International Journal of Advance in Computer Science & its application – IJCSIA 2018
Copyright © Institute of Research Engineers and Doctors , SEEK Digital Library
Volume 7 : Issue 2 [ISSN : 2250-3765] - Publication Date: 25 June , 2018

Hyperspectral image compression using chain codes

Israel Chávez-Delgado, Hermilo Sánchez-Cruz, Juan Humberto Sossa-Azuela

***Abstract*—Usually hyperspectral images are managing by the study of many scientific areas, particularly in microscopic pollution particle content. However, researchers of this field need to handle large amount of data and study the image to recognize the type of particle appeared in environment.**

In this work we face the problem of hyperspectral image compression, by proposing chain code representations and an entropy encoder. We obtain that the most efficient chain code is F8 combined with a context-mixing algorithm.

Keywords*—chain codes, hyperspectral images, compression*

I. Introduction

Hyperspectral images contain a wealth of data, but interpreting them requires an understanding of exactly what properties of the materials we are trying to measure, and how they relate to the measurements. Traditionally experts in the search of particles in microscopy images have used the reflection, absorption and fluorescence properties, in the area of visible as in the invisible spectrum, like infrared or ultraviolet [1].

In reflected-light spectroscopy the fundamental property that we want to obtain is spectral reflectance, that is the ratio of reflected energy to incident energy as a function of wavelength. Reflectance varies with wavelength for most materials because energy at certain wavelength is scattered or absorbed to different degrees [2]. In this case sensors to obtain wavelength can be used, but we use intensities instead of wavelength because is cheaper.

One of the problems faced by the experts is the number of images to analyze, and the size of them, so one of the solutions to this problem, is compression. Thus, before searching properties we need to reduce size of the files, we cannot just analyze the complete images because depending of the number of specters that we are using it will increase the space of memory. Therefore, in this work we tackle the problem of compression data for this kind of images by using chain codes.

In this work we use a simple of concrete Stone, in four spectrum colors, blue, green, ultraviolet and violet, in Figure 1 we show the sample obtained with white light. The image was taken using a microscope Iroscope MG-321FL.

Israel Chávez-Delgado, Hermilo Sánchez-Cruz
Universidad Autónoma de Aguascalientes, Centro de Ciencias Básicas
Mexico
chavez.israel25@gmail.com, hsanchez@correo.uaa.mx

Juan Humberto Sossa-Azuela
Instituto Politécnico Nacional, Centro de Investigación en Computación
Mexico
humbertosossa@gmail.com

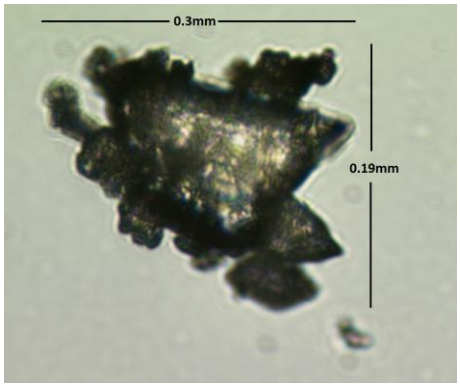


Figure 1. Concrete stone sample obtained with white light.

With the chain codes, we can obtain the information required without occupying much space in memory storage. Commonly chain codes are used to represent the contour of binary images. So, we can take advantage of this method to use it in grayscale images, to find pollution particles in microscopic samples. This pollution particles have high pixel intensity, so, these parts have a specific set of substrings of the chain codes. If we can find these substrings, we could find the pollution particles in all the images more easily.

Although, there are the Crack Codes, given by F4 [3], 3OT [4] and VCC [5], as a first approach to the representation and compression problem in hyperspectral images, we used the chain codes F8 [3], AF8 [6] and F26 [7] because they are adequate to represent grayscale images, and are more versatile, to maintain shape information given by intensities when visiting the pixel coordinates in position and intensity. This chain codes to grayscale image are richer in symbology and information, giving us the facility to find characteristics and patterns, unlike the Crack Codes that are orthogonal.

For this reasons, the chain codes we use are: F8, AF8 and F26.

Freeman's code for eight directions (F8) is introduced in 1961, it has the set of symbols F8 = {0, 1, 2, 3, 4, 5, 6, 7} denoting a movement in a counter-clockwise with an angle of 45° between each symbol-direction. See Figure 2.

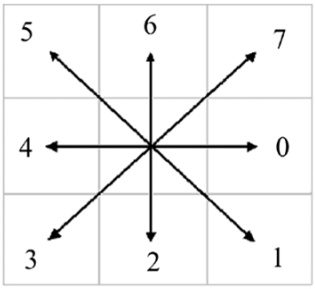


Figure 2. *The eight directions of the Freeman code.*

International Journal of Advance in Computer Science & its application – IJCSIA 2018
Copyright © Institute of Research Engineers and Doctors , SEEK Digital Library
Volume 7 : Issue 2 [ISSN : 2250-3765] - Publication Date: 25 June , 2018

AF8 was proposed by Kui and Zalik in 2005, it is based on changes obtained with every pair of F8 code vectors when following the contour [8]. This chain code is invariant under rotation. See Figure 3.

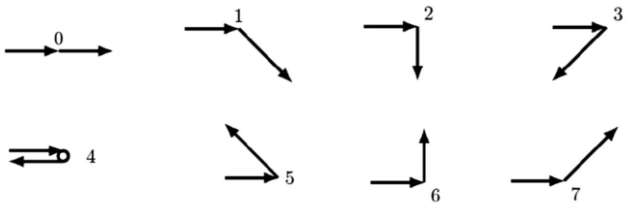


Figure 3. *The eight directions of AF8 code.*

F26 chain code works using 26 neighborhood, which means face, edge and vertex-connectivity. As with F8, this chain code is not invariant under rotation [7, 9]. See Figure 4.

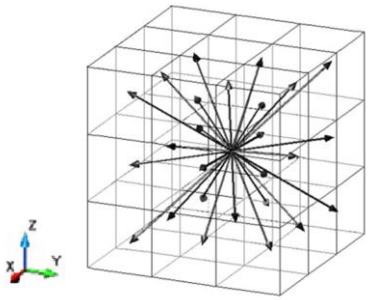


Figure 4. *The 26 directions of the Freeman code.*

A. *Compression Methods*

In this subsection we describe briefly, the different compression algorithms used in this work.

M. J. Weinberger, et al presented their LOCO-I/JPEG-LS lossless image compression algorithm, this has great compression ratios, equaling or even beating schemes based on arithmetic coding [10].

Wu and Memon introduced their Context-based, Adaptive, Lossless Image Coding (CALIC) algorithm, this was superior to its immediate predecessor, the Universal Context Modeling (UCM) method for 2% [11].

David A. Huffman developed his compression algorithm in 1952 called Huffman algorithm. This algorithm take an alphabet of n symbols along with their associated frequencies, and produces a Huffman code for that alphabet and those frequencies [12].

Arithmetic coding is a form of entropy encoding used in lossless data compression. When a string is converted to arithmetic encoding, frequently used characters will be stored with fewer bits and not-so-frequently occurring characters will be stored with more bits, resulting in fewer bits used in total [13].

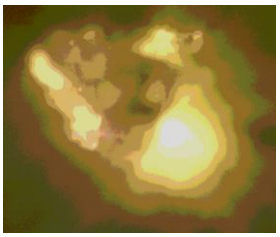
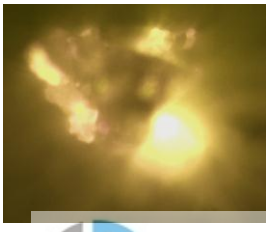
In Section 2 we explain the method used in this work, in Section 3 we present the application of our method and his respective explanation and in Section 4 we make a discussion and analysis of the results and in section.

II. Method

Before the analysis in the images is performed, we need to reduce the noise in the background. For this we tried with different filters, like, Gaussian blur, median, posterization, average blur and reduce noise. With these filters the size of the files decreased, but only one of them keep the same visual information as the original images, posterization. Which is the reduction of the tonal range of an image to a few similar colors, involves the conversion of a continuous gradation of tone to several regions of smaller number of tones [14].

In Table I we can see the behavior of the posterized images. We obtained results using posterization of 8, 16, 32, 64 and 128 gray levels to see which one is the best to analyze. We can see in the Table I, in posterization 8 and 16 the distortion can be observed easily, so this two posterized files are discarded immediately because the loss of visual information is high. In the posterization 32 the distortion is not so high, but it can be observed also; so we discarded it too. We cannot see clearly the difference between the posterized images with values 64 and 128, so we need to compare their respective chain codes files.

TABLE I. POSTERIZED IMAGES COMPARISON

Original	Posterized 8
	
Posterized 16	**Posterized 32**
	
Posterized 64	**Posterized 128**
	

International Journal of Advance in Computer Science & its application – IJCSIA 2018
Copyright © Institute of Research Engineers and Doctors , SEEK Digital Library
Volume 7 : Issue 2 [ISSN : 2250-3765] - Publication Date: 25 June , 2018

Once we obtain the posterized images we proceed to obtain the chain codes. For this we have to obtain the pixel intensity of the grayscale image. Then, we go through the values in zig-zag order, *i.e.* the first row is visited from left to right, the second from right to left and so on. But there is a little difference in how this method works with F8 and how it works with F26. The difference is that in F8 we obtain the symbols only for the values on the rows, and in F26 since it gives us the possibility to move in a 3D way, we obtain the same symbols plus extra symbols that represents a move to the next row. See Fig. 5.

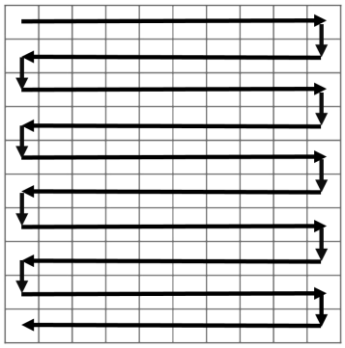


Figure 5. Obtaining chain codes in zig-zag order.

We represent the direction symbols of F26 using letters, from A to Z. In our case we don't use the 26 symbols, we only use 11 of them, eight that represent the same movements as in F8, and three that represent the movement to one row to the next. In Table II we show the movement direction of the 11 symbols we use.

TABLE II. MOVEMENT DIRECTION OF F26 USED SYMBOLS

Symbol	Movement (x, y, z)	Symbol	Movement (x, y, z)
A	(0, 1, 0)	Z	(0, 0, -1)
I	(0, 1, 1)	Q	(0, 1, -1)
Y	(0, 0, 1)	O	(1, 0, 0)
M	(0, -1, 1)	G	(1, 0, 1)
E	(0, -1, 0)	W	(1, 0, -1)
U	(0, -1, -1)		

To codify the image first we obtain its pixel intensity matrix, and start at the top-left value. With the aid of an auxiliary variable *AuxVar* with value 0 at the beginning, we compare it with the intensity *I(i, j)*, in this case *i* and *j* are 0. We follow the next algorithm.

1. If *I(i, j)* is greater, we increase *AuxVar* by one, and add the symbol 6 for F8 and Y for F26.
2. If the *I(i, j)* is equal, we compare *AuxVar* with *I(i,j+1)*.
   a. If *AuxVar* is greater by two or more, we decrease *AuxVar* by one, and add the symbol 2 for F8 and Z for F26.
   b. If *AuxVar* is greater by one, we decrease *AuxVar* by one, add the symbol I for F8 and Q for F26 and set the value of *I(i, j+1)*.
   c. If *AuxVar* is less, we increase *var* by one, add the symbol 7 for F8 and I for F26 and move to *I(I, j+1)*.
   d. If *AuxVar* is equal, add the symbol 0 for F8 and A for F26 and move to *I(I, j+1)*.

Note: We must take into account that for even rows the direction of how we obtain the chain codes is from left to right, and for odd rows the direction is from right to left, the above method is explained for and even row, so if we want to use it for an odd row we just have to change the right symbol for the left symbol.

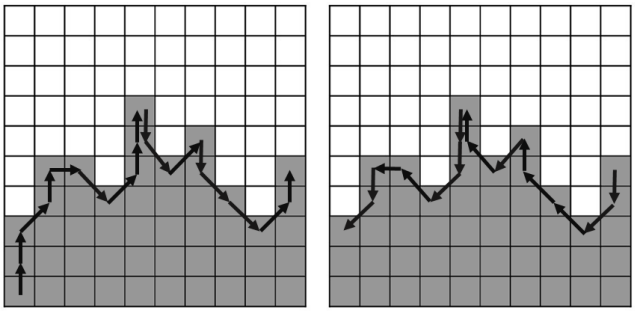


Figure 6. *Example of an odd an even row coding the intensities of an image*

In Figure 6, the example in the left is the codification of the first row, that is why it begins in the pixel 0, 0 (being 0, 0 the lower-left pixel) of the matrix, only in this case we have to start there, in the next rows we will start in the higher intensity pixel in the first column (in even rows the first column will be the left one and in odd rows it will be the right one).

In F26 we have to add a few extra steps since F26 give us the possibility to obtain the chain code of the transition from one line to another. So we will use this extra steps only if we reach the last value of the line, and we will compare *AuxVar* with the *I(i+1, j)*.

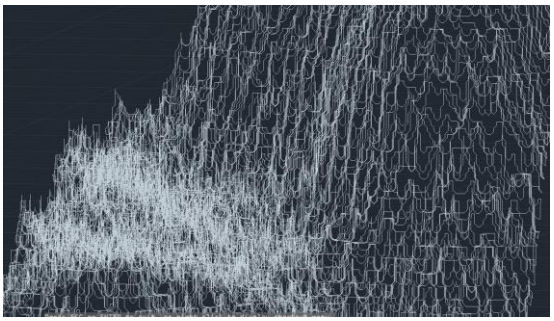
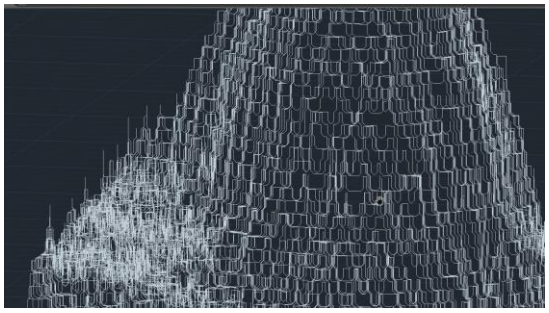
1. If *AuxVar* is greater by two or more, we decrease *var* by one, and add Z.
2. If *var* is greater by one, we decrease *var* by one, add W and move to *I(i+1, j)*.
3. If *var* is less, we increase *var* by one, add G and move to *I(i+1, j)*.
4. If *var* is equal, we add O and move to I(i+1, j).

After obtaining the chain codes we can realize that the posterization reduce the noise in the images without losing visual information and reduced peaks in the chain codes giving chains with more uniform symbols, reducing the inconsistency in these, this is a great help when we want to

obtain regular languages of sub chains, making the process classification more accurate and in less time.

We can see in Table III the behavior of the chain codes in the original and posterized image. In the original there are many peaks, and in some areas there is too much noise, and in the posterized almost all the peaks were flattened, making the noise to be reduced.

TABLE III. CHAIN CODES BEHAVIOR USING POSTERIZATION

Original	
	
Posterized 64	
	

After obtaining the chain codes we proceed to compress the chain codes with the most common compressors (Huffman, Arithmetic, JLS, CALIC and COMI). As we mentioned before, we rejected the posterization values of 32, 16 and 8, so we first proceed to compare the compression of images posterized at 64 and 128. We take into account the next step, which is apply Huffman, arithmetic and COMI algorithms. Comparing the posterized image at 128 with those of 64, the difference of the size is not that high, but about 100 Kb, hence, it has to outperform the JLS and CALIC, and the size of posterized 128 files, is smaller than those of CALIC, but is bigger than JLS files. Thus, we decide to use the posterized image at 64.

In Figure 8 we see how the posterization affect the images depending the value of posterization we use. We can observe easily that lower the value, greater the error. The error is the difference between the original image and the posterized image. Here we can realize why we chose to work with the value 64, the error in 64 is almost 0, the average value in all the images is 1.6% so we can conclude that loss of visual information cannot be seen with an error of 1.6% or less.

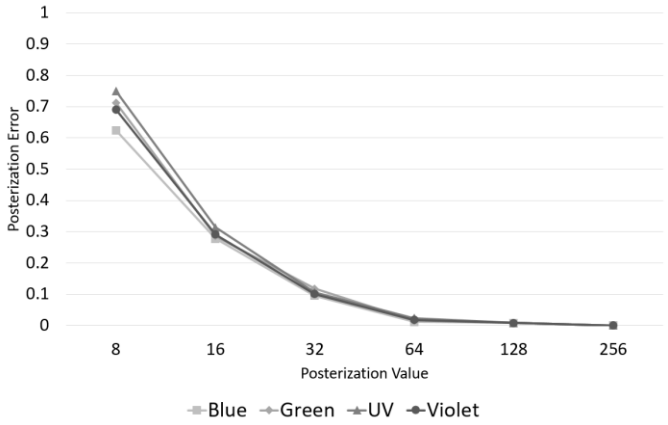


Figure 7. *Percentage of error using Posterization..*

III. Application

To apply our method, we use a sample composed of four images obtained with different spectrum lights, blue, green, ultraviolet and violet of a concrete stone, where the chain codes where obtained through the method explained in Section 2.

Below it is shown the results of the compressors applied on the chain codes, first there are Huffman and Arithmetic, Tables 4 represent the average results in bytes of these compressors. We can see that Huffman gives files with bigger size than those obtained with arithmetic, although the difference is not much. If we compare the results in the three tables we can realize that the chain code that works better with these compressors is AF8 and the best compressor is Arithmetic.

TABLE IV. F8 HUFFMAN & ARITHMETIC

File	Original	Huffman	Arithmetic	Average
F8	3,667,930	1,024,671	956,047	**1,882,883**
AF8	3,666,890	800,422	773,743	**1,747,018**
F26	3,670,538	1,174,138	1,126,666	**1,990,447**
Average	**3,668,453**	**999,744**	**952,152**	

We think that we can improve the results of the previous compressors and decided to use other compressors like CALIC and JPEG-LS and COMI that give us results of smaller size with respect to the Huffman and Arithmetic. Although JPEG-LS give better results in the Green Spectrum the expert makes the analysis in all the images, as some properties may appear in some images and in others not. Table 5 shows the results obtained using the other compressors. We used COMI for the three chain codes we are using: F8, F26 and AF8. They gave similar results in almost all spectral images, but the one that give us the best results was F8. For COMI we use the letter C.

TABLE V. COMPRESSION OF POSTERIZED IMAGES AT 64 GRAY LEVELS

	Blue	Green	UV	Violet	Average
BMP	1,303,158	1,303,158	1,303,158	1,303,158	**1,303,158**
CALIC	470,925	296,256	486,540	477,250	**432,743**
JLS	437,249	129,340	313,178	307,615	**296,846**
C. F8	276,006	239,837	308,901	294,177	**279,730**
C.AF8	281,514	244,422	314,398	299,536	**284,968**
C. F26	276,911	240,730	310,015	295,222	**280,720**

As we can observe, using posterized 64 we outperform JLS and CALIC except for the Green specter, in that case we only outperform CALIC. Something that caught our attention was that with Huffman and Arithmetic AF8 compress better than F8, but using COMI F8 is the one that compress better, we have to analyze why this happens, but as a first approximation we can say that is because Huffman and Arithmetic do not use reliance on previous symbols to compress and COMI use a combination of dependencies.

IV. Results

The main reason for image compression is to reduce the data without affecting visual information and properties of images. If we try to analyze them without reducing the noise, the chain codes strings are too big, increasing the number of symbols and the time of the analysis. In Fig.10-12 we show the behavior of the symbols before and after using posterization.

Figure 8 represents the frequency of the symbols of F8, we can see that there are two symbols that dominate over the others in both original and posterized images, 2 and 6. And observing Figure 9(b) we can see that the frequency of the symbols 0 and 4 is bigger than the original, this is because the noise in the original make many peaks, and when we posterize, those peaks are flattened making more back and forward directions.

In Figure 9, we see the frequency of the symbols of AF8 and as in Figure 8 the shapes remain in both images, the original and posterized at 64. But the difference between AF8 and F8 after posterization is that the noise reduction works better for AF8. In AF8 the reduction symbols is so high in almost all except only in one, reducing even to zero one symbol (3). The only symbol that was increased is 0, and this is because this symbol represent a straight movement no matter if it is up, down or even tilted. As in F8 the noise reduction create more flattened areas, but unlike F8, AF8 is invariant under rotation, so it has no symbols for respective directions, so one symbol can be used for many directions.

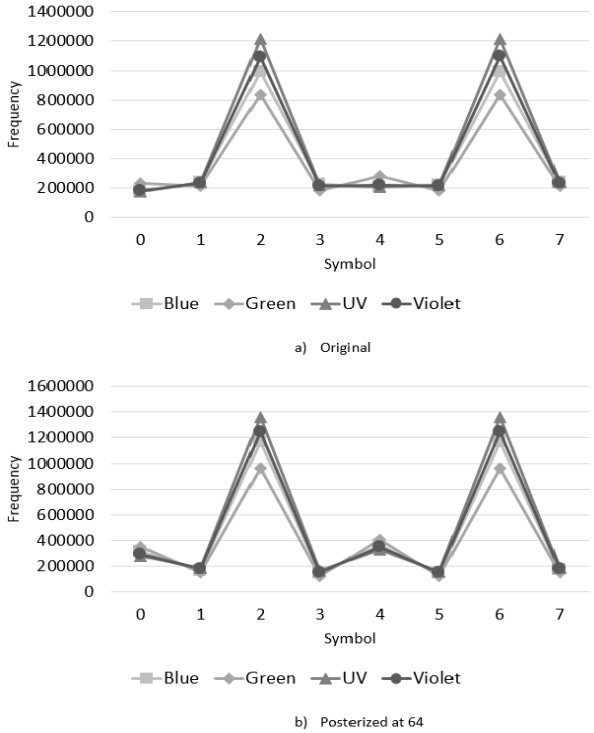


Figure 8. *Frequency of F8 symbols, Original and Posterized 64.*

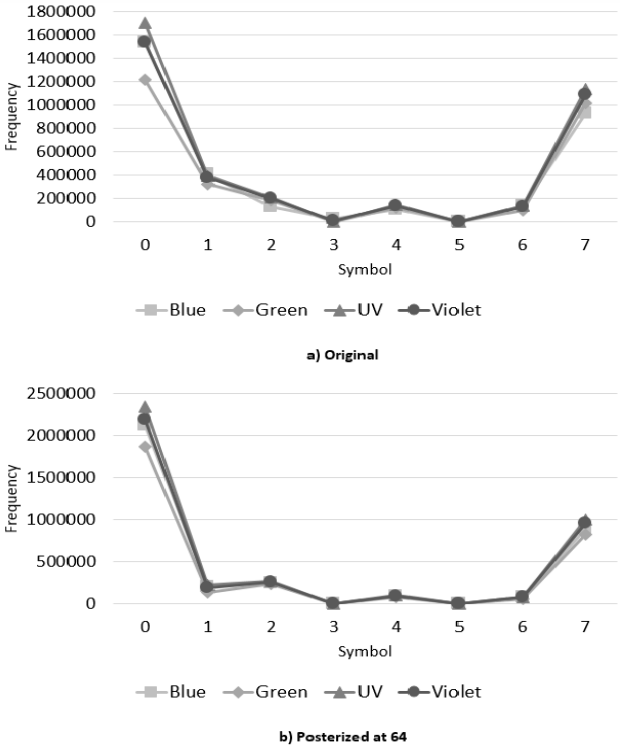


Figure 9. *Frequency of AF8 symbols, Original and Posterized 64.*

In Figures 8 and 9 we observe that they maintain the same behavior, which is the reduction of symbols that could represents peaks in the image, and the symbols representing more flat movements were increased.

v. Conclusions and future work

After following our proposing method, we can realize that a sound way to compress chain codes in hyperspectral images is using posterization and COMI, and the best value to use in posterization is 64 gray levels. As we explained, values lower than 64 produce visual loss, which it leads to loss of properties, something that is not feasible at the moment of analyze the images for experts. However, higher values than 64 obtain larger chain codes making it a more complex the analysis.

Knowing which is the best way to decrease noise in the images, now we have to find which is the best chain code to work in the analysis before the method is applied. Analyzing the behavior of the chain codes after and before noise reduction, the chain code that gives better compression is F8 using COMI, this happens because COMI use different context to compress, and the one that allows decreasing the number of symbols more than the others is AF8, keeping almost all the direction moves to two symbols, 0 and 7.

In future works we will analyze which aspect is better at the time of analysis, the one that have smaller files or the one that decrease more the number of symbols. Reducing processing time is important since we do not know how many specters we will be using, we have to make thus the process as fast as possible, taking into account that the results are also correct.

Acknowledgements

The second author thanks to Universidad Autónoma de Aguascalientes for its support under grant PIING 16-9. H. Sossa-Azuela thanks SIP-IPN under project 20170693 and CONACYT under project 65 (Fronteras of Science).

References

[1] Randall B. Smith, Introduction to Hyperspectral Imaging, MicroImages, vol I. MicroImages, 2012, pp. 3-5.

[2] A. Zehtabian, H. Ghassemian, Automatic Object-Based Hyperspectral Image Classification Using Complex Diffusions and a New Distance Metric, vol. 54, IEEE Transactions on Geoscience and Remote Sensing, 2016, pp. 4106-4108.

[3] H. Freeman, On The Encoding of Arbitrary Geometric Configurations, IRE Transactions on Electronic Computers EC-10, 1961, pp. 260-268.

[4] H. Sánchez-Cruz, R. M. Rodríguez-Dagnino, Compressing bilevel images by means of a three-bit chain code, Optical Engineering SPIE, vol. 9, 2005, pp. 1-8.

[5] E. Bribiesca, A new chain code, Pattern Recognition: The Journal of the Pattern Recognition Society, vol. 32, 1999, pp. 235-251.

[6] Y. Kui-Liu, B. Zalik, An efficient chain code with Huffman coding, Pattern Recognition, vol. 38, 2005, pp. 553-557.

[7] H. Freeman, Computer processing of line drawings, ACM Computing Surveys, vol. 6, 1974, pp. 57-97.

[8] H. Sánchez-Cruz, H. H. López-Valdez, Equivalence of chain codes, Journal of Electronic Imaging, vol. 23, 2014, pp. 1- 11.

[9] H. Sánchez-Cruz, H. H. López-Valdez, F. J. Cuevas, A new relative chain code in 3D, Pattern Recognition, vol. 47, 2013, pp. 769-788.

[10] M. J. Weinberger, G. Seroussi, G. Sapiro, The LOCO-I lossless image compression algorithm: principles and standarization into JPEG-LS, IEEE Trans. Image Process, vol. 9, 2000, pp-1309-1324.

[11] X. Wu, N. D. Memon, Context-based, adaptative, lossless image coding, IEEE Trans, Commun, vol 45, 1997, pp.437-444.

[12] D. A. Huffman, A method for the construction of minimum-redundancy codes, Proc. IRE, vol. 40, 1952, pp- 1098-1101.

[13] David J.C MacKay, Information Theory, Inference, and Learning Algorithms, Cambridge University Press, version 7.2, 2003, pp. 110-130.

[14] NIIR Board, The Complete Book on Printing Technology, Asia Pacific Business Press, 2003, pp. 186.

About Author (s):

Israel Chávez-Delgado is an engineer in Computer Systems graduated in 2014 from Universidad Autónoma de Aguascalientes (UAA) in Mexico, and is currently a master's degree student in Science with Option to Computing and Mathematics in UAA.

Hermilo Sánchez-Cruz received the PhD degree in sciences (computing) from the National Autonomous University of Mexico (UNAM) in 2002. He received the BSc degree in physics from the UNAM in 1995. He is a full time professor at the UAA in Mexico. He has collaborated on projects related to biomedical images and in recognition of Mesoamerican images. His areas of interest are pattern recognition, bidimensional and 3D image recognition, and computer vision.

Humberto Sossa was born in Guadalajara, Jalisco, Mexico in 1956. He received a B. Sc. Degree in Electronics from the University of Guadalajara in 1981, a M. Sc. In Electrical Engineering from CINEVESTAV-IPN in 1987 and a Ph. D. in Informatics from the National Polytechnic Institute of Grenoble, France in 1992. He is a full time professor at the Centre of Computing Research of the National Polytechnic Institute of Mexico. His main research interests are in Pattern Recognition, Artificial Neural Networks, Image Analysis, and Robot Control using Image Analysis.