate at 1 / 2 Resolution (F = 5, and s = 4) for Moon

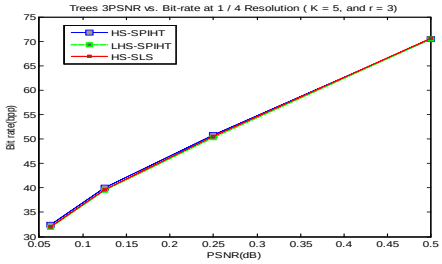

Figure 8a: PSNR vs. Bit-rate at 1 / 4 Resolution (F = 5, and s= 3) for Trees

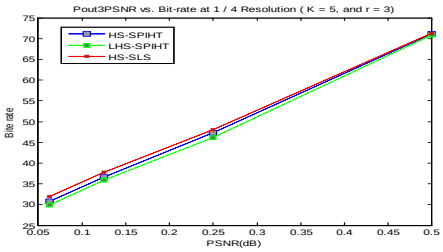

Figure 8b: PSNR vs. Bit-rate at 1 / 4 Resolution (F = 5, and s= 3) for Pout

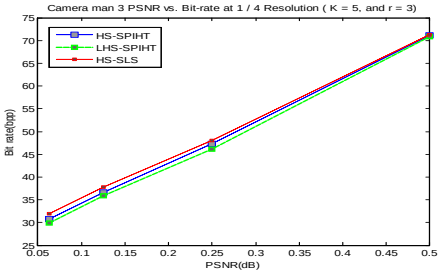

Figure 8c: PSNR vs. Bit-rate at 1 / 4 Resolution (F = 5, and s= 3) for Camera man

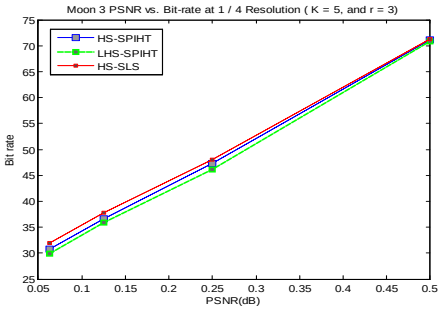

Figure 8d: PSNR vs. Bit-rate at 1 / 4 Resolution (F = 5, and s= 3) for Moon

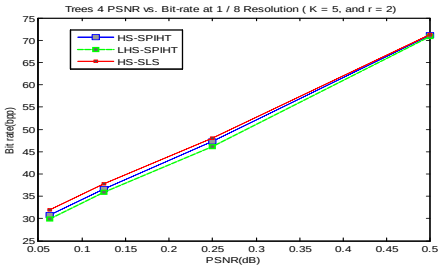

Figure 9a: PSNR vs. Bit-rate at 1 / 8 Resolution (F = 5, and s= 3) for Trees

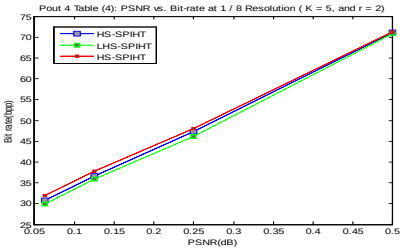

Figure 9b: PSNR vs. Bit-rate at 1 / 8 Resolution (F = 5, and s= 3)for Pout

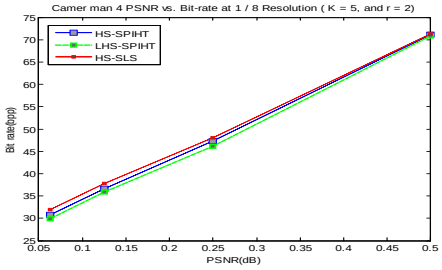

Figure 9c: PSNR vs. Bit-rate at 1 / 8 Resolution (F = 5, and s= 3)for Camera man

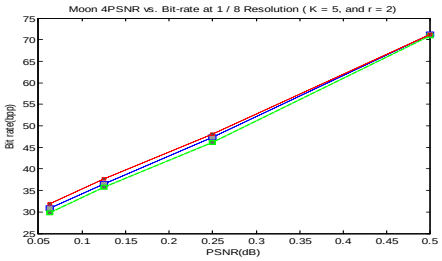

Figure 9d: PSNR vs. Bit-rate at 1 / 8 Resolution (F = 5, and s= 3)for Moon

4.2. Complexity

Previously as mentioned, due to HS-SLS need very little extra processing for encoding the resolution levels that the HS-SLS includes a slight complexity increment with respect to SLS in increasing order at the encoder and for decoding the resolution level and for the scaling process in increasing order at the decoder. Otherwise, the SPIHT has much lower complexity than LHS-SPIHT due to the SPIHT needs to process all image pixels two times per bit-plane coding pass in addition the above tasks. Figures 10a-d indicate the complexity of the algorithms exemplified by the CPU processing time (measured in milliseconds (msec)) desired by the algorithm to decode and to encode, at several bit-rates, the full scale "Trees" image. As it can be obviously demonstrated, the proposed LHS-SLS runs slower than HS-SPIHT.

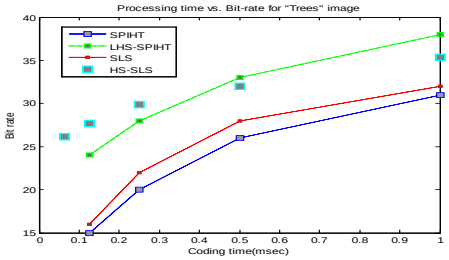

Figure 10a: Processing bite rate vs. time coding for "Trees" image

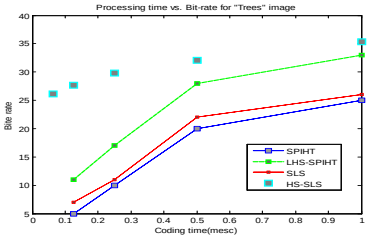

Figure 10b: Processing bite rate vs. time decoding for "Trees" image

4.3. Requirements of memory

The algorithm, at a given bit-rate, to compress the image when the amount of computer memory needed to measuring that is the memory requirement. It is worth to noting that the HS-SPIHT can use different size lists whose sizes depends on compression rate and the image size. Moreover, the using of dynamic memory allocation is from a necessity which in turn slows the algorithm. Therefore, the lists are assigned the maximum size to avoid this problem. The maximum memory of SPIHT is the same as HS-SPIHT which is given by [11]:

$$\mathrm{MEM}_{\mathrm{SPIHT}}^{\mathrm{mas}} = \mathrm{d}\left(\frac{9\mathrm{LG}}{32}\right) \text{ Bytes} \quad (6)$$

Where d is number of bits to save the pixel (r, c) coordinates, $\mathrm{d} = [\log_2(\mathrm{L})] + [\log_2(\mathrm{G})]$. The LHS-SPIHT uses, with total memory of (L × G)/2 Bytes, a fixed memory with an average of 4 bits/pixel. Lastly, the maximum memory of SLS is equals to fixed memory which is used the HS-SLS which is given by [14]:

$$\mathrm{MEM}_{\mathrm{SLS}} = \mathrm{d}\left(\frac{\mathrm{LG}}{32}\right) + \frac{\mathrm{LG}}{4} \text{ Bytes} \quad (7)$$

For example, the size is (512×512), d = 2 ×log2(512) = 18 bits for gray-scale full resolution image. Hence the total memory wanted by LHS-SPIHT, HS-SPIHT and HS-SLS are 1160 KB, 128 KB and 208 KB respectively. The HS-SLS has, with respect with LHS-SPIHT, only 80 KB of memory increment. However, HS-SPIHT still requires much greater memory than HS-SLS.

5. Conclusions

The Highly Scalable Single List SPIHT (HS-SLS) is presented in this paper. The original rate scalable SLS coder successfully is extended with the proposed HS-SLS

algorithm to a highly scalable scheme that supports combined rate scalability and resolution. By arranging the data that is implemented easily in increasing order of resolution levels using resolution-dependent parts list and resolution-dependent coding passes. Fully scalable bit-stream of the HS-SLS algorithm, for all lower spatial resolution decoding, that is flexibly using a very pure scaling process that is achieved on-the-fly and without the need to decode the main bit-stream. The LHS-SPIHT encoder has bad PSNR and higher complexity than HS-SLS as indicated from the experimental results. In many applications, the proposed scheme can be involved such as retrieval systems and image storage, and especially over heterogeneous networks when is multimedia information transmission, where a wide variety of users need to be differently serviced according data processing capabilities and to their network access. As well, very suitably for hardware implementation due to the fixed size memory usage is made with the HS-SLS algorithm. In the end, for volumetric and 3-D image compression systems are very useful the HS-SLS algorithm due to this algorithm give its benefiting from its reduced simplicity and memory.

References

[1] Mrs. Nimse Madhuri S Int." Scalable Image Encryption Based Lossless Image Compression "Journal of Engineering Research and Applications ISSN: 2248-9622, Vol. 4, Issue 10(Part - 6), October 2014, pp.51-55.
[2] W. A. Pearlman and A. Said "Image Wavelet Coding Systems: Part II of Set Partition Coding and Image Wavelet Coding Systems Foundations and Trends_ in Signal Processing" Vol. 2, No. 3 (2008) 181–246_c 2008.
[3] Yang Y., et al "Scalable image coder for mobile device" 6th International Conference on Information, Communications &Signal Processing > 1 – 5, 2007.
[4] Jie Liang, "Highly Scalable Image Coding for Multimedia Applications ACM Multimedia" 97 - Electronic Proceedings November 8-14, 1997.
[5] R. Kavitha, Sundararajan .M, Arulselvi S., "Proficient Image Compression using the Wavelet Transform and Fuzzy C-Means Clustering "International Journal of Innovative Research in Science, Engineering and Technology Vol. 4, Issue 5, May 2015.
[6] Jin Li, "Image Compression - the Mechanics of the JPEG 2000", Microsoft Research, Signal Processing, One Microsoft Way, Bld. 113/3374, Redmond, WA 98052.
[7] J. Vis. "Scalable line-based wavelet image coding in wireless sensor networks Commune. Image" R. 40 (2016) 418–431.
[8] W. A. Pearlman and A. Said "Image Wavelet Coding Systems: Part I of Set Partition Coding and Image Wavelet Coding Systems Foundations and Trends_ in Signal Processing" Vol. 2, No. 3 (2008) Vol. 2, No. 2 (2008) 95–180.
[9] A. Said and W. A. Pearlman, "A new, fast and efficient image codec based on set partitioning in hierarchical trees, " IEEE Transactions on Circuits and Systems for Video Technology, vol. 6, no. 3, pp. 243–250, June 1996.
[10] S. Tripathi, A. Ahirwar "Improved Image Compression by Set Partitioning Block Coding by Modifying SPIHT (IJCSIT)" International Journal of Computer Science and Information Technologies, Vol. 7 (5), 2016, 2152-2157
[11] Ranjan, K. et al, "Listless Block-tree Set Partitioning Algorithm for Very Low Bit-rate Embedded Image Compression," Elsevier, International Journal of Electronics and Communications, pp 985-995, 2012
[12] http://www.cipr.rpi.edu/research/SPIHT
[13] Danyali H. and Mertins A., "Flexible, Highly scalable, Object-based Wavelet Image Compression Algorithm for Network Applications," IEE Proc. Visual Image Signal processing, Vol. 151, No. 6, pp. 498-512, Dec. 2004
[14] M. Alamand E. Khan, "Listless Highly Scalable Set Partitioning in Hierarchical Trees Coding for Transmission of Image over Heterogeneous Networks", International Journal of Computer Networking, Wireless and Mobile Communications (IJCNWMC), Vol.2, Issue 3 pp. 36-48, Sep 2012.
[15] D. Taubman et al, "Embedded Block Coding in JPEG2000," Signal Processing: Image Communication Vol. 17, No. 1, pp. 49-72, Jan. 2002
[16] Ordentlich E. et al, "A Low-Complexity Modeling Approach for Embedded Coding of Wavelet Coefficients," Proc. IEEE Data Compression Conf. (Snowbird), pp. 408-417, March 1998
[17] Ali Kadhim Al-Janabi, "Low Memory Set-Partitioning in Hierarchical Trees Image Compression Algorithm", International Journal of Video & Image Processing and Network Security IJVIPNS-IJENS, Vol.13, No.02, pp. 12-18, April 2013

Author Profile

Dr. Yahya Ali Lafta was born in Hilla, Iraq, in February 1960. He graduated in engineering Electronics aviation in 1982 from academic engineering air Force Yugoslavia and received Force Yugoslavia and received the M.Sc. degree in 1984, both from the University of Belgrade, Yugoslavia. In 2012, he received the Ph.D. degree from the University of LJMU, U.K. He is currently a Lecturer at the Department electronics& communication faculty engineering of Kufa, Iraq. His research interests include video coding and networking, compressed-domain algorithms, robust coding techniques for wireless communications, and multimedia networks.