ISSN- 2350-0530(O), ISSN- 2394-3629(P)
Index Copernicus Value (ICV 2018): 86.20
DOI: https://doi.org/10.29121/granthaalayah.v7.i12.2019.312

INTERNATIONAL JOURNAL OF RESEARCH – GRANTHAALAYAH
A knowledge Repository

Science

COMPRESSION OF HIGH-RESOLUTION VOXEL PHANTOMS BY MEANS OF B+ TREE

A. Kavinilavu [1], Dr. S. Neelavathy Pari [2]
[1, 2] Department of Computer Technology, Anna University, Chennai 600044, India

Abstract

Data structures are chosen to save space and to grant fast access to data by it's key for a particular structural representation. The data structures surveyed are linear lists, hierarchical structures, graph structures. B+ tree is an expansion of a B tree data structure which allows efficient insertions, deletions and search operations. It is used to store a large amount of data that cannot be stored in the main memory. B+ tree leaf nodes are connected together in the form of a singly linked list to make search queries more efficient and effective. The drawback of binary tree geometry is that the decrease in memory use comes at the expense of more frequent memory access, might slow down simulation in which frequent memory access constitutes a significant part of the execution time. Processing and compression of voxel phantoms without loss of quality. Voxels are often utilized in the visualization and analysis of medical and scientific (logical) information. Voxel phantoms which comprise a set of small volume components appeared towards the end of the 1980s and improved on the first scientific models. These are the models of the human body. These phantoms are an extremely exact representation. Fetching of records in the equal number of disk accesses and to reduce the access time by reducing the height of the tree and increasing the number of branches in the node.

Keywords: Data Structures; B+ Tree; Processing; Compression; Voxel Phantoms; Memory.

Cite This Article: A. Kavinilavu, and Dr. S. Neelavathy Pari. (2019). "COMPRESSION OF HIGH-RESOLUTION VOXEL PHANTOMS BY MEANS OF B+ TREE." *International Journal of Research - Granthaalayah*, 7(12), 199-208. 10.29121/granthaalayah.v7.i12.2019.312.

1. Introduction

Data structures provide a means to manage large amounts of data efficiently for uses such as large databases and internet indexing services. Usually, efficient data structures are key to designing efficient algorithms. Data structures can be used to organize the storage and retrieval of information stored in both the main memory and secondary memory.

In linear data structure elements are connected in a linear fashion by means of logically or in sequence memory locations. Static memory allocation and dynamic memory allocation are the two ways to represent linear structures in memory. The possible operations are traversal, searching,

ISSN- 2350-0530(O), ISSN- 2394-3629(P)
Index Copernicus Value (ICV 2018): 86.20
DOI: 10.5281/zenodo.3597651

insertion, deletion. A linear list is a finite, linearly ordered set of elements $L = \{e_1, \ldots, e_n\}$. Used to store data that must be processed in a particular order. For example, a list may contain a lot of directions to a display processor, a set of changes to be applied to an image, or a set of points constituting a polygonal approximation of the boundary of a region. In every one of these sets, the ordering of the elements is significant. Linear lists are frequently described by the strategy for insertion and deletion of elements. A queue is a list with a specified front and end in which elements are inserted at the end and deleted from the front. The first element to enter a queue is the element to leave and to be processed. A display list is a frequently used example of a queue.
A stack is a list with a designated top and bottom in which elements are inserted and deleted at the top. The first element to enter a stack is the last element to leave. Transformation stacks are generally used with recursive picture structures. An ordered list is a list in which the elements are arranged according to a prespecified ordering function f. If the current list is $L = \{e_1, \ldots, e_n\}$, e is a new element to be inserted, and if $f(e_i) < f(e) < f(e_i+l)$, then e is inserted between e_i and e_i+l. Ordered lists are frequently used to store tables of alphanumeric elements. Ordered lists of points or line segments are also common in many image processing algorithms.

The data structure where data items are not organized consecutively is called a non-linear data structure. The data elements of the non-linear data structure could be associated with more than one element to reflect a spatial relationship among them. Each data element is a non-linear structure cannot be navigated in a single run. Instances of non-linear structures are trees and graphs. A tree is recursively defined as a set of one or mode nodes where one node is designated as the root of the tree and all the remaining nodes can be partitioned into non-empty sets each of which is a sub-tree of the root [5][10][21].

A tree is a collection of nodes where these nodes are organized progressively and structure a parent-child relationship. Numerous researchers adopt the strategy that an image is a progressive system (hierarchy). An entire picture can be broken into parts, the parts into smaller parts, and so on down to the degree of primitives in line drawings or pixels in gray-tone images. The data structure used to represent hierarchic data is a tree. A tree is a couple (N, R), where N is a finite set of elements called nodes, including one unique node are called the base (root) of the tree. $R \subset N \times N$ is a binary relation with the property that for each $n \subset N$, (n, r) is not in R, and for each $n_1 \in N$, $n_1 \neq r$, there exists one and only one node $n_2 \in N$ to such that $(n_2, n_1) \subset R$. In the event that $(n_1, n_2) \in R$, at that point n_1 is the parent of n_2, and n_2 is a child of n_1. If $n \in N$, the subtree headed by n is the arrangement of nodes $S_n = \{m \in N \mid$ there exists a grouping of nodes $n_1, \ldots, n_p$ to such an extent that $n_1 = n$, $n_p = m$, and $(n_i, n_i+1) \subset R$, $i = 1, \ldots, p-1\}$. A leaf is a node that has no children.

A line drawing representation in a tree, the root represents the whole drawing, and the subtrees headed by the two children of the root represent the significant segments. The primitive segments represent the leaves of the tree, for example, straight-line sections (segments) or curves (arcs). An AVL tree, the height of the two sub-trees of a node may vary by at most one considered. Because of this property, the AVL tree is otherwise called a height-balanced tree. A binary search tree, otherwise called an ordered binary tree, is a variation of binary trees where the nodes are arranged in an order. A binary tree is a data structure that is characterized as a collection of elements called nodes. In a binary tree, the topmost element is known as the root node, and every node has 0, 1, or at the most 2 children. Sequential linear representation of trees is done using single or one-

ISSN- 2350-0530(O), ISSN- 2394-3629(P)
Index Copernicus Value (ICV 2018): 86.20
DOI: 10.5281/zenodo.3597651

dimensional arrays is the simplest technique for memory representation, it is inefficient as it requires a lot of memory space.

A threaded binary tree is equivalent to that of a binary tree however, with a distinction in putting away the null pointers. In the linked representation, various nodes contain a null pointer, either in their left or right fields or in both. This space is squandered in storing a null pointer can be effectively used to store some other valuable snippet of data. A B tree is intended to store sorted information of data and permits search, insertion, and deletion operations to be performed in logarithmic amortized time. A B tree of order m (the most extreme number of children that every node can have) is a tree with all the properties of an M-way search tree.

A B+ tree is a data structure regularly utilized in the execution of database files (indexes). Every node of the tree contains an ordered-lists of keys and pointers to lower-level nodes in the tree. It is a balanced tree wherein each way from the root (base) of the tree to a leaf is of the same length. All information of data is stored in the leaves. Internal nodes store just keys. Each of the leaf nodes is interconnected with one another leaf nodes for quicker access. Keys are utilized for directing a search to the best possible leaf. The target key is less than a key in an internal node, at that point the pointer just to one side (left) is followed. With the condition that the target key is more prominent or equivalent to the key in the internal node, at that point the pointer to its right is followed. B+ Tree combines features of Indexed Sequential Access Method (ISAM) and B Trees. The linked list allows for rapid traversal [10].

A graph is a collection of a finite number of vertices and an edge that connect these vertices. The edges represent the connections among vertices that stores information of data elements. The data structure used to represent connections that may not be progressive (hierarchical) is the general graph structure. A graph is a pair (V, E) where V is a set of elements called nodes and the R subset of V X V is a binary relation. If the relation R is symmetric, then the graph is an undirected graph, and when (v_1, v_2) is an element of R, there is an edge connecting v_1 and v_2. For example, given a set of boundary points of objects, the connection i characterized by E_1 i E_2 if the line segment joining E_1 to E_2 is an interior line segment with respect to the boundary set, an edge is a symmetric relation. In a graph representing the relation i on some boundary set, an edge connects a pair of points if the line segment joining them is an interior segment. Such a graph has been utilized in a shape recognition algorithm [3].

The Depth-First Search (DFS) is used for traversing a finite graph. DFS traverses the depth of any particular path before exploring its breadth. It explores one subtree before returning to the current node and then exploring the other subtree. DFS uses a stack instead of a queue. It traverses a graph in a depth-ward motion and gets the next vertex to start a search when a dead end occurs in any iteration. Breadth-First Search (BFS) is used for traversing a finite graph. It visits the neighbor vertices before visiting the child vertices. BFS uses a queue for the search process and gets the next vertex to start a search when a dead end occurs in any iteration. It traverses a graph in a breadth-ward motion. It is used to find the shortest path from one vertex to another. The main purpose of BFS is to traverse the graph as close as possible to the root node. BFS is a different approach to traversing the graph nodes.

ISSN- 2350-0530(O), ISSN- 2394-3629(P)
Index Copernicus Value (ICV 2018): 86.20
DOI: 10.5281/zenodo.3597651

Processing of images to acquire the images in the format of pixels using YCbCr. It represents the color as brightness and two color-difference signals, while RGB represents the color as red, green and blue. In YCbCr, the Y represents the brightness, Cb represents blue minus (B-Y) and Cr represents red minus (R-Y) [22].

Lossless compression is highly important for images obtained at a great cost, such as space or medical images. In this case, even marginal loss of data may destroy some details required during further processing, or add artifacts that lead to erroneous interpretation. Compression of pictures with no loss of value and reduce memory space expended using the PNG compression algorithm [14][16]. Generate a tree structure using a B+ tree algorithm to store the pixels of the pictures. Branch pruning and branch canonicalization are performed to remove the undesirable or comparative pixels. Traversal of nodes to retrieve the picture stored in the database. Traversing among the nodes is simpler using a B+ tree algorithm [17][21].

2. System Architecture

2D Images are processed and compressed using appropriate algorithms. Storing and retrieval of images in and from the database using B+ - Tree data structures illustrated in Figure 1.

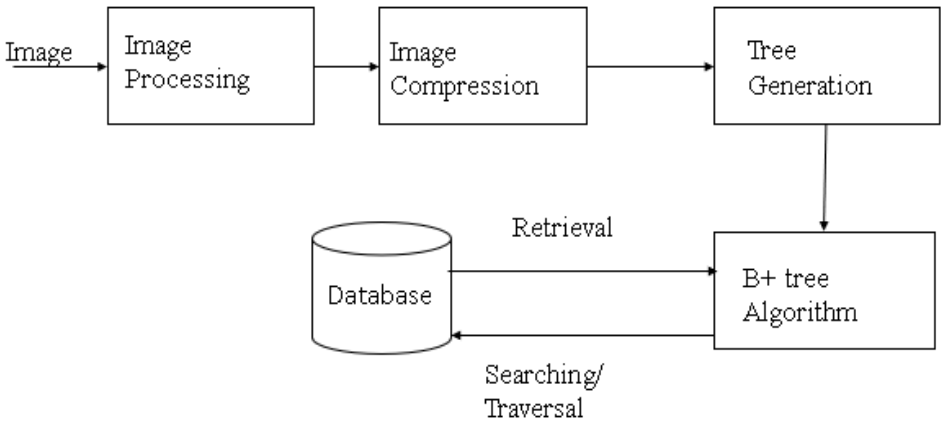

Figure 1: System Architecture

Image Processing and Compression Module
Processing of images to obtain the images in the format of pixels. Transformation of normal RGB format images to YCbCr format which is a light format. These images, when compressed does not have much difference in view by human eyes, no much loss of data. Compression of images without any loss of quality and to reduce memory space consumed.

Algorithm YCrCb
YCbCr is a color space used as part of the color image pipeline in digital photography and video systems. YCbCr is not an absolute color space.

ISSN- 2350-0530(O), ISSN- 2394-3629(P)
Index Copernicus Value (ICV 2018): 86.20
DOI: 10.5281/zenodo.3597651

Algorithm YCrCb ( )
for each image
calculate
Y = 0.2126 *r + 0.7152*g+0.0722*b
Cb = 0.5389*(blue-Y)
Cr = 0.6350*(red-Y)
end

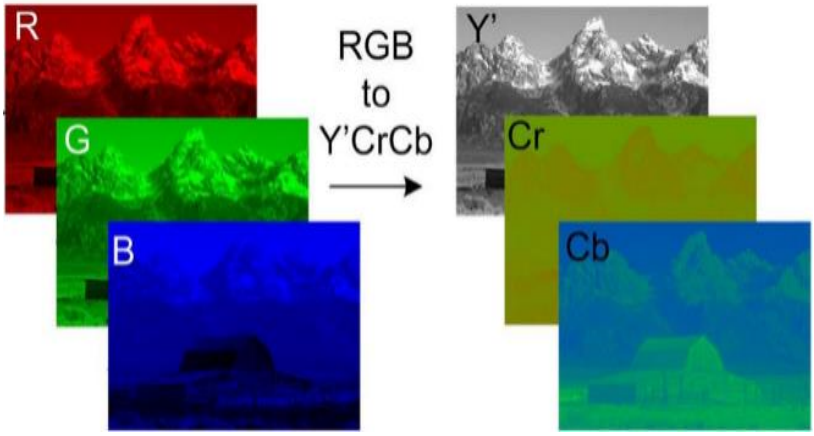

Figure 2: RGB to YCrCb Process

Figure 2 gives a clear view of how the image is processed from an RGB image to the YCrCb image.

Image Compression
JPEG compression has two phases: lossy and lossless. The lossless stage works in a typical manner, by utilizing fewer bits of code, the higher likelihood images on the rest of the information. Since the Y component is more sensitive to the human eye, it needs to be more correct and Cb and Cr are less sensitive to the human eye. It need not be more accurate. When in JPEG compression (lossy compression), it uses these sensitivities of the human eye and eliminates the unnecessary details of the image. Discrete Cosine Transform (DCT) expresses a sequence of many data points in terms of a sum of cosine functions at different frequencies.

Block-based DCT
Block-based DCT for any color image, after the pre-processing step implies expelling and RGB to YCbCr, every single one of the new three planes (Y, Cb, Cr) is partitioned into blocks. Each square is DCT transformed. Thresholding-based lossy compression calculations, two sources of data loss are encountered: The principal loss is because of the thresholding procedure. The subsequent one because of the quantization stage. The Inverse Discrete Cosine Transform (IDCT) is used to retrieve the image from its transform representation in the decoder.

ISSN- 2350-0530(O), ISSN- 2394-3629(P)
Index Copernicus Value (ICV 2018): 86.20
DOI: 10.5281/zenodo.3597651

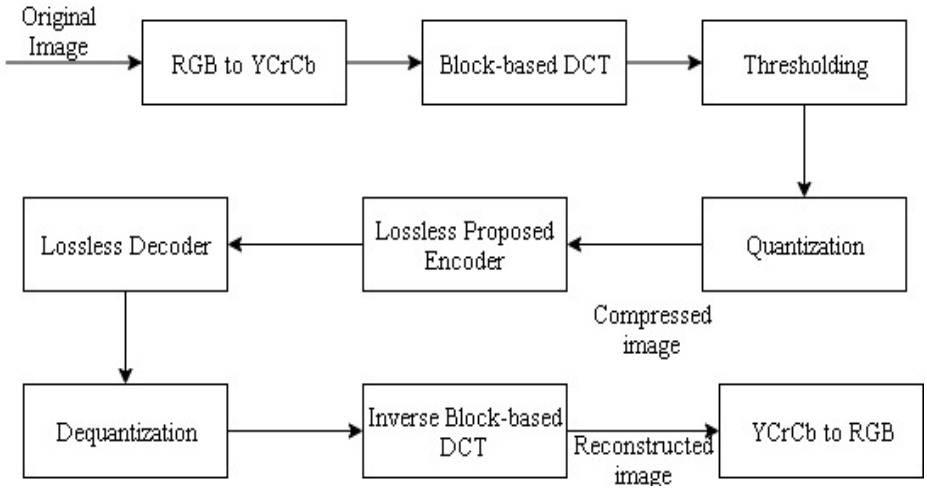

Figure 3: Image Compression and Decompression

Figure 3 explains about conversion of an original RGB image to the YCrCb image, compresses the image obtained. Reconstruct the original image using inverse block-based DCT.

Quantization

A lossy compression technique used to decrease the number of bits needed to represent the transformed coefficients. For compression of the image, truncating a portion of the coefficients will not influence or affect the others. This truncation is the lossy procedure associated with compression. For high compression, the DCT coefficients are standardized by various scales, as per the quantization matrix.

Threshold - Threshold value range for Chrominance of blue and Chrominance of red is as follows 76< a <127 and 132 < b < 173 where a =Cb, b=Cr.

Entropy Coding

Entropy coding accomplishes lossless compression by encoding the quantized DCT coefficients more minimally dependent on their statistical characteristics. Image encoding which loads multiple images as input. For each image loaded DCT operation is applied and quantized. The entropy encoding is performed and forms stream i for the i^{th} image. Stream of images is combined together to form a compressed bitstream. Decoding is performed where all the compressed bitstreams obtained from the encoding of images.

Tree Generation Module

Generate a tree structure using a B+ tree algorithm to store the pixels of the images. Allocate the new leaf and move half the buckets elements to the new bucket. Insert the new leaf's smallest key and address into the parent. If the parent is full, split it too. Add the middle key to the parent node. Repeat until a parent is found that need not split. If the root splits, create a new root that has one key and two pointers.

ISSN- 2350-0530(O), ISSN- 2394-3629(P)
Index Copernicus Value (ICV 2018): 86.20
DOI: 10.5281/zenodo.3597651

```
Algorithm insert ( )
for each node
  if node ≠ full
    insert a record
  else
    split(child)
    allocate <—new(leaf)
    new node <— move node elements
    x_child <— parent_ptr [i]
    if parent = full
      split(parent)
      parent <— add middle-key to parent
      repeat
         until a parent ≠ full
         if root = full
           split(root)
             create a new root
         endif
    endif
  endif
end
```

Retrieval of images Module

Traversal of nodes to retrieve the image stored in the database. Traversing among the nodes is easier using a B+ tree algorithm. Search queries more efficient and effective.

```
Algorithm search ( )
for each search key
     apply binary search (a record)
     if record = search key
         Return record
     else if
           current node = leaf node
           print element not found
     endif
     endif
end
```

The leaf nodes of the tree are connected together in the form of a singly linked list to make search queries more efficient and effective. A binary search on the records in the current node is performed. If a record with the search key is found, then return that record. If the current node is a leaf node and the key is not found, then report an unsuccessful search. Otherwise, follow the proper branch and repeat the process.

ISSN- 2350-0530(O), ISSN- 2394-3629(P)
Index Copernicus Value (ICV 2018): 86.20
DOI: 10.5281/zenodo.3597651

3. Implementation Details

The proposed work is implemented in anaconda3 tool using python code. The input given is in the form of images that are processed using the YCrCb algorithm in python coding and compressed using the PNG compression algorithm in python coding. The compressed image is read using python code to get the pixels which are taken as input to generate a B+ tree. Pixels with similar characteristics are stored under a single node to save space and for easy access. Based on the pixel values obtained for the RGB image in the form of red, green blue components, the B+ tree image is constructed.

4. Results

Images are processed using YCrCb processing and compressed by means of PNG compression which is given as input to reconstruct the image by means of a B+ tree.

Figure 4: Original RGB Image

Figure 4 is given as input which is processed using YCrCb processing and compressed using PNG compression to reduce the memory space.

Figure 5: YCrCb Processed image

Figure 5 is the output obtained after image processing and compression which is given as input for B+ tree generation. YCrCb processed image is easy to store in B+ tree to reduce memory and for fast access of the image.

ISSN- 2350-0530(O), ISSN- 2394-3629(P)
Index Copernicus Value (ICV 2018): 86.20
DOI: 10.5281/zenodo.3597651

Figure 6: B+ Tree Constructed image

The processed image is given as input, the pixel values are extracted for the image. Then the RGB components for each image are found to construct a B+ tree. Based on the pixel values of the components calculated, the tree is constructed. The image is retrieved by using the image id. Figure 6 is the B+ tree constructed image

5. Conclusion

The proposed project presents the basic concept of B+-trees in order to obtain compression of images to reduce the memory required and for easy retrieval. Data structures are chosen to save space and to grant fast access to data by it's key for particular structural representation. B+ tree is an expansion of B tree data structure which allows efficient insertions, deletions and search operations. B+ tree leaf nodes are connected together as an independently connected record to make search queries progressively effective. Voxels are often utilized in the visualization and analysis of medicinal and scientific(logical) information. Fetching of records in an equivalent number of disk accesses and to scale down the access time by reducing the height of the tree and increasing the number of branches in the node. In the future explore storing and retrieval of 3D images and videos. Videos are segmented into the frame for better results. The images are extracted from the videos and then processed. With high-resolution videos, the extraction of still frames can be done easily. Since high-resolution videos capture more information, frames captured are detailed, sharp, and or of higher quality to work.

References

[1] Andreu Badal and Aldo Badano (2019), 'Reducing the Memory Requirements of High-Resolution Voxel Phantoms by Means of a Binary Tree Data Structure', IEEE Transactions on Radiation and Plasma Medical Sciences, Vol. 3, No. 1, pp. 76-82

[2] Cha-Keon Cheong, Kyung-Sik Cho, and Seok-won Lee (2000), 'Significance Tree Image Sequence Coding With DCT-based Pyramid Structure', IEEE International Conference on Image Processing, pp. 859-862

[3] Dan Dolonius, Erik Sintorn, Viktor K¨ampe, and Ulf Assarsson (2019), Compressing Color Data for Voxelized Surface Geometry, IEEE Transactions on Visualization and Computer Graphics, Vol. 25, No. 2, pp. 1270-1282

[4] Debin Zhao, Ying Ki Chan, and Wen Gao (2001), 'Low-complexity And Low-memory Entropy Coder for Image Compression', IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, No. 10, pp. 1140-1145

[5] Denis V. Ivanov, Eugene P. Kuzmin, Sergey V. Burtsev (1999), 'Progressive Image Compression Using Binary Trees', International Conference Graphicon, Moscow, Russia.

ISSN- 2350-0530(O), ISSN- 2394-3629(P)
Index Copernicus Value (ICV 2018): 86.20
DOI: 10.5281/zenodo.3597651

[6] Detlev Marpe, Heiko Schwarz, Sebastian Bosse, Benjamin Bross, Philipp Helle, Tobias Hinz, Heiner Kirchhofer, Haricharan Lakshman, Tung Nguyen, Simon Oudin, Mischa Siekmann, Karsten S¨uhring, Martin Winken, and Thomas Wiegand (2010), 'Video Compression Using Nested Quadtree Structures, Leaf Merging, and Improved Techniques for Motion Representation and Entropy Coding', IEEE Transactions On Circuits and Systems for Video Technology, Vol. 20, No. 12, pp. 1676-1687

[7] Dobie and Phill Lewis (1991), 'Data structures for image processing in C', Pattern Recognition Letters, Vol. 12, pp. 457-466

[8] Donald Meagher (1982), 'Geometric Modelling Using Octree Encoding', Computer Graphics and Image Processing, Vol.19, pp.129-147

[9] Feiyu Chen, Predrag R. Bakic, Andrew D. A. Maidment, Shane T. Jensen, Xiquan Shi, and David D. Pokrajac (2015), 'Description and Characterization of a Novel Method for Partial Volume Simulation in Software Breast Phantoms', IEEE Transactions On Medical Imaging, Vol. 34, No. 10, pp. 2146-2161

[10] Gyu Sang Choi, Byung-won On, and Ingyu Lee (2015), 'PB+-tree: PCM-aware B+-tree', IEEE Transactions on Knowledge and Data Engineering, Vol. 27, No. 9, pp. 2466-2479

[11] Jiangtao Cui, Bin Xiao, Zhiliang Yin (2010), Speed up linear scan in high-dimensions using extended B+-tree, International Conference on Database Technology and Applications, pp. 1-4

[12] Jingqi Ao, Sunanda Mitra, Brian Nutter (2014), 'Fast and Efficient Lossless Image Compression Based on CUDA Parallel Wavelet Tree Encoding', IEEE International Conference on Image Analysis and Interpretation, pp. 21-24

[13] John Alan Robinson, A. Druet, and N. Gosset (2000), 'Video Compression with Binary Tree Recursive Motion Estimation and Binary Tree Residue Coding', IEEE Transactions on Image Processing, Vol. 9, No. 7, pp. 1288-1292

[14] LINDA Shapiro (1979), 'Data Structures for Picture Processing: A Survey', Computer Graphics and Image Processing, Vol. 11, pp. 162-184

[15] Li Wern Chew, Li-Minn Ang, and Kah Phooi Seng (2008), 'New Virtual SPIHT Tree Structures for Very Low Memory Strip-Based Image Compression', IEEE Signal Processing Letters, Vol. 15, pp. 389-392

[16] Marc Alzina, Wojciech Szpankowski, and Ananth Grama (2002), '2D-Pattern Matching Image and Video Compression: Theory, Algorithms, and Experiments', IEEE Transactions on Image Processing, Vol. 11, No. 3, pp. 318-331

[17] Philippe Salembier, and Luis Garrido (2000), 'Binary Partition Tree as an Efficient Representation for Image Processing, Segmentation, and InformationRetrieval', IEEE Transactions on Image Processing, Vol. 9, No. 4, pp. 561-576

[18] Riccardo Distasi, Michele Nappi, and Sergio Vitulano (1997), 'Image Compression by B-Tree Triangular Coding', IEEE Transactions On Communications, Vol. 45, No. 9, pp. 1095-1099

[19] Sho Tanimoto And Theodosios Pavlidis (1975), 'A Hierarchical Data Structure for Picture Processing', Computer Graphics and Image Processing, Vol. 4, pp. 104-119

[20] Steven Leslie Horowitz And Theodosios Pavlidis (1976), 'Picture Segmentation by A Tree Traversal Algorithm', Journal of The Association for Computing Machinery. Vol. 23, No. 2, pp. 368-388.

[21] Xiaolin Wu, and Yonggang Fang (1995), 'A Segmentation-Based Predictive Multiresolution Image Coder', IEEE Transactions on Image Processing, Vol. 4, No. 1, pp. 34-47

[22] Zixiang Xiong, Xiaolin Wu, Samuel Cheng, Jianping Hua (2003), 'Lossy-to-lossless Compression of Medical Volumetric Data Using Three-dimensional Integer Wavelet Transforms', IEEE Transactions on Medical Imaging, Vol. 22, No. 3, pp. 459-470

*Corresponding author.
E-mail address: bu_yun@ fisheries.unhas.ac.id