International Journal of Advances in Computer Science & Its Applications – IJCSIA
Copyright © Institute of Research Engineers and Doctors
Volume 9 : Issue 1- [ISSN : 2250-3765] - Publication Date: 10 May, 2019

A Mapping and Sorting Hybrid Technique for Color Image Palette Extraction

Gonzalo Urcid[1] and Rocío Morales-Salgado[2]

***Abstract*—This paper introduces a hybrid computational technique that combines a vector to scalar mapping and single key numerical sorting to extract the color palette of an image coded in RGB (Red-Green-Blue) space. Thus a data set of color pixels in three dimensional integer space is mapped to an appropriate scalar data set that can be used to perform a fast numerical sort. The resulting ordered data set can be readily scanned to count all distinct colors and then remapped to its original dimensionality to obtain the image color palette. Explicit direct and inverse mappings are provided including several examples to illustrate the proposed mapping-sorting technique.**

***Keywords*— color count, color image, color palette, data sorting, multi-dimensional mappings**

I. Introduction

Counting all distinct elements in a given data set as well as obtaining the corresponding data subset of non-repeated elements is a common procedure useful in diverse areas such as, e.g., algorithmic analysis, data science or signal and image processing. In the aforementioned areas, several computational techniques are available for studying multiset mathematical manipulation [1,2], clustering trends in time series [3,4], or color image quantization and transfer [5,6].

In this paper, we focus our attention to the basic problem of extracting the color palette of a given image coded in RGB (Red-Green-Blue) space and consequently obtaining a fast count of all distinct colors without necessarily assuming that the given image has been previously quantized. For that purpose, we introduce a hybrid computational technique based on a vector to scalar mapping followed by single key numerical sorting.

Our work is organized as follows: Section II, gives the mathematical background material used in Section III, in which the mapping-sorting technique is described as applied to some color images. Finally, in Section IV we give the conclusions and a few pertinent comments to the research presented here.

1 Gonzalo Urcid
Optics Department (INAOE)
Tonantzintla 72840, Mexico

2 Rocío Morales-Salgado
Information Technology and Data Science Department, UPAEP
Puebla 72410, Mexico

II. Mathematical Background

A. *Vector to Scalar Mappings*

Let X denote an RGB color coded image whose data structure is a 3-packed matrix of numbers belonging to the integer set defined, per color channel, by the dynamic range specified by $[0, L-1]$ where L denotes the maximum number of gray levels. Then a color pixel is a triplet or vector $\mathbf{c} = (r,g,b)$ where r,g,b belong to $[0, L-1]$. The direct vector to scalar mapping or equivalently, the *scalar coding function* is given by

$$s(\mathbf{c}) = s(r, g, b) = r + Lg + (L^2+1)b\,. \tag{1}$$

The numerical coefficients assigned to each channel value are chosen to produce a gap between "redness", "greeness", and "blueness", avoiding possible overlaps of mapped values when permuting r, g, and b. If $r = b = g$, a color is the same as a gray tone. For example, color black corresponds to $r = g = b = 0$ and color white corresponds to $r = g = b = L-1$. The r, g and b coefficients in (1) are established as functions of L in order to make the number pairs $\{1, L\}$, $\{1, L^2+1\}$, and $\{L, L^2+1\}$, relatively prime. Thus, these relationships allow us to establish the inverse scalar to vector mapping or equivalently, the *vector decoding function*. The inverse mapping is then constructed as the following piecewise function for a given coded scalar s,

$$\mathbf{c}(s) = \begin{cases} (0, 0, 0) \Leftrightarrow s = 0 \\ (s, 0, 0) \Leftrightarrow s \in [1, L-1] \\ (r, (s - \quad\quad\; ^2 r) / L, \\ (\rho, (\mu - \rho) / L, \lfloor s / (L^2+1) \rfloor) \Leftrightarrow s \geq L^2+1 \end{cases} \tag{2}$$

In the third conditioned equality of (2), $r = \mathrm{mod}(s, L)$. Similarly, in the fourth conditioned equality, $\mu = \mathrm{mod}(s, L^2+1)$ and $\rho = \mathrm{mod}(\mu, L)$. Also notice the use of the integer floor function for the "blue" coordinate in the last condition of (2). As will be shown in Section III, the computer tests are realized on 24-bit color images where each color channel is an 8-bit grayscale image. Hence, $L = 2^8 = 256$ and the previous two equations are worked out numerically using, $L-1 = 255$, $L^2-1 = 65535$, and $L^2+1 = 65537$. We remark that since we are mapping three-dimensional vectors to a set of scalars, there are 6 possible scalar coding functions given by (1) by considering a different permutation of the coefficients $\{1, L, L^2+1\}$. If another permutation of $\{1, L, L^2+1\}$ is selected, the vector decoding function in (2) should be changed accordingly. The use of (1) assumes that the packed matrix X is converted to a list of triplets using row-column or column-row scanning.

SEEK DIGITAL LIBRARY

B. *Sorting and Counting*

As explained in the last paragraph of Section II, the packed matrix *X* representing a color image is first converted to a list of 3D vectors in order to be mapped, using (1), to a single scalar set whose elements will appear in random fashion and many of them possibly repeated across the whole set. Briefly stated, the result obtained with the scalar coding function is a multiset of integral values in the range $[0, L^3+L-2]$. Therefore, these integer values are interpreted as a list of single keys in a random access file that can be sorted quickly using appropriate fast algorithms.

From the many available single key internal sorting algorithms, two of the fastest are the quicksort [7] and heapsort [8] algorithms. Our selection of the heapsort algorithm is based on the fact that for a large number of elements, as in the case of color pixels in a given image, its computational complexity in the worst case (for an almost unordered list) is similar to the average case, which is not the same situation for the quicksort algorithm. The technical aspects just mentioned between quicksort and heapsort are fully explained in detail in [9]. Specifically, we use the sorting built-in functions provided by the Mathcad [10] software whose implementation follows the corresponding algorithm given in [11,12]. Recall that both quicksort and heapsort are not stable sorting algorithms, meaning that after a key list has been ordered, equal or repeated keys are brought together but do not retain necessarily their original relative order. However, since we are dealing with integer numbers, stability is not an issue for the purpose of counting distinct colors [13] and extracting an image color palette.

The simple computational procedure Distinct(**v**) with linear complexity, shown in (3), stores the first ocurrence of a given key or scalar coded pixel p in $A_{k\,0}$ (1st column) as well as the number of times c equal keys are found together in $A_{k\,1}$ (2nd column). Then, the number of rows in the output array **A** is the number of distinct elements in the sorted input vector **v** obtained using (1).

function Distinct(**v**)

$\mathbf{v} \leftarrow \mathbf{stack}(\mathbf{v}, -1)$; $m \leftarrow \mathbf{rows}(\mathbf{v})$

$k \leftarrow 0$; $p \leftarrow v_0$; $c \leftarrow 1$

for $i \in 1..m-1$

$A_{k0} \leftarrow p$; $A_{k1} \leftarrow c$; $c \leftarrow c+1$

if $p < v_i$

$p \leftarrow v_i$; $k \leftarrow k+1$; $c \leftarrow 1$

return A (3)

In (3), besides the programming reserved words **function**, **for**, **if**, and **return**, the Mathcad built-in array functions **stack** and **rows** are used to simplify the above pseudo-code. Note that, after calling this function, the first column of **A** (2nd subindex is 0) contains the *coded color palette* and variable k gives the *color count* (final value of 1st subindex).

III. Color Palette Extraction

From the explanation given previously in relation to function Distinct(**v**) displayed in (3) it is immediate that the first column of the two-dimensional output matrix **A** corresponds to the ordered list of scalar coded color pixels and serves as input to a second function displayed in (4), named BuildPal(**v**), that builds as image, although possibly smaller in size than the given color image, the palette of distinct colors.

function BuildPal(**v**)

$m \leftarrow \mathbf{rows}(\mathbf{v})$; $n \leftarrow \left\lceil \sqrt{m} \right\rceil$

for $i \in 0..n-1$

for $j \in 0..n-1$

$k \leftarrow i * n + j$

$\mathbf{u} \leftarrow \mathbf{if}(k \le m-1, \mathbf{c}(v_k), \mathbf{c}(v_{m-1}))$

$\mathbf{u} \leftarrow \mathbf{u}^T$

$R_{ij} \leftarrow u_0$; $G_{ij} \leftarrow u_1$; $B_{ij} \leftarrow u_2$

P ← **augment**(**R**, **G**, **B**)

return P (4)

Again in (4), besides the programming reserved words **function**, **for**, **if** (as function), and **return**, the Mathcad built-in array functions **rows** and **augment** are used to simplify the corresponding pseudo-code. Note that, after calling this function, array **P** is a square packed matrix whose matrix components **R**, **G**, and **B** contain, respectively, the red, green, and blue channels of the extracted *color palette*.

A. *Palette Ordering*

In digital photo, image, and graphic design software, color reduction functions are available to quantize a color image, for example, to 256 colors in order to make easy the display of the corresponding quantized color palette. Possible arrangements for quantized distinct colors are by index, luminance or hue ordering. For a given color image *X*, if p denotes the number of rows and q represents the number of columns, then the number of pixels in *X* is pq.

In this work, since the complete color palette is extracted without any quantization level, the way in which the palette is ordered is by index color, as found by scanning the color image from the top left corner pixel, (0,0),to the bottom right corner pixel $(p-1,q-1)$ and follows the specific permutation selected for the scalar coding function given in (1) of the coefficients affecting the r, g, and b channels. Also, following procedure (4), the extracted color palette is reshaped as a square image that generally is smaller in size if compared to the input color image size. The use of the ceil function applied to the square root of m (distinct colors) is the lateral dimension of the desired palette.

SEEK DIGITAL LIBRARY

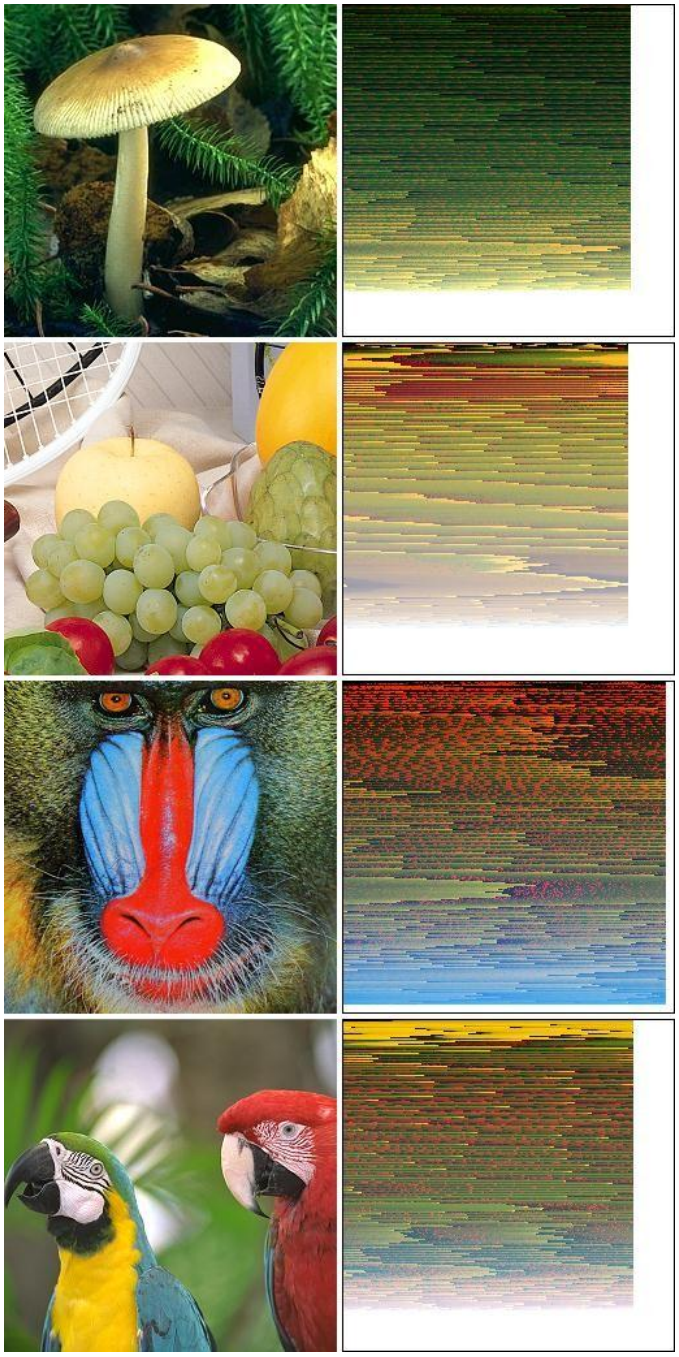

Figure 1. A set of natural color images (left) and the corresponding extracted color palettes (right)

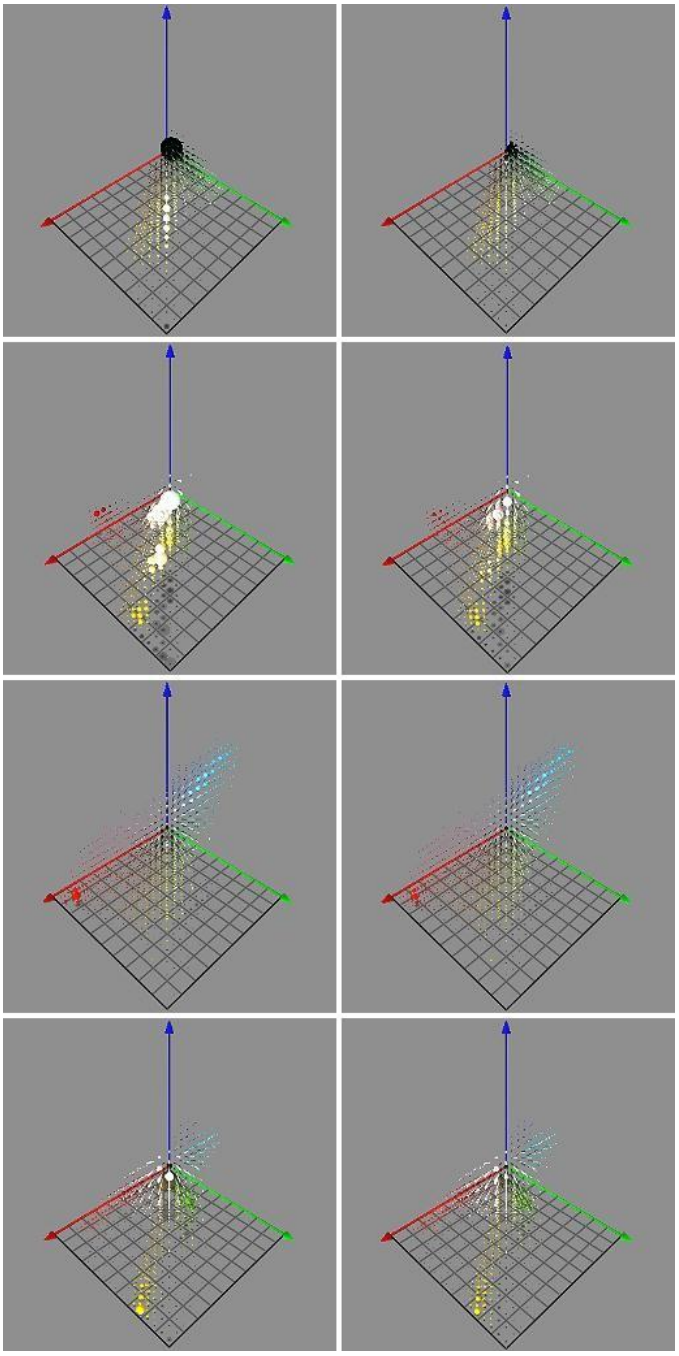

Figure 2. 3D histograms of natural color images (left) and 3D histograms of extracted color palettes (right)

B. *Computer Experiments and Results*

The mapping-sorting technique explained previously was applied to 50 public domain RGB color images of size 256 × 256 pixels. Functions (1) and (2) as well as procedures (3) and (4) were implemented in Mathcad's programming language and verified with Matlab. Figure 1 shows a subset of natural color images together with their extracted unique colors reshaped as a square image and Fig.2 depicts the corresponding 3D histograms in RGB space. In Fig. 1, the image color palette is enclosed with single pixel black border lines to emphasize its smaller size when compared to the input image size and, in Fig. 2, notice that smaller spherical neighborhoods, although barely noticeable, represent the unique colors.

Similarly, Fig. 3 shows a subset of synthetic color images with the corresponding unique colors organized as a square image and Fig. 4 depicts the associated 3D histograms in color space. In this last figure, since the number of distinct colors in a synthetic image is much less than the number of pixels contained in the original image, the size of dominant color neighborhoods are significantly reduced. The 3D histogram representation in RGB color space shown in Figs. 2 and 4 was realized using the ColorSpace software [14].

SEEK DIGITAL LIBRARY

Figure 3. A set of synthetic color images (left) and the corresponding extracted color palettes (right)

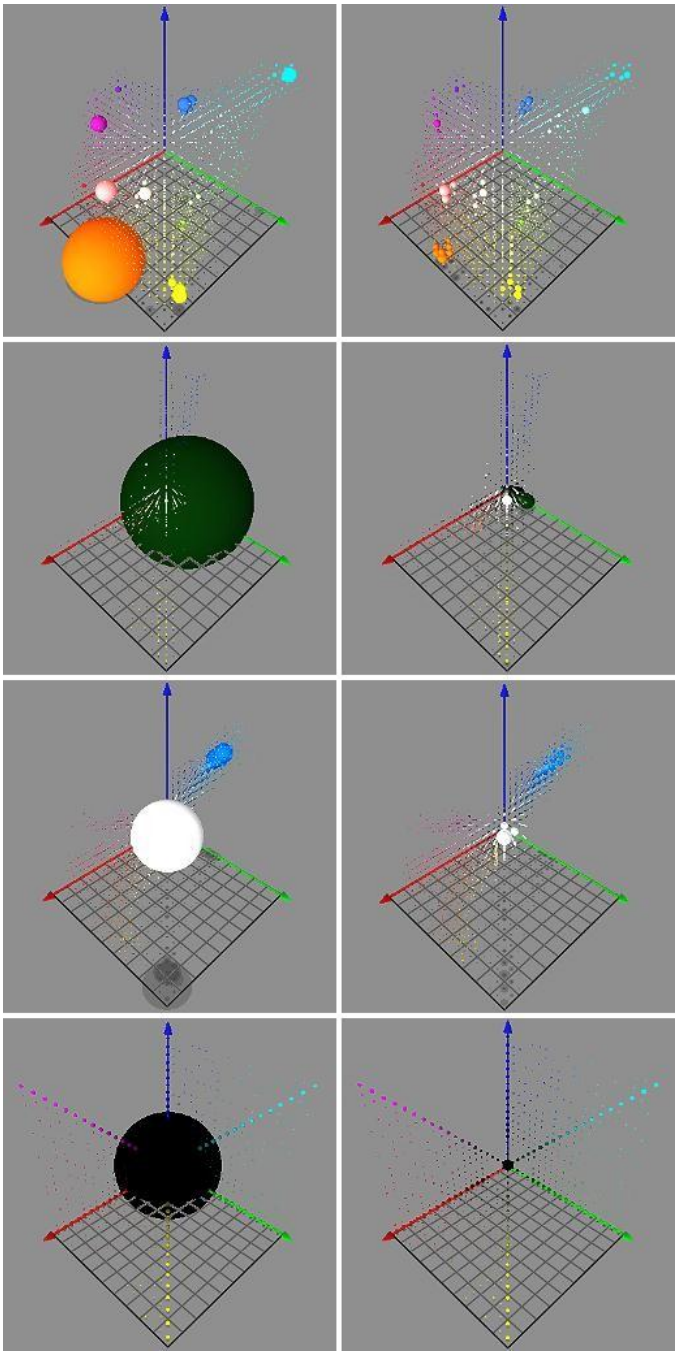

Figure 4. 3D histograms of synthetic color images (left) and 3D histograms of associated color palettes (right)

The *mean time* spent in all steps involved to count and extract the color palette was less than 750 milliseconds for each test image using a standard PC computer. Figure 5 gives a block diagram with the steps required to implement the mapping-sorting technique. The overall computational complexity is of order $m \log m$, where $m = pq$ is the number of picture elements of an input color image X.

IV. Conclusions

In this paper we have discussed a novel approach to find the proper subset of a multiset of vectors consisting of its distinct elements, using as visual examples, RGB color images as sets of color pixels in three-dimensional space. The proposed technique maps a set of vectors to a set of scalars, uses the heapsort algorithm to order them, and remaps the unique distinct scalars to extract the color palette from a given image.

Illustrative examples using public domain images were given showing their corresponding palettes and the exact color count, including the overall time spent in applying the mapping-sorting hybrid technique. Future work considers aspects, such as, the generalization of the direct and inverse mappings to higher dimensions and the realization of further tests for applications of mapping-sorting to data clustering in pattern recognition or segmentation of multispectral imagery.

SEEK DIGITAL LIBRARY

Acknowledgment

Gonzalo Urcid thanks the National Council of Science and Technology (SNI-CONACYT) in Mexico City for partial financial support through grant No. 22036.

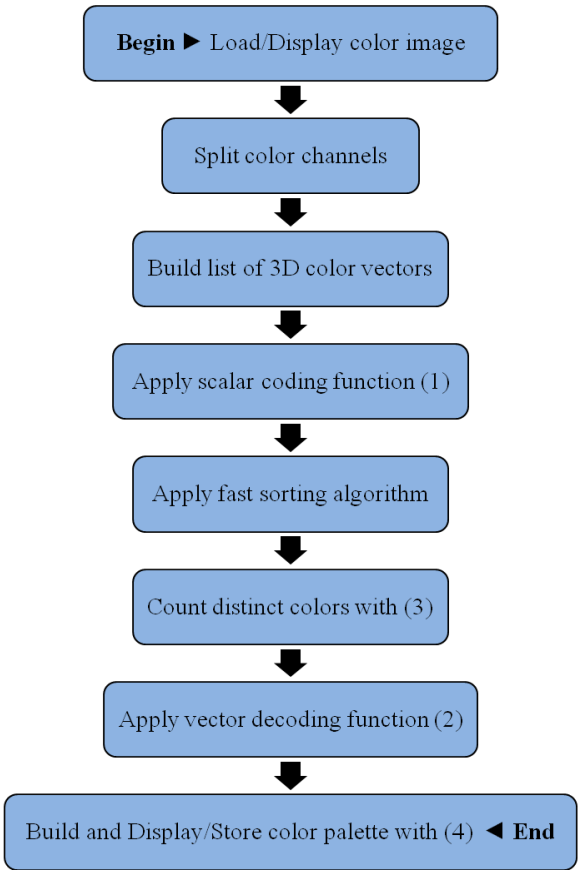

Figure 3. Flow diagram of the steps involved in the mapping-sorting technique for color count and palette extraction

References

[1] D. Knuth, "Evaluation of powers," in The Art of Computer Programming, Vol. 2 (Seminumerical Algorithms 3rd Ed.), Reading, MA: Addison Wesley, 1998, pp. 461−485.

[2] D. Knuth, "Permutations of a multiset," in The Art of Computer Programming, Vol. 3 (Sorting and Searching 2nd Ed.), Boston, MA: Addison Wesley, 1998, pp. 22−35.

[3] D. Lemire, M. Brooks, and Y. Yan, "An optimal linear time algorithm for quasi-monotonic segmentation," Proceedings of the 5th IEEE International Conf. on Data Mining, Houston, TX, November 2005.

[4] K. Aouiche and D. Lemire, "A comparison of five probabilistic view-size estimation techniques in OLAP," Proceedings of the ACM 10th International Workshop on Data Warehousing and OLAP, Lisbon, Portugal, pp. 17−24, November 2007.

[5] G. Urcid, L.D. Lara-R., and E. López-M., "A dendritic lattice neural network for color image segmentation," Proceedings of SPIE, Applications of Digital Image Processing XXXVIII, San Diego, CA, Vol. 9599, pp. 95992O:1−10, August 2015.

[6] G. Urcid, R. Morales-S., and G.X. Ritter, "Multivariate data mapping based on dendritic lattice associative memories," Proceedings of the 4th IEEE Latin American Conf. on Computational Intelligence, Arequipa, Peru, pp. 1−6, November 2017.

[7] D. Knuth, "Quicksort," in The Art of Computer Programing, Vol. 3 (Sorting and Searching 2nd Ed.), Boston, MA: Addison Wesley, 1998, pp. 113–138.

[8] D. Knuth, "Heapsort," in The Art of Computer Programing, Vol. 3 (Sorting and Searching 2nd Ed.), Boston, MA: Addison Wesley, 1998, pp. 144–158.

[9] D. Knuth, "Sorting: Summary, history, and bibliography," in The Art of Computer Programing, Vol. 3 (Sorting and Searching 2nd Ed.), Boston, MA: Addison Wesley, 1998, pp. 380–391.

[10] Mathcad 15.0 (M045), "Sorting functions," in Mathcad Help, PCT-Parametric Technology Corporation, 2015.

[11] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, "Quicksort and Heapsort," in Numerical Recipes, The Art of Scientific Computing, 3rd Ed., New York, NY: Cambridge University Press, 2007, pp. 423–428.

[12] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, "Sorting and Order Statistics," Part II in Introduction to Algorithms, 3rd Ed., Boston, MA: The MIT Press, 2009, pp. 147−228.

[13] D. Lemire, "Counting exactly the number of distinct elements: sorted arrays vs hash sets?," at https://lemire.me/blog/2017/05/23/counting-exactly-the-number-of-distinct-elements-sorted-arrays-vs-hash-sets/, WordPress, 2017.

[14] P. Colantoni, "3D color space visualization," in ColorSpace User Manual Ver. 1.0, 2004, pp. 6−9.

About Author (s):

Gonzalo Urcid received his B.Eng. (1982) and M.Sc. (1985) both from the University of the Americas in Puebla (UDLAP), Mexico, and his Ph.D. (1999) in Optical Sciences from the National Institute of Astrophysics, Optics, and Electronics (INAOE) in Tonantzintla, Mexico. Since 2001 is a National Researcher from the Mexican National Council of Science and Technology (SNI-CONACYT), and currently is an Associate Professor in the Optics Department at INAOE. His research interests include digital processing of multichannel images, artificial lattice neural networks, and pattern recognition.

Rocío Morales Salgado received her B.Eng. (1994) and M.Ed. (1998), respectively, from the Autonomous University of Puebla (BUAP) and the Iberoamerican University (UIA). She obtained her Ph.D.(2006) in Information Technologies and Decision Analysis by Puebla State Popular University (UPAEP). Her research interests are software engineering, data science, and e-business. She is currently Chair and Professor of the graduate programs in Information Technology and Electronic Business as well as the Master's Degree in Data Science and Business Intelligence at UPAEP.

SEEK DIGITAL LIBRARY