Spatial Complexity Reduction in Remote Sensing Image Compression by Atomic Functions

Viktor O. Makarichev, Vladimir V. Lukin, Iryna Brysina, Benoit Vozel

Spatial Complexity Reduction in Remote Sensing Image Compression by Atomic Functions

Viktor O. Makarichev, Vladimir V. Lukin, Iryna V. Brysina, Benoit Vozel

***Abstract*—Remote sensing (RS) digital images have a great variety of applications in solving real-world problems. Modern sensors provide this type of data of a very high resolution, which, in combination with a great number of acquired images, makes a problem of compressing RS-images of particular importance. In this letter, discrete atomic compression (DAC) and a problem of its spatial complexity reduction are considered. This approach provides data compression and protection features in combination with such image representation that is ready for applying different artificial intelligence methods. For this reason, its application to image processing is relevant. Several modifications that provide reducing the spatial complexity of DAC are proposed, and their efficiency is analyzed. In particular, it is shown that, using a block splitting procedure, it is possible to get a significant decrease in additional memory expenses without DAC's efficiency degradation in terms of lossy image compression.**

***Index Terms*—lossy image compression, spatial complexity, atomic functions, discrete atomic transform, discrete atomic compression.**

I. Introduction

REMOTE sensing (RS) digital images are an essential part of big data. They are widely used in land cover classification [1], agriculture [2], ecology monitoring [3], roads extraction [4], and many other applications. Increased computing power allows applying various artificial intelligence algorithms to processing and analysis of RS-images [5]. Moreover, an explosive hardware development provides huge RS-images databases (e.g., [6]). Also, modern sensors produce digital images of a very high resolution with a huge number of pixels [7]. For this reason, memory, traffic and time expenses have increased significantly. In order to solve this problem, image compression methods are applied [8, 9].

There are techniques and algorithms that can be used for compressing data of any type, whilst there exist methods that have been developed for compressing some specific data, in particular, images [8, 9]. The latter are usually more powerful.

Image compression methods can be divided into lossless and lossy. The former ones provide reconstruction of a compressed image without any distortions. The latter ones allow obtaining greater memory savings, although an image processed cannot be decompressed without loss of quality [8, 9].

A large number of different image compression techniques has been developed. Among them, JPEG occupies a special place [9]. Since its creation and wide spread in the early 1990s, many attempts have been made to replace it. In particular, the algorithm JPEG2000 was developed [9, 10]. The development of new image compression methods and algorithms does not stop. Coders such as SPIHT [11], AGU [12], ADCT [13], BPG [14], WebP [15], QOI [16] and many others have been designed.

New challenges and global problems determine the need to modify existing technologies as well as to develop new ones that meet modern requirements. First, due to an increase of cybercriminals' activity, data protection has become of special importance. RS-images are not an exception, especially, if taking into account their applications to defense technologies and systems [17]. Second, in order to decrease the thermal pollution and CO_2-emission, which are provided by data centers, green technology principles and requirements must be satisfied [18]. Third, for different reasons including a need for telecommunication networks load reduction, edge computing approaches are applied [19]. Whereas, image processing is still often carried out by devices of limited capabilities. Hence, development of low resource intensive image processing algorithms able to provide RS image compression and protection features in combination with data representation convenient for further analysis such as classification, retrieval, and recognition is relevant.

In the current paper, discrete atomic compression (DAC), which is lossy image compression algorithm [20, 21], is considered. Unlike JPEG, AGU, ADCT and WebP based on discrete cosine transform (DCT), discrete atomic transform (DAT) is applied in DAC. DAT is discrete data transform constructed using the so-called atomic functions

$$up_s(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{itx}\prod_{k=1}^{\infty}\frac{\sin^2(st(2s)^{-k})}{s^2t(2s)^{-k}\sin(t(2s)^{-k})}dt,$$

where $s \geq 1$. For the case $s = 1$, this is the famous V.A. Rvachev up-function [22]. Generally, $up_s(x)$ was introduced in [23]. In [24], it has been shown in terms of image compression efficiency that DAT is as useful as DCT, which is classic data processing tool [25]. An important advantage of DAT compared to DCT is computational complexity. It has been shown in [26] that time complexity of DAT is $O(N)$, where N

Corresponding author: B. Vozel.

Viktor O. Makarichev is with the National Aerospace University "KhAI", Kharkiv 61070, Ukraine (email: v.makarichev@khai.edu).

Vladimir V. Lukin is with the National Aerospace University "KhAI", Kharkiv 61070, Ukraine (email: v.lukin@khai.edu).

Iryna V. Brysina is with the National Aerospace University "KhAI", Kharkiv 61070, Ukraine (email: iryna.brysina@gmail.com).

Benoit Vozel is with the University of Rennes 1, France (email: benoit.vozel@univ-rennes1.fr).

is a size of the data processed. At the same time, complexity of DCT is $O(N^2)$, although fast algorithms for computing DCT provide $O(N \log N)$ [25]. For this reason, application of DAT instead of DCT could be preferable. Moreover, there is built-in data protection feature in DAC [20]. Besides, image processing by DAC allows obtaining such image representation convenient for applying various artificial intelligence methods [21]. Compared to JPEG, the algorithm DAC provides higher memory reduction for visually lossless case (for $PSNR \geq 35\ dB$ [27]). Therefore, the use of DAC in lossy compression of RS-images is promising.

DAC is usually applied to an entire image and this can cause memory problems. The aim of the current research is to study possibility to reduce the spatial complexity of the algorithm DAC. The paper novelty and contribution consist in proposing the ways to decrease memory costs. First, for the non-core case, we propose how to decrease the number of large size matrices due to their reuse. Second, for the core case, we provide spatial complexity reduction due to applying the DAT with reasonable size blocks (e.g., 512×512 pixel), and show that, on the average, performance characteristics remain the same.

II. Formulation Of The Problem

Currently, we consider 24-bit full color digital images. In this case, input data is the matrix A, each element of which specifies the following three components of the corresponding pixel: red (R), green (G) and blue (B). Denote by h a number of rows and by w a number of columns of the matrix A, i.e. h and w are, respectively, the height and width of an image processed.

According to the standard algorithm analysis approach [27], when investigating a complexity of image processing algorithm, the functions *T(h, w)* and *M(h, w)*, which are respectively time and memory costs required for its execution, should be estimated in terms of asymptotic symbols. The function *T(h, w)* is time complexity of the algorithm studied, and *M(h, w)* is its spatial complexity. In this research, we concentrate on the function *M(h, w)* that describes spatial complexity of the algorithm DAC and its modifications proposed further. In particular, we analyze a number of auxiliary matrices, which are used in the algorithm.

Recall that DAC is a lossy image compression algorithm. This means that decompression provides the image B, which does not coincide with the source image A. Further, two groups of modifications of DAC, namely core and non-core, are proposed. We call a modification non-core if it provides the same compressed file and the same reconstructed image B as the basic version of DAC. Otherwise, a modification is called core. In this case, a performance of the modified algorithm should be studied using such indicators as compression ratio (CR) and such metrics of quality loss as maximal absolute deviation (MAD), root mean square error (RMSE), peak signal-to-noise ratio (PSNR).

In this study, several non-core and core modifications of the DAC algorithm are proposed. The main task is to study them in terms of spatial complexity analysis. Also, efficiency of DAC with core modifications is compared to its basic version using lossy image compression performance indicators. We start with a description of the basic version of the DAC algorithm and analyze ways to improve it in order to reduce its spatial complexity. Then, non-core modifications are introduced. After that, one core modification is suggested and analyzed.

III. The Algorithm DAC

In Fig.1, full color image compression by DAC, is shown. An input is $h \times w$ matrix A of RGB-components. At the first step, color space transform is applied and three matrices Y, Cr and Cb of luma and chroma components, respectively, are computed. Further, these matrices are processed independently sharing the same steps. Next, DAT is applied to each of Y, Cr, Cb and the matrices Ω_Y, Ω_{Cr}, Ω_{Cb} of DAT-coefficients are computed, respectively. Then, the elements of these matrices are quantized. Finally, the quantized DAT-coefficients are encoded using a combination of Golomb codes (GC) and binary arithmetic coding (BAC). The context adaptive BAC {CABAC) is applied. Its input is a binary stream obtained by coding quantized DAT-coefficients with GC. The output of DAC is a DAC-file that contains the image compressed.

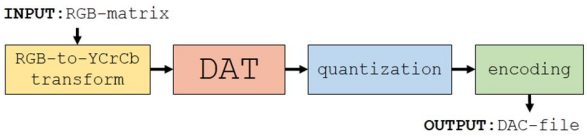

Fig. 1. DAC: full color image compression.

Consider DAC-compression in more detail, taking into account the additional memory resources, which are required in order to transform the input of the algorithm into its output. First, the RGB-to-YCrCb conversion is applied. Three $h \times w$ matrices are required for storing its result. Second, the matrix transform DAT is used. It is a discrete wavelet transform that is constructed using non-stationary infinitely differentiable wavelets with a compact support. A system of these wavelets constitutes a basis of spaces of linear combinations of up_s–function shifts. It is a set of properties of atomic functions $up_s(x)$, in particular, smoothness, compact support, as well as good approximation properties, that makes DAT a promising tool for discrete data processing and analysis [26].

DAT is performed as follows [26]. Let S be a source $h \times w$ matrix. An array DAT, which is the discrete wavelet transform based on atomic functions $up_s(x)$, is applied to each row of the matrix S and the matrix Ω_{buffer} of intermediate values is computed. Then, the array DAT is applied to each column of the matrix Ω_{buffer} and the "final" matrix of DAT-coefficients is obtained. As shown in [26], the spatial complexity of the array transform DAT is linear in the size of the array processed, i.e. it requires *O(n)* of auxiliary memory, where n is a size of an array. Sequential application of this procedure to each row of S requires *O(w)* additional memory expenses. When applying the array transform DAT to each column of Ω_{buffer}, *O(h)* of additional memory is required. Hence, spatial complexity of the matrix transform DAT is equal to *O(max(w, h))*, which is insignificant comparing to other memory costs. So, the

principal memory resources required at the second step of the DAC algorithm are three matrices Ω_Y, Ω_{Cr}, Ω_{Cb} of DAT-coefficients corresponding to *Y*, *Cr*, *Cb*, respectively, as well as the buffer matrix Ω_{buffer} (actually, the same buffer can be used in order to obtain Ω_Y, Ω_{Cr}, and Ω_{Cb}). Note that the elements of *Y*, *Cr*, *Cb*, Ω_Y, Ω_{Cr}, Ω_{Cb} and Ω_{buffer} are floating-point values. Third, the DAT-coefficients are quantized. The approach applied here is detailed in [20]. An output of this step is three matrices: $Q\Omega_Y$, $Q\Omega_{Cr}$, $Q\Omega_{Cb}$. The elements of these matrices are integers. Finally, the quantized DAT-coefficients are encoded using a combination of GC and BAC. Each element $q\omega_{ij}$ of the matrices $Q\Omega_Y$, $Q\Omega_{Cr}$, $Q\Omega_{Cb}$ is mapped to the binary code: if $q\omega_{ij} \geq 0$, then $q\omega_{ij} \leftrightarrow \underbrace{11\ldots1}_{2k-1}0$, where $k = q\omega_{ij}$; otherwise, $q\omega_{ij} \leftrightarrow \underbrace{11\ldots1}_{2k}0$, where $k = -q\omega_{ij}$. After that, the bit stream obtained is compressed using BAC. Whereas, this step can be performed for just one scan of $Q\Omega_Y$, $Q\Omega_{Cr}$, $Q\Omega_{Cb}$ with applying *O(1)* of additional memory [8], i.e. without significant memory costs. The output is DAC-file with the compressed image.

It follows that spatial complexity of DAC-compression is *O(hw)*, in particular, seven $h \times w$ matrices *Y*, *Cr*, *Cb*, Ω_Y, Ω_{Cr}, Ω_{Cb} and Ω_{buffer} of floating-point values and three matrices $Q\Omega_Y$, $Q\Omega_{Cr}$, $Q\Omega_{Cb}$ of integers are required.

In order to reconstruct the image compressed, a set of steps, each of which is inverse to the corresponding step of DAC-compression, is applied. It is clear that spatial complexity of DAC-decompression is *O(hw)* and the same matrices *Y*, *Cr*, *Cb*, Ω_Y, Ω_{Cr}, Ω_{Cb}, Ω_{buffer}, $Q\Omega_Y$, $Q\Omega_{Cr}$, $Q\Omega_{Cb}$ are required.

When processing small images, a size of auxiliary matrices is also small, and, hence, spatial complexity of DAC-compression/decompression is low. But, when operating big data, especially, large datasets of high-resolution RS-images, it occurs to be huge causing memory problems.

IV. Non-core Modifications of DAC

In this Section, non-core modifications, which do not affect compression efficiency of the DAC algorithm, are suggested.

This implies that a rather large number of $h \times w$ matrices of floating-point values are used for storing intermediate data. Our proposition is to reuse some containers.

We propose to apply the matrices *Y*, *Cr*, *Cb* for storing both luma/chroma components and the corresponding DAT-coefficients. This idea is based on the fact that just quantized values of DAT-coefficients are needed, when obtaining compressed DAC-file. Such a suggestion can be implemented as follows: $Y \xrightarrow{DAT\ rows} \Omega_{buffer} \xrightarrow{DAT\ columns} Y$, i.e. first, the array DAT is applied to each row of *Y* and the matrix Ω_{buffer} is obtained; then, the array DAT is applied to each column of Ω_{buffer} and the result of this step is stored in the matrix *Y*. The same steps are repeated for the matrices of chroma components: $Cr \xrightarrow{DAT\ rows} \Omega_{buffer} \xrightarrow{DAT\ columns} Cr$, $Cb \xrightarrow{DAT\ rows} \Omega_{buffer} \xrightarrow{DAT\ columns} Cb$. We note that the same buffer matrix Ω_{buffer} is used – no other memory is required.

It is obvious that this modification does not change the output of our algorithm, and, hence, it is non-core. Its implementation provides the following:

1) Ω_Y, Ω_{Cr}, Ω_{Cb} are not needed;

2) after performing the DAT-step, the matrices *Y*, *Cr*, *Cb* contain DAT-coefficients.

Another non-core modification is provided by features of the encoding procedure used in DAC. Note that for the used method of lossless coding of DAT coefficients no statistical information is required. Hence, quantizing and encoding can be combined and executed in a single process. In other words, when applying a scan of matrices of DAT-coefficients and quantizing their elements, it is proposed to encode the integer values computed at once and put the obtained data to the DAC-file. Therefore, the need for the matrices $Q\Omega_Y$, $Q\Omega_{Cr}$ and $Q\Omega_{Cb}$ is eliminated.

Hence, it follows that instead of seven $h \times w$ matrices of floating-point values and three $h \times w$ matrices of integers, just four $h \times w$ matrices of floating-point values are required. Whereas, an output remains unchanged.

The same modifications can be applied at decompression stage. Indeed, when decoding data, dequantization can be performed. i.e. the first two steps of decompression can be combined. As a result, the dequantized DAT-coefficients are obtained. These values are stored in the matrices *Y*, *Cr*, *Cb*. Then, three matrices of luma and chroma components are computed: $Y \xrightarrow{inv.DAT\ columns} \Omega_{buffer} \xrightarrow{inv.DAT\ rows} Y$, $Cr \xrightarrow{inv.DAT\ columns} \Omega_{buffer} \xrightarrow{inv.DAT\ rows} Cr$, $Cb \xrightarrow{inv.DAT\ columns} \Omega_{buffer} \xrightarrow{inv.DAT\ rows} Cb$. Here, inverse DAT is applied.

Finally, by performing the YCrCb-to-RGB conversion, the matrix *B* of the image decompressed is obtained. Note that the DAT processing of *Y*, *Cr, Cb* should be performed sequentially in order to implement the idea proposed.

Hence, spatial complexity reduction of DAC is achieved without compression efficiency degradation. For the modified DAC algorithm, four matrices are required. When processing high resolution RS-images, these matrices can be huge. In the next Section, core modification, which provides reducing these memory expenses, are suggested.

V. Core Modification of DAC

A. DAC with Block Splitting

Here, we propose to consider the use in DAT the so-called block splitting procedures. Processing with the block splitting means that, at the first step, an image matrix is divided into a group of blocks, and, then, all other steps are applied to each of the blocks obtained. If this approach is used, then spatial complexity is defined by memory expenses, required for processing the block of the maximal size.

Block splitting procedure is used in many DCT-based image compression algorithms: JPEG, WebP, AGU, ADCT, etc. Whereas, a size of blocks can be fixed (for example, 8×8 in JPEG and 32×32 in AGU). Also, it can be chosen dynamically depending on image content (see, for instance, the algorithm ADCT).

When applying this data compression approach in lossy compression with settings, which provide high distortions, block artifacts may be obtained. In order to remove them, various

deblocking methods are used, although it requires additional time and computational resources.

When applying DCT to image processing, a variety of acceptable block sizes is limited. One of the most important reasons is non-locality of trigonometric functions [25]. Actually, it is this feature that has led to development and studying of such locally supported functions as wavelets [25] and atomic functions [22]. We suggest to apply the block splitting in the DAC algorithm. We propose to fix the sizes *m* and *n* of the blocks; then, perform a scan of $m \times n$ blocks of RGB-matrix applying the following steps to each block:

1) RGB-to-YCrCb conversion, which provides three $m \times n$ matrices *Y*, *Cr*, *Cb* of luma and chroma components;

2) DAT of *Y*, *Cr*, *Cb*; the result is three matrices of DAT-coefficients;

3) quantizing and encoding of DAT-coefficients obtained at the previous step.

It is clear that if the non-core modifications, which are proposed in Section IV, are implemented, then just four matrices of the size $m \times n$ are required. So, the spatial complexity of DAC with block splitting is $O(4mn)$. Moreover, it does not depend on the size of the image processed.

Hence, the suggested modification provides significant reduction of memory expenses, especially, when processing high resolution images. Nevertheless, the following question arises: how does it affect compression efficiency of DAC? Further, we show that the block splitting can be applied in DAC without significant effects on its performance in terms of conventional criteria used in lossy image compression.

B. Investigation of Compression Efficiency

Now, we investigate the proposed modification of DAC. The following efficiency indicators are used:

1) compression ratio: $CR = \frac{size\ of\ source\ file}{size\ of\ compressed\ file}$;

2) maximum absolute deviation: $MAD = \max\limits_{i=1,..,N} |X_i - Y_i|$;

3) root mean square error: $RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_i - Y_i)^2}$;

4) peak signal-to-noise ratio: $PSNR = 20 \log_{10} \frac{255}{RMSE}$;

where $(X_1, \ldots, X_N)$ and $(Y_1, \ldots, Y_N)$ are source and reconstructed data respectively.

In this research, satellite images of European Space Agency are used[1]. This test dataset consists of 100 high resolution images of a total size more than 5.4 GB.

We note that a presence of a lot of small objects and sharp changes of color intensity in combination with large areas of relatively constant color is an important feature of these data.

Here, we use the same quality loss control mechanism as in [20]. In order to compare DAC and its modified version, each test image is processed by these, actually, different algorithms with different loss of quality setting. The values of CR, MAD, RMSE and PSNR are computed in each case.

In Fig. 2 and Table 1, the results obtained are given. In particular, Fig. 2 shows scatter plots of RMSE vs MAD for the test images processed by the basic version DAC and DAC with image splitting into 512×512 blocks. Also, in Table 1, a dependence of the mean value of CR on the mean value of PSNR is presented. In addition, the results of a detailed analysis of the test image processing are given. Data, including rate/distortion curves in different forms, source images, their decompressed versions, and values of the considered compression efficiency indicators, are available online[2].

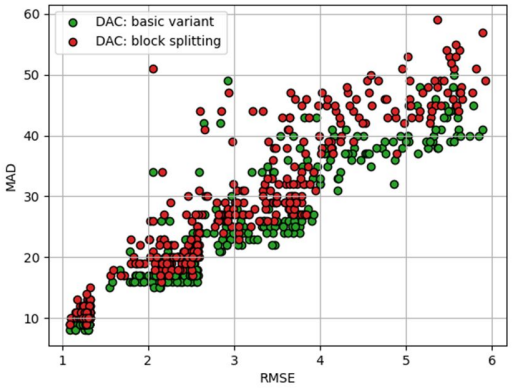

Fig. 2. Scatter plots of RMSE vs MAD.

TABLE I
DEPENDENCE OF CR ON PSNR

PSNR, dB	CR	
	DAC: basic version	DAC: block splitting
35.13	7.38	7.56
38.18	4.94	5.01
40.96	3.77	3.78
46.06	2.27	2.24

Analyzing the results given in Table 1, one can see that the basic version of DAC and DAC with image splitting into 512×512 pixel blocks provide nearly the same compression efficiency measured by CR for each value of PSNR considered.

Next, Fig. 2 indicates almost the same scatter plots of RMSE vs MAD, although, they do not coincide. In particular, the difference is minor, when applying such settings that provide small distortions measured by MAD and RMSE (it is clear that PSNR is high in this case), i.e. near lossless image compression is guaranteed. Nevertheless, a difference between the results obtained increases for greater distortions and, hence, higher CR.

So, it can be stated that the application of splitting images into 512×512 blocks to the DAC algorithm does not change its compression efficiency, when average loss of quality measured by PSNR is not less than 35 dB (actually, in this case, distortions are invisible for a human eye).

Further, the basic version of DAC and its modified version have the same time complexity $T(h, w) = O(hw)$. However, their spatial complexity is totally different: in the case of basic version of DAC, we get $M_{basic}(h, w) = O(4hw)$; when the block splitting is applied, we obtain $M_{modified}(h, w) = O(1)$, i.e. it does not depend on a size of the image compressed. If we

[1] https://www.esa.int/ESA_Multimedia/Images

[2] Results of test data processing are available at the link to Google-drive: https://drive.google.com/drive/folders/1opsSnTB07vXd3FQd4bvDBGI9ObtfbK8h?usp=sharing

combine this with the results concerning compression performance, we see that application of splitting images into 512×512 blocks for DAC can be recommended for practical use. Also, it follows that if an algorithm processes data, which are well located in memory, then higher performance might be obtained [29]. Therefore, splitting images into blocks of a size smaller than 512×512 might be preferable. Nevertheless, this may reduce an efficiency of CABAC [8], as well as image representation feature provided by DAC [21]. So, the size considered seems to be a reasonable trade-off. Finally, the basic variant of DAC and its modified version are compared using four metrics that differ significantly and verify the effectiveness of the proposed method. Indeed, MAD and PSNR quality loss metrics indicate local and average distortions, respectively. In addition, CR is a measure of compression performance. And finally, the spatial complexity, which is expressed by an asymptotic notation, indicates the memory costs required to run the algorithm.

VI. Conclusions

In this paper, a set of modifications providing spatial complexity reduction of the DAC algorithm has been proposed. It has been shown that multiple usage of matrices and combining quantization with encoding, which can be performed due to the approach applied, provides about two times decrease in auxiliary memory required for image compression and decompression. The proposed approach is purely practical, i.e. it focuses on efficient software implementation, which is especially important in processing large size images or for systems with low hardware capabilities.

Also, it has been suggested to apply the block splitting procedure in order to reduce the size of buffer matrices. The basic version of DAC has been compared to DAC with splitting image into 512×512 blocks, and it has been shown that both compressors provide nearly the same results, when processing RS-images. For this reason, application of DAC with block splitting should be recommended. Nevertheless, a block size, which guarantees better compression than 512×512, may exist. So, the following question naturally arises: what is the best one? Another question is: what is an efficiency of application of chroma subsampling that is used in many image compression algorithms, for instance, in JPEG? This will be a task for further research. Finally, it is supposed that the modifications suggested can be applied in various image compression algorithms, especially, in new ones based on other atomic functions.

References

[1] Xin-Yi Tong, Gui-Song Xia, Qikai Lu, Huanfeng Shen, Shengyang Li, Shucheng You and Liangpei Zhang, “Land-cover classification with high-resolution remote sensing images using transferable deep models,” *Remote Sens. Environ.*, vol. 237, 111322, 2019, doi: 10.1016/j.rse.2019.111322.

[2] R.P. Sishodia, R.L. Ray and S.K. Singh, “Applications of Remote Sensing in Precision Agriculture: A Review,” *Remote Sens.*, vol. 12, 3136, 2020.

[3] A.M. Lechner, G.M. Foody and D.S. Boyd, “Applications in Remote Sensing to Forest Ecology and Management,” *One Earth*, vol. 2, no. 5, pp. 405-412, 2020, doi: 10.1016/j.oneear.2020.05.001.

[4] R. Lian, W. Wang, N. Mustafa and L. Huang, "Road Extraction Methods in High-Resolution Remote Sensing Images: A Comprehensive Review," *IEEE J. Sel. Top. Appl.*, vol. 13, pp. 5489-5507, 2020.

[5] N. Kussul, M. Lavreniuk, S. Skakun and A. Shelestov, "Deep Learning Classification of Land Cover and Crop Types Using Remote Sensing Data," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 5, pp. 778-782, 2017, doi: 10.1109/LGRS.2017.2681128.

[6] The European Space Agency. *Sentinel Online*. Available: https://sentinel.esa.int/web/sentinel/sentinel-data-access.

[7] D. Olson and J. Anderson, “Review on unmanned aerial vehicles, remote sensors, imagery processing, and their applications in agriculture,” *Agronomy Journal*, vol. 113, no. 2, pp. 971-992, 2021, doi: 10.1002/agj2.20595.

[8] K.Sayood, *Introduction to data compression*, 5th ed., Morgan Kaufman, Cambridge, MA, USA, 2017, 765 p.

[9] Y.-Q. Shi and H. Sun, *Image and Video Compression for Multimedia Engineering*, 3rd ed., in CRC Press, Taylor & Francis Group, Boca Raton, FL, USA 2019, 576 p.

[10] Overview of JPEG 2000. *JPEG*. Available: https://jpeg.org/jpeg2000/index.html.

[11] Image Compression with Set Partitioning in Hierarchical Trees. *SPIHT*. Available: http://www.spiht.com.

[12] AGU - DCT Based High Quality Image Compression. *Mykola Ponomarenko Homepage*. Available: https://ponomarenko.info/agu.htm.

[13] ADCT: A NEW HIGH QUALITY DCT BASED CODER FOR LOSSY IMAGE COMPRESSION. *Mykola Ponomarenko Homepage*. Available: https://ponomarenko.info/adct.htm.

[14] BPG Image format. *Fabrice Bellard’s Home Page*. Available: https://bellard.org/bpg/.

[15] WebP Compression Techniques. *WebP*. Available: https://developers.google.com/speed/webp/docs/compression.

[16] The Quite OK Image Format for Fast, Lossless Compression. *QOI*. Available: https://qoiformat.org/.

[17] M. Shimoni, R. Haelterman and C. Perneel, "Hypersectral Imaging for Military and Security Applications: Combining Myriad Processing and Sensing Techniques," *IEEE Trans. Geosci. Remote Sens.*, vol. 7, no. 2, pp. 101-117, 2019, doi: 10.1109/MGRS.2019.2902525.

[18] V. Kharchenko and O. Illiashenko, “Concepts of Green IT Engineering: Taxonomy, Principles and Implementation,” in *Green IT Engineering: Concepts, Models, Complex Systems Architectures*, Studies in Systems, Decision and Control, Springer, Cham, 2016, vol. 74, pp. 3-19, doi: 10.1007/978-3-319-44162-7_1.

[19] Y. Mansouri and M. Ali Babar, “A review of edge computing: Features and resource virtualization,” *JPDC*, vol. 150, pp. 155-183, 2021.

[20] V. Makarichev, I. Vasilyeva, V. Lukin, B. Vozel, A. Shelestov, N. Kussul, “Discrete Atomic Transform-Based Lossy Compression of Three-Channel Remote Sensing Images with Quality Control,” *Remote Sens.*, vol. 14, 125, 2022, doi: 10.3390/rs14010125.

[21] V. Makarichev, V. Lukin, O. Illiashenko, V. Kharchenko, “ Digital Image Representation by Atomic Functions: The Compression and Protection of Data for Edge Computing in IoT Systems,” *Sens.*, vol. 22, 3751, 2022, doi: 10.3390/s22103751.

[22] V.A. Rvachev, “Compactly supported solutions of functional-differential equations and their applications,” *Russ. Math. Surv.*, vol. 45, no. 1, pp. 87-120, 1990, doi: 10.1070/RM1990v045n01ABEH002324.

[23] V.O. Rvachov and G.O. Starets, “On certain atomic functions and their applications,” *Dokl. Ukr. Acad. Sci*., Ser. A, no. 11, pp. 22-24, 1983.

[24] V. Makarichev, V. Lukin and I. Brysina, "Progressive DCT-based coder and its comparison to atomic function based image lossy compression," in *2022 IEEE 16th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*, Lviv-Slavske, Ukraine, 2022, pp. 01-06, doi: 10.1109/TCSET55632.2022.9766871.

[25] C.K. Chui and Q. Jiang, *Applied Mathematics: Data Compression, Spectral Methods, Fourier Analysis, Wavelets, and Applications*, Atlantis Press, Paris, 2013, 573 p.

[26] V. Makarichev, V. Lukin and I. Brysina, “On the Applications of the Special Class of Atomic Functions: Practical Aspects and Perspectives,” in *Integrated Computer Technologies in Mechanical Engineering*, Lecture Notes in Networks and Systems, Cham, Springer, 2021, vol. 188, pp. 42-54, doi: 10.1007/978-3-030-66717-7_4.

[27] V. Makarichev, V. Lukin, I. Brysina, B. Vozel and K. Chehdi, “Atomic wavelets in lossy and near-lossless image compression,” in *Image and Signal Processing for Remote Sensing XXVI*, 2020, vol. 11533, 1153313.

[28] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms*, 3rd ed., The MIT Press, 2009, 1292 p.

[29] R. Bryant and D. O’Hallaron, *Computer Systems: A Programmer’s Perspective*, 3rd ed., Paerson, 2015, 1128 p.