Entropy Quad-Trees for High Complexity Regions Detection

Rosanne Vetro, Dan A. Simovici, Wei Ding

University of Massachusetts Boston, Department of Computer Science

Abstract

This paper introduces entropy quad-trees, which are structures derived from quad-trees by allowing nodes to split only when those correspond to sufficiently complex sub-domains of a data domain. Complexity is evaluated using an information-theoretic measure based on the analysis of the entropy associated to sets of objects designated by nodes. An alternative measure related to the concept of box-counting dimension is also explored. Experimental results demonstrate the efficiency of entropy quad-trees to mine complex regions. As an application, we used our proposed technique in the initial stage of a crater detection algorithm using digital images taken from Mars surface. Additional experimental results are provided that demonstrate the crater detection performance and analyze the effectiveness of entropy quad-trees for high-complexity regions detection in the pixel space with significant presence of noise. This work is focused on 2-dimensional image domains, but can be generalized to higher dimensional data.

Keywords: *Entropy, Box-Counting Dimension, Quad-trees, Circular Hough Transform*

Entropy Quad-Trees for High Complexity Regions Detection

The concept of complexity relates to the presence of variation. In science there are many approaches that characterize complexity. A variety of scientific fields have dealt with complex mechanisms, simulations, systems, behavior and data complexity as those have always been a part of our environment. In this work, we focus on the topic of data complexity which is studied in information theory. While randomness is not considered complexity in certain areas such as those related to the study of complex systems, information theory tends to assign high values of complexity to random noise. Many fields benefit from the identification of content or noise related complex areas. In data hiding, adaptive steganography takes advantage of high concentration of self-information on high complexity areas originated from both content and noise to embed data. The authors of [1] describe the benefits of selective embedding related to the reduction of perceptual degradation for transform domain steganographic techniques. Bio diversity is another area where complexity can be used for identification and localization of different species. In this case, the complexity originated from content is more important than the one originated from noise.

Our goal in this paper is to introduce a variant of quad-trees for mining high complexity sub-domains of a data domain. A *quad-tree* is a tree structure defined on a finite set of nodes that either contains no nodes or is comprised of a root node and 4 quad-subtrees. In a full quad-tree, each node is either a leaf or has degree exactly 4. Our variant of quad-trees requires that each node that has descendants corresponds to a region that has a sufficient level of diversity as assessed by the value of an information-theoretical measure. We also present an alternative measure that has its roots in fractal geometry where the so called box-counting dimension (BCD) is used to determine the fractal dimension of a set S in a Euclidean space R^n. We then provide an algorithm to capture high complexity areas of 2 dimensional images domains and observe that diversity originated from both data content and noise are mined.

As an application, we used our proposed data structure in the initial stage of a crater detection algorithm. The algorithm is composed by two methods. The first method uses an information-theoretical approach with entropy quad-trees to create an edge filter that generates a binary image from complex areas which may contain edges. The second method applies a Circle Hough Transform (CHT) with modified threshold to detect the presence of circular shapes in complex areas. The new threshold is imposed to increase the quality of the results given the lack of prior knowledge about the number of craters in an image and the difficulty to estimate a good threshold for the minimum number of votes required in the parameter space to indicate true center points. Efficient methods for crater detection such as [2], [3] and many others referenced by the authors of [4] have been proposed. We provide a distinct approach where no external pre-processing of the original image other than conversion to the JPEG format and resizing is needed. Likewise, no external image filters are used.

In the next section we introduce a framework for the rest of the paper. The notion of entropy associated to a partition is presented as well as its usefulness in measuring diversity. In section 2 we introduce the proposed data structures, an algorithm for high complexity detection and explain the searching process. In subsection 2.1 we describe the information-theoretic method used for mining complex sub-domains. An alternative method uses the concept of box-counting dimension and is introduced in subsection 2.2. We provide a brief description about implementation details in subsection 2.3. In subsection 2.4 we discuss the experiments and compare the results generated by both methods. In Section 3 we introduce the crater detection algorithm. In subsection 3.1 we describe the information theoretic method used for mining complex subareas that may contain edges. The CHT method with modified threshold is described in subsection 3.2. Subsection 3.3 contains a description of the experiments and major challenges we faced. Finally, Section 4 contains our conclusions and ideas for future work.

1. Partitions, Entropy, and Trees

The notion of entropy quantifies the uncertainty associated with probability distributions. Let S be a finite set. A *partition* on S is a non-empty collection of non-empty subsets of S, $\pi = \{B_1,...,B_n\}$ such that

(i) $B_i \cap B_j = \emptyset$ for $1 \leq i, j \leq n$ and $i \neq j$;

(ii) $\cup\{B_i \mid 1 \leq i \leq n\} = S$.

The sets $B_1,...,B_n$ are referred to as the *blocks* of π.

We denote by Part(S) the set of partitions of S. For $\pi, \sigma \in$ Part(S) define $\pi \geq \sigma$ if each block B of π is a union of blocks of σ. It is well-known that the relation "$\geq$" is a partial order on Part(S). The largest partition on S is the single-block partition $\omega_S = \{S\}$, while the smallest partition on S is $\iota_S = \{\{x\} \mid x \in S\}$.

We now define a partial order relation $\geq_k$ on Part(S) as follows. If $\pi = \{B_1,...,B_n\}$ and $\sigma = \{C_1,...,C_m\}$, then $\pi \geq_k \sigma$ if the following conditions are satisfied:

1. there exists a sub-collection of σ that consists of k blocks $\{C_{j_1},...,C_{j_k}\}$ such that $\cup\{C_{j_\ell} \mid 1 \leq \ell \leq k\}$ is a block B_h of π;
2. for $1 \leq i \leq n$ and $i \neq h$, B_i is a block of σ.

For k=2 the relation $\geq_2$ is the direct coverage relation, where the larger partition π is obtained by fusing two blocks of σ.

If π ∈ Part(S) and $\pi = \{B_1,...,B_n\}$, its entropy is the number

$$H(\pi) = -\sum_{i=1}^{n} \frac{|B_i|}{|S|} \log_2 \frac{|B_i|}{|S|}$$

which is actually the entropy of the discrete probability distribution

$$\boldsymbol{p} = \left(\frac{|B_1|}{|S|}, \dots, \frac{|B_n|}{|S|}\right)$$

Defining the entropy for partitions rather than for probability distributions has the advantage of linking the entropy properties to the partially ordered set of partitions. An important fact is that the entropy is anti-monotonic relative to the partial order defined on partitions. In other words, for π, σ ∈ Part(S), π ≤ σ implies H(π) ≥ H(σ). It is easy to verify that $H(\omega_S)=0$ and that $H(\iota_S) = \log_2 |S|$. This shows that the entropy can be used to evaluate the uniformity of the elements of S in the blocks of π since the entropy value increases with the uniformity of the distribution of the elements of S. Note that as the uniformity increases, so does the associated uncertainty.

If C is a non-empty subset of S, and π ∈ Part(S), the *trace* of π on C is the partition

$$\pi_C = \{B \cap C \mid B \in \pi \text{ and } B \cap C \neq \emptyset\}.$$

The trace of a partition allows us to define the conditional entropy of two partitions. Namely, if π, σ ∈ Part(S) and $\sigma = \{C_1,...,C_m\}$, then the *entropy of* π *conditioned by* σ is the number

$$H(\pi|\sigma) = \sum_{j=1}^{m} \frac{|C_j|}{|S|} H(\pi_{C_j})$$

It can be shown [5, 6] that the conditional entropy is an anti-monotonic function of the first argument and a monotonic function of the second. In other words, $\pi_1 \leq \pi_2$ implies $H(\pi_1|\sigma)$

$\geq H(\pi_2|\sigma)$ and $\sigma_1 \leq \sigma_2$ implies $H(\pi|\sigma_1) \leq H(\pi|\sigma_2)$.

A *measure* on S is a function $m : P(S) \to R_{\geq 0}$ such that $m(U \cup V) = m(U) + m(V)$ for every disjoint subsets U and V of S. For example, if S is the set of pixels of a gray image S, m(U) can be defined as the number of pixels having a certain degree of grayness contained by the subset U.

Let D be a finite set. A D-feature function on S is a function $f : S \to D$. Each feature function $f : S \to D$ defines a partition ker f on S defined by

$$\ker f = \{f^{-1}(d) \mid d \in D, f^{-1}(d) \neq \emptyset\}.$$

We refer to ker f as the *kernel partition* of f.

For example, if S is the set of pixels of an image, we could define f(p) as the degree of grayness of the pixel $p \in S$. Another example that is relevant in the study of biodiversity is to consider a set S of observation points in a territory, and define f(p) as the number of species of birds sighted in a certain day in p.

If $C \subseteq S$, then the characteristics of the trace partition $(\ker f)_C$ define the concentration of the values that f takes on the set C. If $D = \{d_1,...,d_k\}$, the blocks of the partition $(\ker f)_C$ have the relative sizes

$$\frac{|f^{-1}(d_1) \cap C|}{|C|}, \ldots, \frac{|f^{-1}(d_k) \cap C|}{|C|}$$

and the distribution of these sizes can be conveniently represented using a histogram.

Definition 1. *Let* $\Pi = (\pi_1,\pi_2,...,\pi_n)$ *be a descending chain of partitions on* S *such that* $\pi_1 = \omega_S$, $f : S \to D$ *be a feature function,* $m : P(S) \to R_{\geq 0}$ *be a measure defined on* S *and let* $\theta,\mu > 0$ *be two positive numbers referred to as the* entropy threshold *and the* measure threshold, *respectively.*

The entropy tree *defined by* Π, f, m, θ *and* μ *is a tree* $\mathcal{T}(\Pi, f, m, \theta, \mu)$ *whose set of nodes consists of blocks of the partitions* π_i *such that the following conditions are satisfied:*

(i) *the root of the tree is the set* S, *the unique block of* ω_S*;*

(ii) *an edge* (B, C) *exists in the tree only if* $B \in \pi_i$, $C \in \pi_i+1$, *and* $C \subseteq B$*;*

(iii) *if* B *is a block of the partition* π_i, *then* $\mathcal{T}(\Pi, f, m, \theta, \mu)$ *contains the set of edges* $\{(B, C) \mid B \in \pi_i$ *and* $C \in \pi_i+1, C \subseteq B\}$ *if and only if* $H((\ker f)_B) \geq \theta$ *and* $m(B) \geq \mu$.

If $\mathcal{T}(\Pi, f, m, \theta, \mu)$ contains the set of edges $\{(B, C) \mid B \in \pi_i$ and $C \in \pi_i+1\}$ we say that the node B is *split* in the tree $\mathcal{T}(\Pi, f, m, \theta, \mu)$. Since splitting involves a sufficiently large value of the entropy and a node of sufficiently large measure, longer paths in the tree point towards subsets of S that contain a large diversity of values of the feature function f.

An entropy quad-tree is an entropy tree $\mathcal{T}(\Pi, f, m, \theta, \mu)$ such that $\Pi=(\pi_1,...,\pi_n)$ is a descending chain of partitions on S, $\pi_1 \geq_4 \pi_2 \geq_4 \cdots \geq_4 \pi_n$. The entire image area S corresponds to the root of the quad-tree.

The expansion of a node B is based on its entropy value and the predetermined threshold used for the splitting condition, as well as the size of the corresponding subarea. Only nodes with area greater or equal to the defined minimum window size are expanded. The complex areas correspond to leaves at the highest level on the quad-tree.

2. An Algorithm for Detection of High Complexity Regions

The algorithm proposed constructs a full quad-tree related to the image entropy or box-counting dimension concentration to find high complexity areas.

The construction of the quad-tree is based on the measurements of the feature in image sub-areas, which can also be regarded as tree nodes. The algorithm receives as input the gray scale version of an image, a minimum area size for analysis and arguments relevant to the node splitting condition. For the entropy based method described in subsection 2.1, we use a predetermined threshold for the entropy in order to decide whether or not to split a node. For the box-counting dimension method, two distinct arguments are used in the splitting condition: a predefined threshold for the fraction of intercepting boxes or rectangles at any image sub-area and a predefined threshold for the number of gray shades to be considered at the intercepting analysis. The entire image area corresponds to the root of the quad-tree. The expansion of each node is based on its feature value and the predetermined threshold(s) used for the splitting condition, as well as the size of the corresponding sub-area. Only nodes with area greater or equal to the defined minimum area size are expanded.

Our algorithm corresponding to Table 1 outputs a quad-tree showing the feature concentration along the whole image area. In this representation, leaves are assigned with a shade of gray, depending on their location on the tree level. Leaves located closer to the root correspond to areas of the image assigned with darker shades of gray whereas leaves located further from the root correspond to areas of the image assigned with lighter shades of gray. The algorithm also highlights the leaves at the highest tree level with highest feature value. In most cases, those leaves correspond to high complexity regions of the image.

The function *ComputeFeature* evaluates the feature associated with the histogram of the pixels in the node's area. We present two versions for this function in subsection 2.1 and subsection 2.2 as it differs according to the measure used. The recursive method *Split* introduced in Table 2 expands a node if its feature satisfies the method related splitting condition and if its area is greater or equal to the defined minimum area size. A gray shade corresponding to a level in the final tree is assigned to every leaf node by the method *Draw*. The higher the level value, the lighter is the shade of gray assigned to the leaf. Information about each leaf such as its id, feature value and level is saved in a text file by the method *SaveNodeInfo*. The method *Release* frees the memory space previously allocated to a node. Finally, the method *HighlightHighFeatureLeaves* highlights in pink or white the leaves at the highest tree level with highest feature values, corresponding in most cases to high complexity regions. The white color leaves are the ones with the highest feature value among all pink leaves.

2.1 Information-theoretical Method

Our method evaluates the entropy of the local histograms of image sub-areas to find high complexity regions. The partition blocks of a node, used for the entropy analysis, consist of pixels with the same shade of gray.

Table 3 presents an algorithm for the information-theoretic method proposed. It computes the entropy associated with the histogram of the pixels in a node's area. This histogram is created by the method *InsertGrayShade*. The result generated by *ComputeFeature* is successively used by the recursive method *Split* shown in Table 2. Only the nodes corresponding to sub-areas of the image where the entropy is above the predefined entropy threshold and have area greater or equal to the pre-defined minimum area size are expanded. We observed that leaves at the highest level in the resultant quad-tree may naturally have different associated entropy values.

2.2 Box-counting dimension Method

The box-counting dimension is a measure used to determine the fractal dimension of a set S in a metric space. It reflects the variation of the results of measuring a set at a diminishing scale, which allows the observation of progressively smaller details.

Let (S,O$_d$) be a topological metric space and let T be a precompact set. For every positive r, there exists a r-net for T; that is a finite subset N_r of S such that $T \subseteq \{\cup C(x, r|x \in N_r\}$ for every $r > 0$. Denote by $n_T(r)$ the smallest size of an r-net of T. It is clear that $r < r'$ implies $n_T(r) \geq n_T(r')$. The box-counting dimension is introduced next (see [7]).

Definition 2. *Let* (S,O$_d$) *be a topological metric space and let T be a precompact set. The* upper box-counting dimension *of T is the number*

$$ubd(T) = \lim_{r \to 0} \sup \frac{n_T(r)}{\log \frac{1}{r}}$$

The lower box-counting dimension *of T is the number*

$$lbd(T) = \lim_{r \to 0} \inf \frac{n_T(r)}{\log \frac{1}{r}}$$

If ubd(T) = lbd(T), we refer to their common values as the box-counting dimension of T, denoted by bd(T).

We use the box-counting dimension of the local histograms of image sub-areas to find high complexity regions. The box-counting dimension of a sub-area is based on to the number of intercepting boxes in the sub-area.

Definition 3. *A box is a sub-area of the image with size equal to the predefined minimum area size. An intercepting box corresponds to a box where the number of different shades of gray is greater or equal to a predefined threshold.*

The version of the function *ComputeFeature* presented in Table 4 corresponds to the box-counting dimension method. It computes the box-counting dimension associated with the histogram of the boxes in a node's area. As in the Information-theoretic version, the method

InsertGrayShade constructs a histogram of each box in the node or image sub-area. If the area corresponding to the node is equal to a box area, the number of intercepting boxes is the same as the number of different shades of gray in its histogram. Otherwise, the number of intercepting boxes is equal to the number of boxes with histogram containing a number of shades of gray greater or equal to a predefined threshold. When the box-counting dimension method is used, the recursive method *Split* shown in Table 2 expands a node according to a threshold related to the fraction of intercepting boxes found. For instance, a fraction threshold=0.1 represents a node having 10% of intercepting boxes among all its boxes. So in this case, the algorithm expands a node if its box-counting dimension corresponds to a fraction greater than 10% of intercepting boxes. The area corresponding to the node should also be greater or equal to the predefined minimum area size in order to promote expansion. As in the Information-theoretic method, we also observed that leaves at the highest level in the resultant quad-tree may naturally have different BCD values associated. We show in subsection 2.4 that the leaves with highest BCD value among the ones at highest level can better represent high complexity areas of the image.

2.3 System Description

The algorithm was implemented in Java (JDK 6 Update 7) and the program is composed by 8 classes: Main, Image, Tree, EntropyTree, BoxCountingTree, Node, EntropyNode and BoxCountingNode. The class Tree is a super class for the classes EntropyTree and BoxCountingTree and the class Node is a super class for the classes EntropyNode and BoxCountingNode. The class Main instantiates an Image object. The class Image implements the methods for encoding and decoding images, as well as for treating the image prior to the generation of the quad-tree. Image treatment may involve resizing and conversion to gray scale. The class Tree is a super class with attributes and methods shared by both classes EntropyTree and BoxCountingTree. Those two classes, together with the classes EntropyNode and BoxCountingNode contain the implementation of the methods presented in 2.1 and in 2.2. The class Node is a super class with attributes and methods shared by both classes EntropyNode and BoxCountingNode.

2.4 Experimental Results

Experiments were performed over decompressed gray scale version of JPEG images. The use of gray scale images allowed the methods to be applied over a reduced color space. The resultant image files were again compressed and presented as JPEG files. The values chosen for all the thresholds promote a good capture of the complexity. The resultant images and statistics show that the quad-trees generated by both methods are quite similar. Fig. 1(a) and Fig. 1(b) present the relation between the values chosen as threshold for both methods and the percentage of the number of pixels located in high complexity areas relative to the total number of pixels in each sample image. One can notice that the percentages of pixels in high complexity areas generated for each image file are very close in value for both methods. The files corresponding to the quad-trees generated for the first four sample images are presented in Fig. 2. As mentioned in section 2, a gray shade corresponding to a level in the final tree, is assigned to every leaf node. The leaves at the highest tree level with highest feature values, corresponding in most cases to high complexity regions, are highlights in pink or white. The white color leaves are the ones with the highest feature value among all pink leaves. Results for both methods also show the relation between the characteristics of the images and the values used for the node splitting condition. Images corresponding to natural scenes or objects and faces with a textured background require a higher value for the entropy threshold, as well as for the threshold used for the box-counting dimension evaluation in order to capture well the complex regions. Those images present a higher number of pixels located in high complexity areas. Images with objects and faces exposed over a more uniform background require lower values for those parameters. Those images present a lower number of pixels located in high complexity areas.

Although only JPEG files were used in the experiments here presented, the algorithm and methods described in this paper are independent of image type. So in order to compare the results between different formats, we also performed experiments with BMP image files. In this case, each JPEG file was created from an original Bmp image. Results for both formats regarding both methods were also quite similar and demonstrate that our algorithm can capture high complexity domains independent of an image format. We also observed that as we lowered the compression quality of JPEG images, there was a decrease on the number of pixels located in high complexity sub-domains. JPEG compression removes high frequency details from images as considered by Pevny and Fridrich [8]. Furthermore, the number of image artifacts increases as we lower the compression quality. Uncompressed formats (BMP, PCX) or lossless compression formats (PGM, TIFF) usually carry a higher degree of noise and less artifacts. As a consequence of the high frequency removal and addition of more artifacts, JPEG files with low quality usually have less high complexity areas when compared to the correspondent JPEG image files compressed with higher quality and Bmp images.

3. An Algorithm for Crater Detection Using Entropy Quad-trees

We applied entropy quad-trees to the first stage of an algorithm for crater detection. The proposed algorithm locates craters on the surface of Mars, represented by circles in digital images. The algorithm proposed constructs a full entropy quad-tree related to the image entropy concentration to find high complexity areas that can also contain edges. Later, a slightly modified CHT is used to detect the presence of circles in the complex areas found during the entropy analysis. The algorithm receives as input the 8 bits gray scale version of an image, a minimum window size for analysis, a threshold relevant to the node splitting condition, the minimum and maximum radius values for the searched craters and a threshold for the CHT. Its output lists the detected craters as well as their estimated center points highlighted and superimposed over the original image. A text file with data indicating the center points, radius and Hough Space bin points of each detected crater is also generated.

The construction of the entropy quad-tree is based on an information-theoretical method similar to the one described in subsection 2.1. The most significant difference between both versions is that the one used for crater detection also classifies complex areas corresponding to leaves at the highest level on the quad-tree according to the possibility of presence of an edge. Our new method is presented in subsection 3.1.

First, the algorithm determines the average gray intensity of the original image, as well as the low intensity average (average of gray shades below average intensity) and high intensity average (average of gray shades above average intensity). Then, the pixels in each area with minimum size for analysis are mapped to two different sets according to the thresholds corresponding to the average of low intensity shades or the average of high intensity shades of the original image. Crater edges can be found in areas that contain only dark shades of gray or areas containing light shades of gray.

The classification considers the number of pixels in a minimum size window that are above the high intensity average threshold if at least one pixel in the area has gray shade above the average intensity. Otherwise, if all the pixels in the area have low intensity, the classification considers the number of pixels in the minimum size window that have shade below the low intensity average threshold.

Let n be the number of pixels satisfying one of those conditions and h the height of our minimum window. Also, suppose we have a square window. When $h-2 < n < h^2-1$, the entropy value remains considerably high and the area is classified as a leaf that possibly contains an edge. Only those leaves are relevant to our algorithm.

After the high complexity regions that may contain edges are found, the algorithm determines another threshold corresponding to a high intensity shade which is higher than the high intensity average shade, lower than the maximum intensity found in the image and has the highest histogram value among the shades satisfying the couple previous conditions. This last threshold which we will call “near maximum intensity” threshold is used to highlight high intensity pixels corresponding to edges. Pixels with light shades of gray (higher than average intensity) that form edges usually have intensity greater than the “near maximum intensity” threshold.

Finally, the entropy analysis generates a binary image where pixels with shades of gray below the low intensity threshold and pixels with shades of gray above the near maximum intensity threshold are mapped to white. All the other pixels are mapped to black. The resultant binary image corresponds to the output of the entropy analysis and input of the Circle Hough Transform method. As previously mentioned, the original image does not need any pre-processing. The entropy analysis works as an information theoretic edge filter that generates a binary image from complex areas which may contain edges.

Our next step is to apply the CHT to detect circles in the binary image. The CHT method maintains an accumulator array to find triplets (a, b, r) that describe circles where (a, b) is the center of a circle with radius r. Each point (a, b) in the image receives a score value referred to as the *number of votes* equal to the number of points (x, y) fall on the perimeter of the circle (a, b, r). This score is stored in an accumulator array. The detected center points have the highest numbers of votes.

Two stopping conditions are commonly used by the CHT algorithm: the maximum number of circles to be found and a threshold for the minimum number of votes related to a point in the parameter space. In our application, there is no systematic way to reasonably predict both values. Furthermore, it was observed that for any set of radius where the difference between the minimum and maximum radius is relatively small, the chances of a point to represent a real circle center decreases as the number of votes related to the point gets further from the peak value found in the accumulator array. Points with a number of votes relatively far from the peak value usually correspond to near true center points, near center points of poorly delimited circles or points that received votes in the parameter space simply due to noisy pixels that are not part of any circle edge. To alleviate this problem, we created a new threshold for the number of votes corresponding to the maximum distance from the peak value in the accumulator array as our stopping condition for the CHT method. We also restricted each search to small sets of contiguous radii. Details are provided in subsection 3.2.

The algorithm for crater detection corresponds to Table 5. The recursive method *Split* shown in Table 2 was introduced in section 2. Our method *ComputeFeature* corresponds to a slightly modified version of the one presented in Table 3 and described in section 2, since it also classifies each leaf according to the presence of an edge. Only leaves which may contain an edge are considered by the method *ProcessImageEntropy*. This method generates a binary image representing the entropy analysis to find complex areas that may contain edges. Pixels with shades of gray below the low intensity threshold and pixels with shades of gray above the near maximum intensity threshold are highlighting in white. All remaining pixels are mapped to black.

The method *ComputeCHT* detects circles in the binary image with radii between the minimum and maximum values given as arguments. It also highlights the detected craters as well as their estimated center points over the original image and generates a text file with data related to the craters found such as radius, center points and number of points in the bins associated with each center point.

3.1 Information-theoretical Method

Table 6 presents an algorithm for the information-theoretic method similar to the one presented in Table 3. It computes the entropy associated with the histogram of the pixels in the area corresponding to a node. This histogram is created by the method *InsertGrayShade*. *ClassifyLeaf* classifies a minimum area node according to the possible presence of an edge. As previously mentioned, only leaves that may contain edges are relevant to our algorithm. The result generated by *ComputeFeature* is successively used by the recursive method *Split* shown in Table 2.

3.2 Circular Hough Transform Method

The Hough Transform is a standard method for shape recognition in digital images. It was first applied to the recognition of straight lines [9, 10] and later extended to circles [11, 12], ellipses [13], and arbitrary shaped objects [14]. The Circular Hough Transform (CHT) can be used to determine the parameters of a circle when a number of points that fall on the perimeter are known. A circle with radius r and center (a, b) can be described with the parametric equations:

$$x = a + r \cos \alpha \text{ and } y = b + r \sin \alpha, \text{ where } 0 \leq \alpha \leq 2\pi$$

The locus of (x, y) points in the Hough or parameter space falls on a circle of radius r centered at (a, b). The true center point will be common to all parameter circles, and can be found with an accumulator array that stores the number of votes for each point in the parameter space. Multiple circles with the same radius can be found with the same technique.

The main disadvantage of the transform is the fact that the parameter space corresponds to a 3-dimensional space, which makes the computational complexity and storage requirements $O(n^3)$. If the circles in an image are of known radius r, the search can be reduced to a 2-dimensional space.

The method used in our algorithm searches for all circles with radius between two values given as arguments. It differs from other version of CHT methods because of its stopping condition. For reasons previously mentioned, our method does not use the maximum number of circles or the minimum threshold for the number of votes in order to end the search. Instead, it uses a threshold corresponding to the maximum allowed difference between the peak value in the accumulator array of votes and any other number of votes related to a point in the parameter space.

Let $A_{[W][H][R]}$ denote the accumulator array of votes where W is the image width, H is the image height and R depends on the size of the radius set with minimum element rmin and maximum element rmax, and on the value for the chosen radius increment i. Let t denote the introduced threshold and v be the greatest value stored in the accumulator array A corresponding to a point (w, h, r) where $0 \leq w \leq W-1$, $0 \leq h \leq H-1$ and $0 \leq r <= (rmax - rmin) / i$. Then an arbitrary point (w’,h’,r’) where $0 \leq w' \leq W-1$, $0 \leq h' \leq H-1$ and $0 \leq r' <= (rmax - rmin) / i$ having v’ votes in A is detected as a circle center iff $v - v' \leq t$. We observed that our threshold works well for small groups of contiguous radii. Since the size of the group is small, all the radii are close in value and points corresponding to the center of a circle with one of those radii also have a relatively close number of votes in the parameter space. Therefore, the difference between the number of votes corresponding to centers of true circles that are reasonably well delimited cannot be large when the search is performed for a small group of contiguous radii.

Table 7 presents an algorithm for the CHT proposed. *HoughTransform* computes the Hough Transform of the binary image generated during the entropy analysis and *ComputeHS* generates an image corresponding to the Hough Space. *ComputeCenterPoints* finds circles and their center points by checking the accumulator array containing votes for each pixel in the image. *DrawCircles* highlights the detected craters as well as their estimated center points over the original image. *PrintCirclesData* generates a text file with data related to the craters found such as radius, center points and number of points in the bins associated with each center point.

3.3 Experimental Results

Experiments were performed over the decompressed 768x768, 8 bits gray scale version of a JPEG digital image corresponding to a picture of Mars surface presented in Fig. 3(a). This image was obtained from the original 24 bits/pixel PGM digital image labeled 3_24 used as training site by the authors of [3]. 3_24 corresponds to one section of a footprint image (h0905 0000) from the High Resolution Stereo Camera (HRSC) instrument of the MarsExpress orbiter. This footprint is about 8248 x 65448 pixels in size and was split into 264 (6 x 44) sections of 1700 x 1700 pixels each. Image 3_24 corresponds to one of those sections.

The use of gray scale images allowed the methods to be applied over a reduced color space. We used a 3 × 3 minimum area for the entropy analysis and a high entropy threshold equal to 3 due to the heavy presence of texture in the original image. Images corresponding to natural scenes, objects and faces with a textured background or images with a high level of noise contain a large amount of information. It is natural that those images contain more areas with high entropy than images with less textured background. Fig. 4(a) shows the binary image generated during the entropy analysis.

We chose a threshold equal to 30 for the CHT method and divided the search into runs containing 15 contiguous values of the radius. We focused on searching for craters with radii varying from 5 to 52 pixels. It was also observed that the choice regarding the search for only 15 radii at a time, combined with a threshold equals to 30 provided reasonably good results. Since the values of the radius in each run are close, the difference between the number of votes for the points corresponding to centers of well delimited circles is usually not greater than 30. Our algorithm was able to detect 50 craters with radii varying between 5 and 20 pixels, 2 craters with radii varying from 20 to 36 pixels and one crater with radius equal to 45 pixels.

Fig. 3(b) shows the final image generated by the algorithm. The detected craters and estimated center points are highlighted over the original image. Notice that for some craters, the center points are slightly shifted to the left or right of the true center point because the characteristic shadow inside the crater is also detected as an edge by the entropy analysis. As presented in Figs. 4(b), 4(c) and 4(d), the algorithm cleans the areas of the entropy analysis image corresponding to the craters found (by mapping their pixels to black) after each CHT run. This cleaning process helps to decrease the amount of noise and therefore undesirable circles overlapping for subsequent runs.

The heavy presence of texture in the image can highly impact the quality of the inter-mediate image generated by the entropy analysis, which works also as a edge detector tool based on entropy. As the level of texture or noise increases, so does the entropy of regions in the picture. As consequence, the distinction between high entropy nodes that may contain edges becomes harder. On the other hand, the quality of the results generated by the CHT method highly depends on the quality of the entropy analysis image taken as input. Specially, as the detected edges get more and more similar to the real crater edges, it becomes easier for the CHT method to accurately recognize those circles corresponding to craters. We noticed that the image generated by the entropy analysis does not show all the possible true edges corresponding to crater borders. In order to avoid the capturing of heavy noise, we use a high entropy threshold. By using such high threshold, the algorithm cannot capture true crater borders in areas where the variance among the pixels is not high. As a consequence, those craters cannot be detected by the CHT method. Therefore, improving the detection of edges for heavily noisy or textured images during the entropy analysis can directly impact the quality of the final results. Results also show that the algorithm may detect a larger number of false positives craters as the radius increases. Remains of smaller circles that were not completely cleaned from the binary image due to the imperfection of circle edges, may contribute for undesirable circle overlapping in the Hough Space.

4. Conclusions

In this paper we introduced entropy quad-trees and demonstrate that those structures successfully capture high complexity sub-domains of a data domain. The analysis of 2-dimensional image domains showed that similar results are obtained by quad-trees generated with the BCD measure. Quad-trees created for different image formats – medium/high quality JPEG and BMP – are also similar for both measures. We observed that besides capturing image regions corresponding to content related complex areas, entropy and BCD quad-trees also mine other regions with high variance of shades among pixels caused by external factors such as light reflection originated from a camera flash. Nevertheless, the identification of any kind of high complexity region plays an important role for a variety of applications such as data hiding and bio-diversity systems.

As an application of our proposed technique, an algorithm using entropy quad-trees to detect circles that can possibly correspond to craters in images was introduced. The algorithm performs an information-theoretic analysis of the histogram of sub regions of the image in order to find complex areas that may contain edges. A modified CHT detects circles in those complex areas and provides information about center points and radius of the circles found. A threshold corresponding to the maximum distance from the peak value in the accumulator array is used as stopping condition for the method. There is no external pre-processing of the original PGM image 3_24 other than conversion to JPEG format and resizing. No external edge filter is used to process the original image prior to the CHT method. The entropy analysis works as an edge filter that generates the binary image given as input to the CHT method. The heavy presence of noise and texture may compromise the quality of the complex areas found during the entropy analysis and impact the quality of the final results. Therefore, by improving the robustness of the entropy analysis against heavy noise and texture, more craters will accurately be detected.

We intend to extend the application of information-theoretical techniques to other structures associated with spatial data sets such as grid-files, KD-trees, and R-trees. Another area of great potential is the application of entropy quad-trees to the identification of terrain areas that contain a high level of biodiversity.

REFERENCES

1. K.Solanki, O.Dabeer, U.Madhow, B.S.Manjunath, S.Chandrasekaran: Robust image-adaptive data hiding: Modeling, source coding, and channel coding. Proceedings of the Annual Allerton Conference on Communication Control and Computing 41(2) (2003) 829– 838
2. B.D.Bue, T.F.Stepinski: Machine detection of martian impact craters from digital topography data. IEEE Transactions on Geoscience and Remote Sensing 45(1) (2007) 265–274
3. E.R.Urbach, T.F.Stepinski: Automatic detection of sub-kilometer craters in high resolution planetary images. Planetary and Space Science 57 (2009) 880–887
4. G.Salamuniccar, S.Loncaric: Gt-57633 catalougue of martian impact craters developed for evaluation of crater detection algorithms. Planetary and Space Science 56 (2008) 1992–2008
5. Simovici, D.A., Jaroszewicz, S.: An axiomatization of partition entropy. IEEE Transactions on Information Theory 48 (2002) 2138–2142
6. Simovici, D., Jaroszewicz, S.: Generalized conditional entropy and decision trees. In: Extraction et Gestion des connaissances – EGC 2003, Lavoisier, Paris (2003) 363–380
7. Simovici, D.A., Djeraba, C.: Mathematical Tools for Data Mining – Set Theory, Partial Orders, Combinatorics. Springer-Verlag, London (2008)
8. Pevny, T., Fridrich, J.: Benchmarking for steganography. In: Information Hiding: 10th International Workshop, IH 2008, Sana Barbara, CA, USA. Volume 5284., (Springer)
9. V.F.Leavers: Survey: which hough transform? Computer Vision Graphics and Image Pro-cessing:Image Understanding 58(2) (1993) 250–264
10. J.Illingworth, Kittler, J.: Survey: a survey of the hough transform? Computer Vision Graphics and Image Processing 44(1) (1988) 87–116
11. R.O.Duda, P.E.Hart: Use of the hough transformation to detect lines and curves in pictures.

Communications of the Association of Computin Machinery 15(1) (1972) 11–15

12. E.R.Davis: A modified hough scheme for general circle location. Pattern Recognition Letters 7 (1987) 37–43
13. R.K.K.Yip, P.K.S.Tam, D.N.K.Leung: Modification of hough transform for circles and ellipses detection using a 2-dimensional array. Pattern Recognition 25 (1992) 1007–1022
14. D.C.W.Pao, H.F.Li, R.Jayakumar: Shape recognition using the straight line hough transform: theory and generalization. IEEE Transactions on Pattern Analysis and Machine Intelligence 14(11) (1992) 1076–1089

Rosanne Vetro received the B.S. degree in Computer Science from the Federal University of Rio de Janeiro, Brazil. After graduation, she worked for 3 years as a team leader in the Research and Development Engineering Department of TV Globo, a Brazilian broadcast company. In recognition of her work on television broadcast systems, she received an Outstanding Engineer Award. She also held a visiting researcher position at the Tokyo Science & Technology Research Labs of NHK, a Japanese broadcast company, where she contributed to the development of the ISO/IEC MPEG-7 International Standard. Since 2008, she has been pursuing a Ph.D. degree in Computer Science of the University of Massachusetts Boston. Her research interests are in the area of data mining with application to multimedia content, clustering, security and data hiding. In 2010, she received a best paper award for her work on mining high complexity regions.

Dr. Dan Simovici is a Professor of Computer Science and Graduate Program Director at the University of Massachusetts Boston. He obtained his Ph.D. from the University of Bucharest and is the author of several books and of more than 150 research papers. His most recent book "Mathematical Tools for Data Mining" was published by Springer-Verlag in 2008. His main research interests are in Data Mining (clustering, genetic algorithms in data mining, graph mining, classification) and in algebraic and information-theoretical methods in Multiple-Valued logic. He is a managing editor of the "Journal for Multiple-Valued Logic and Soft Computing".

Wei Ding has been an Assistant Professor of Computer Science at the University of Massachusetts Boston since 2008. She received her Ph.D. degree in Computer Science from the University of Houston in 2008. Her main research interests include Data Mining, Machine Learning, Artificial Intelligence, Computational Semantics, and with applications to astronomy, geosciences, and environmental sciences. She has published more than 30 referred research papers and has 1 patent. She is the recipient of a Best Paper Award at IEEE ICCI 2010, a Best Poster Presentation award at ACM SIGSPAITAL GIS 2008, and the Best PhD Work Award between 2007 and 2010 from the University of Houston. Her research projects are currently sponsored by NASA, DOE, and NSF.

Figure 1(a). Fraction of pixels in high complexity areas of the image given the corresponding lower bound used for the entropy.

Figure 1(b). Fraction of pixels in high complexity areas of the image given the corresponding lower bound used for the BCD threshold.

Figure 2(a-d). Sample simulation results for original images comparing the corresponding Entropy Tree(Top right in each subfigure) and the corresponding BCD Tree(bottom right in each subfigure).

Figure 3(a). Original image from Mars surface(768x768, 24 bits per pixel).

Figure 3(b). Final image generated by the algorithm where detected craters and their estimated centers are highlighted in pink or white.

Figure 4(a). Intermediate result of crater detection. Binary image generated by the entropy analysis.

Figure 4(b). Intermediate result of crater detection. Binary image generated after detection of craters with radius between 5 and 20. The pixels corresponding to the circles detected by the CHT are mapped to black.

Figure 4(c). Intermediate result of crater detection. Binary image generated after detection of craters with radius between 21 and 36. The pixels corresponding to the circles detected by the CHT are mapped to black.

Figure 4(d). Intermediate result of crater detection. Binary image generated after detection of craters with radius between 37 and 52. The pixels corresponding to the circles detected by the CHT are mapped to black.

Figure 1(a)

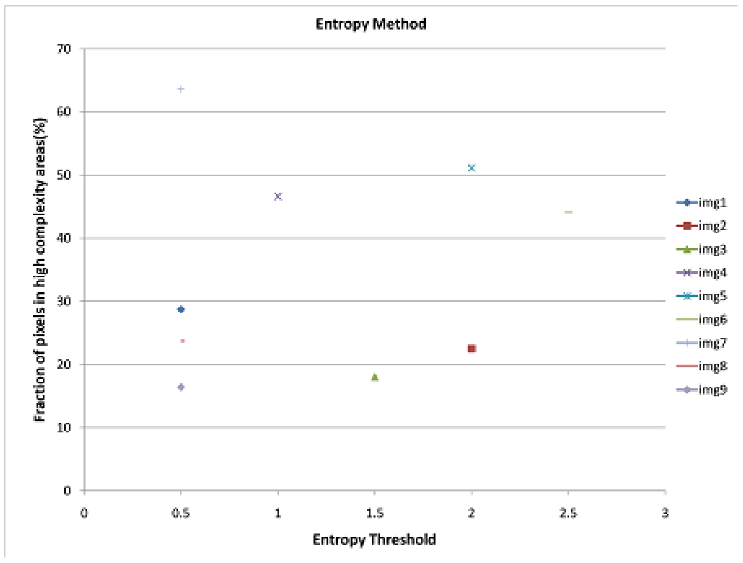

Figure 1(b)

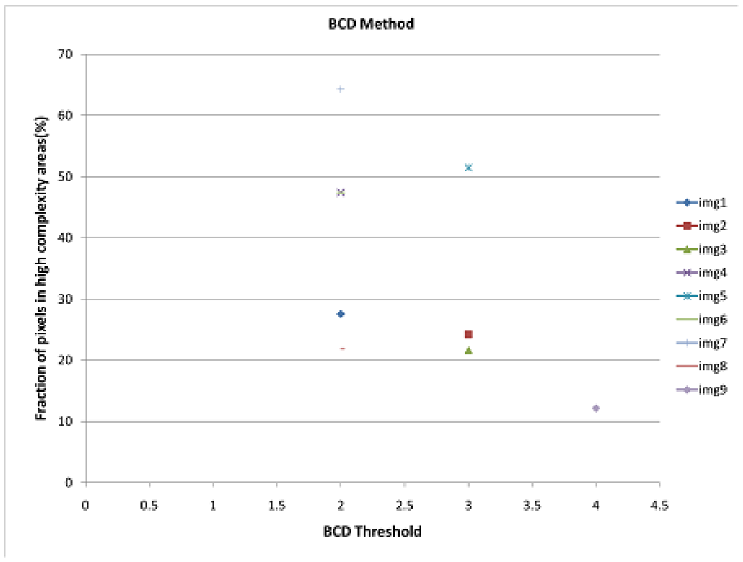

Figure 2(a)

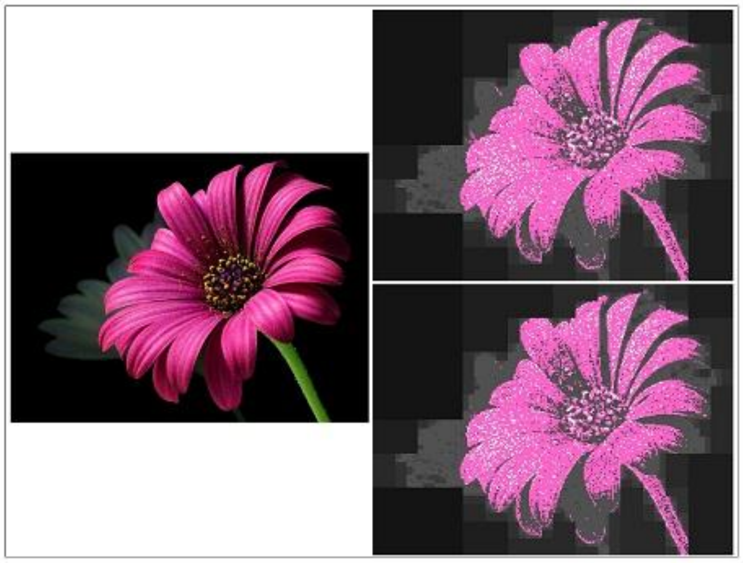

Figure 2(b)

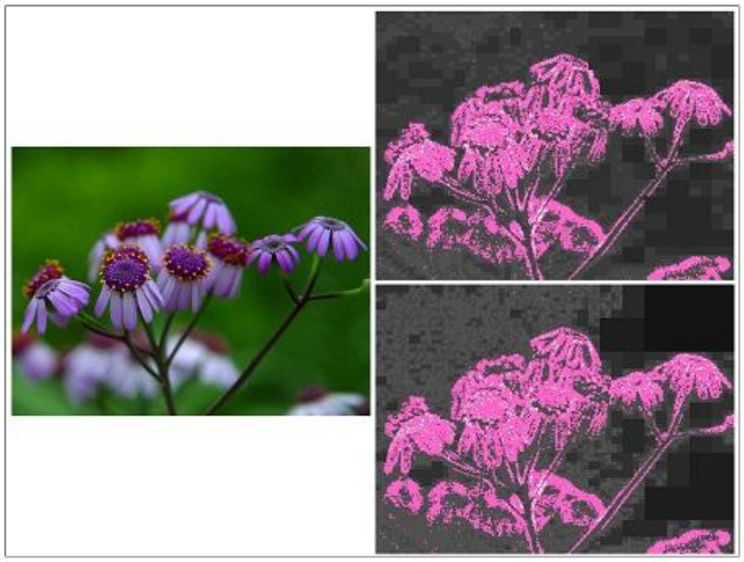

Figure 2(c)

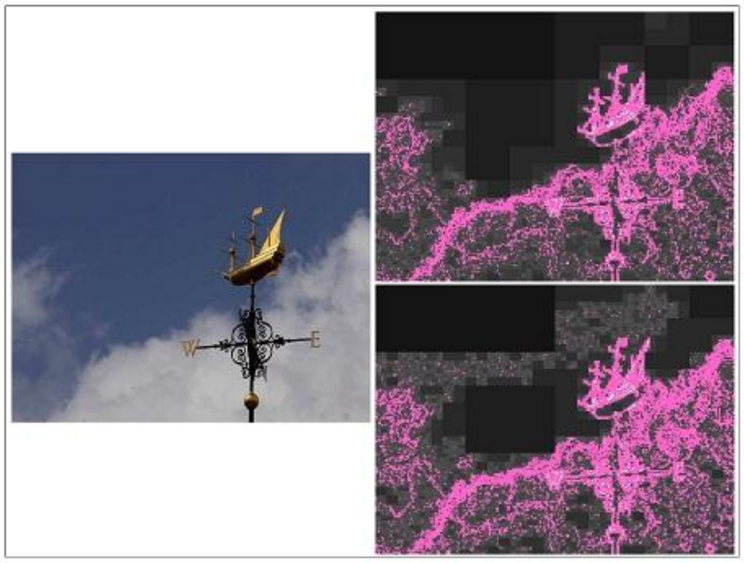

Figure 2(d)

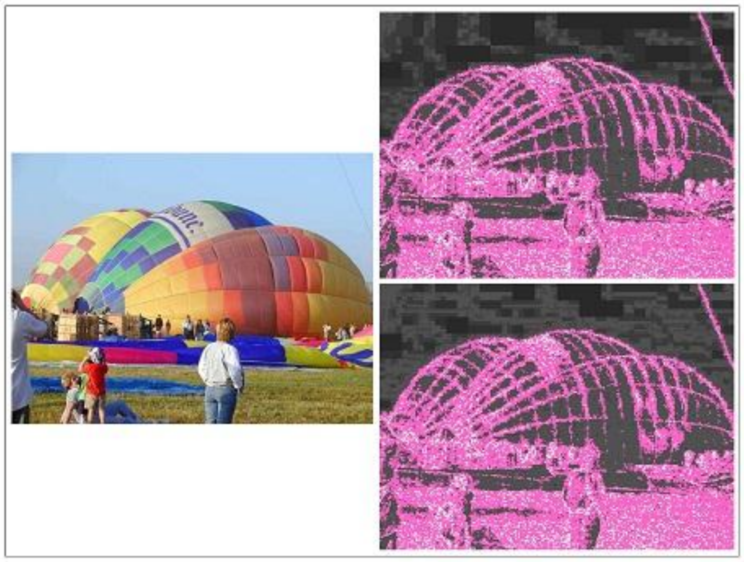

Figure 3(a)

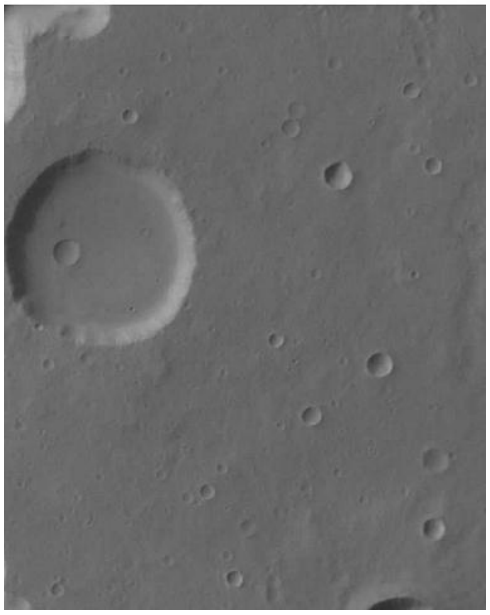

Figure 3(b)

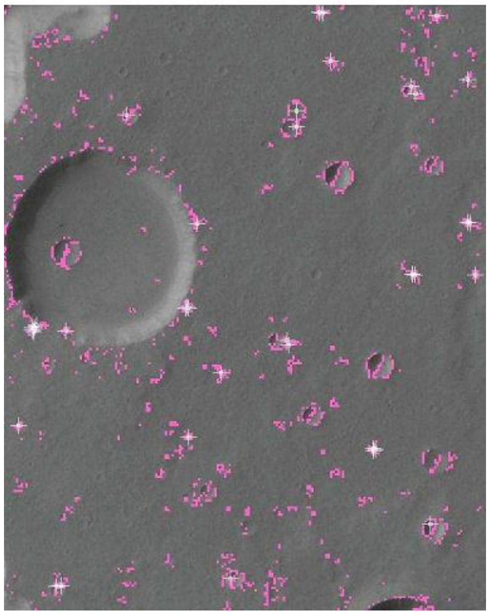

Figure 4(a)

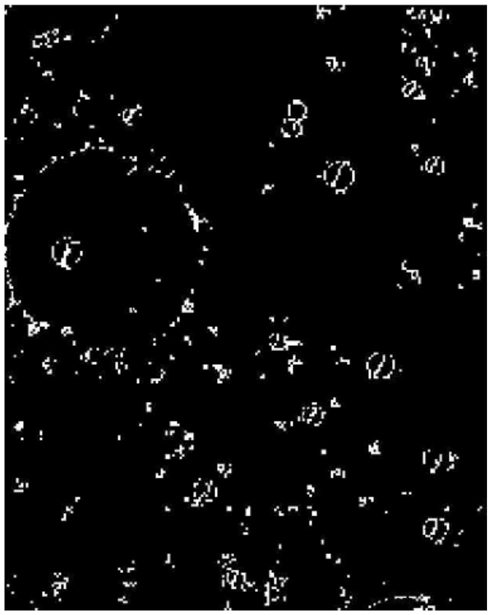

Figure 4(b)

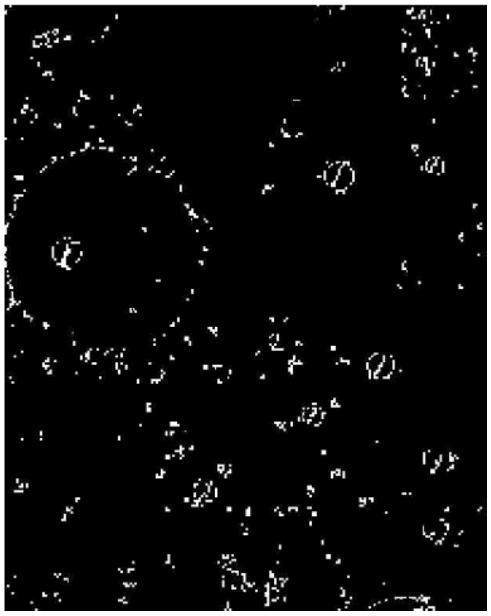

Figure 4(c)

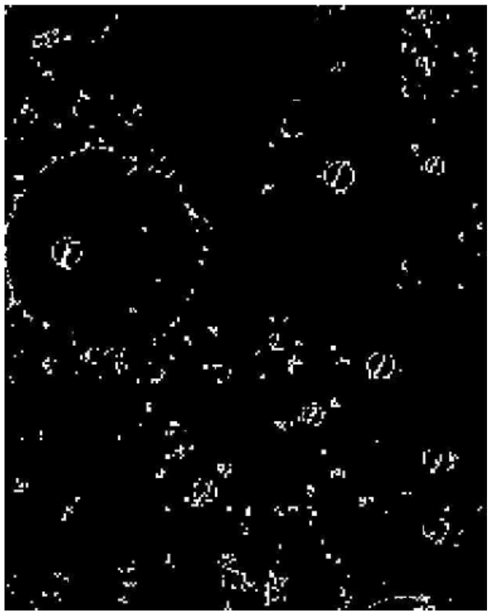

Figure 4(d)

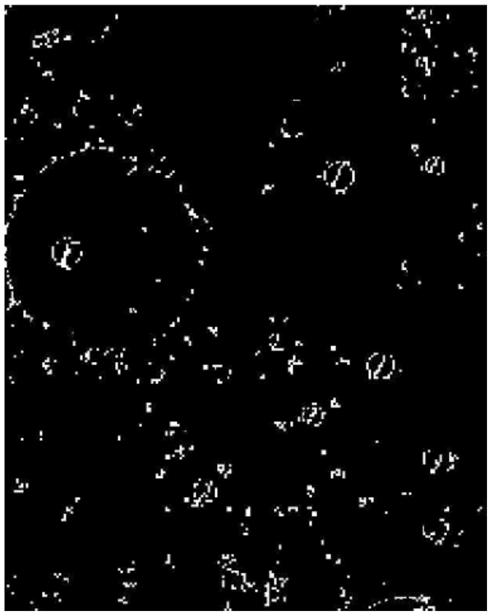

Table 1. Algorithm 1: ComputeHCRegions(*image, minArea, thr1, thr2*)

Table 2. Algorithm 2: Split(*n*)

Table 3. Algorithm 3: ComputeFeature(*n*) for Information-Theoretical Method

Table 4. Algorithm 4: ComputeFeature(*n*) for Box-Counting Dimension Method

Table 5. Algorithm 5: ComputeCraters(*image, minArea, thrEntropy, minRadius, maxRadius, thrCHT*)

Table 6. Algorithm 6: ComputeFeature(*n*) for crater detection application

Table 7. Algorithm 7: ComputeCHT(entropyImg, minRadius, maxRadius, thrCHT)

Algorithm 1 ComputeHCRegions(*image*, *minArea*, *thr1*, *thr2*)

Input: Gray scale image, minimum area size for the analysis, feature threshold, threshold corresponding to the number of shades of gray (used only by the BCD method)

Output: Quad-tree showing the feature concentration along the whole image area

```
  nId ←ROOT
  nLevel ← 0
  root ← newNode(nId, nLevel, image.width, image.height)
  ComputeFeature(root)
  Split(root)
  HighlightHighFeatureLeaves()
```

Algorithm 2 Split(*n*)

Input: A node *n* from a quad-tree

Output: Expands the node creating four children, if node satisfies the necessary requirements

```
if (n.feature > method lower bound) and (n.area > minArea) then
    nLevel ← n.level +1
    nId ← n.id + A
    topLeft ← newNode(nId, nLevel, n.rect.x, n.rect.y, n.rect.width/2, n.rect.height/2)
    ComputeFeature(topLeft)
    nId ← n.id + B
    topRight ← newNode(nId, nLevel, n.rect.x + n.rect.width/2, n.rect.y, n.rect.width/2,
    n.rect.height/2)
    ComputeFeature(topRight)
    nId ← n.id + C
    bottonLeft ←newNode(nId, nLevel, n.rect.x, n.rect.y + n.rect.height/2, n.rect.width/2,
    n.rect.height/2)
    ComputeFeature(bottonLeft)
    nId ← n.id + D
    bottonRight ← newNode(nId, nLevel, n.rect.x + n.rect.width/2, n.rect.y+ n.rect.height/2,
    n.rect.width/2, n.rect.height/2)
    ComputeFeature(bottonRight)
    Release(n)
    Split(topLeft)
    Split(topRight)
```

```
     Split(bottonLeft)
     Split(bottonRight)
**else**
      SaveNodeInfo(n)
      Draw(n)
      Release(n)
**end if**
```

Algorithm 3 ComputeFeature(*n*)

Input: A node *n* from a quad-tree

Output: The node entropy related to the histogram of the pixels in the area.

```
entropy ← 0
for all pixel in n.area do
    InsertGrayShade(histogram,pixel.shade)
end for
for all shade in histogram do
    p ← number of pixels with shade
    s ← total number of pixels in the node
    g ← (p / s)
    entropy− =(g) × ($lg_2$(g))
end for
return entropy
```

Algorithm 4 ComputeFeature(*n*)

Input: A node *n* from a quad-tree

Output: Box-counting dimension associated to the node's area.

```
  boxesIntercepting ← 0
  bcd ← 0
  for all box in n.area do
      for all pixel in box do
          InsertGrayShade(box.histogram,pixel.shade)
      end for
      if n.area = box.area then
          boxesIntercepting ← box.histogram.size
      else if box.histogram.size ≥ threshold then
          boxesIntercepting ← boxesIntercepting +1
      end if
      Release(box.histogram)
  end for
  if boxesIntercepting > 0 then
      bcd− = boxesIntercepting / $\lg_{10}$(1 / (n.area))
  end if
  return bcd
```

Algorithm 5 ComputeCraters(*image, minArea, thrEntropy, minRadius, maxRadius, thrCHT*)

Input: 8 bits gray scale version of an image, a minimum area size, entropy threshold, the minimum and maximum radius, CHT threshold

Output: Detected craters as well as their estimated center points highlighted and superimposed over the original image; a text file with data indicating the center points, radius and Hough Space bin points of each detected crater

```
    nId ← ROOT
    nLevel ← 0
    root ← newNode(nId, nLevel, image.width, image.height)
    ConputeFeature(root)
    Split(root)
    entropyImg ← ProcessImageEntropy()
    ComputeCHT (entropyImg, minRadius, maxRadius, thrCHT )
```

Algorithm 6 ComputeFeature(*n*)

Input: A node *n* from a quad-tree

Output: The node entropy related to the histogram of the pixels in the area.

```
entropy ← 0
for all pixel in n.area do
    InsertGrayShade(histogram,pixel.shade)
end for
for all shade in histogram do
    p ← number of pixels with shade
    s ← total number of pixels in the node
    g ← (p / s)
    entropy− =(g) × (lg$_2$(g))
end for
if (n.area == minArea) then
    relevantLeaf ← ClassifyLeaf(histogram)
    if (!relevantLeaf) then
        return 0
    end if
end if
return entropy
```

Algorithm 7 ComputeCHT(entropyImg, minRadius, maxRadius, thrCHT)

Input: image generated by the entropy analysis, minimum and maximum radius, new threshold for stop condition

Output: Detected circles, their estimated center points and radius

```
    HoughTransform(entropyImg,minRadius,maxRadius)
    ComputeHS()
    ComputeCenterPoints(thrCHT)
    DrawCircles()
    PrintCirclesData()
```