dx.doi.org/10.7437/NT2236-7640/2016.01.004
Notas Técnicas,v. 6, n. 1, p. 28–51, 2016

Texture Classification based on Spectral Analysis and Haralick Features

Classificação de Texturas mediante análise espectral e Parâmetros de Haralick

Manuel Blanco Valentín*
Coordenação de Atividades Técnicas (CAT/CBPF),
Centro Brasileiro de Pesquisas Físicas
Rua Dr. Xavier Sigaud, 150, Ed. César Lattes,
Urca, Rio de Janeiro, RJ. CEP 22290-180, Brasil

Clécio Roque de Bom†
Centro Federal de Educação Tecnológica Celso Suckow da Fonseca,
Rodovia Mário Covas, lote J2, quadra J
Distrito Industrial de Itaguaí,
Itaguaí - RJ. CEP: 23810-000, Brasil

Márcio P. de Albuquerque,‡ Marcelo P. de Albuquerque,§ e Elisângela L. Faria¶
Coordenação de Atividades Técnicas (CAT/CBPF),
Centro Brasileiro de Pesquisas Físicas
Rua Dr. Xavier Sigaud, 150, Ed. César Lattes,
Urca, Rio de Janeiro, RJ. CEP: 22290-180, Brasil

Maury D. Correia**
Centro de Pesquisas e Desenvolvimento Leopoldo Américo Miguez de Mello – CENPES
PETROBRAS, Av. Horácio Macedo,
950, Cidade Universitária,
Rio de Janeiro, RJ. CEP 21941-915, Brasil
Submetido: 01/01/2016 Aceito: 16/05/2106

Abstract: In this work we discuss a method to classify a set of texturized images based on the extraction of their Haralick Features. This kind of Classification is capable of providing texture-based measurements (such as contrast or correlation) and use them as main parameters to classify the same type of patterns in other images. In order to improve the classification success ratio a spectral analysis of the textures and, therefore, the use of filters, before the classification step, is proposed here. In this work the classification success has been evaluated using Mean and Canny filters. On the other hand, the Principal Component Analysis is used to optimize the features extracted for the patterns on each image, before introduced into the classifier. With this method the classifying success ratio for the KTH-TIPS subset and 10,000 different permutations was increased –in average– from 72.28% to 84.25%.

Keywords: Haralick Features, Texture Classification, Image Processing, Spectrum Analysis, Principal Component Analysis.

Resumo: Neste trabalho é discutido um método de classificação de Texturas baseado na extração de parâmetros de Haralick. Este tipo de classificação é capaz de fornecer medidas de texturas (como por exemplo contraste ou correlação) em padrões presentes em imagens e usá-las para classificar o mesmo tipo de padrões em outras imagens. Com o objetivo de melhorar a taxa de sucesso na classificação foi proposto realizar uma análise espectral e, por tanto, o uso de filtros em passo prévio ao processo de classificação. Este trabalho apresenta uma avaliação do desempenho da classificação por meio do uso dos filtros Média e Canny. Por outro lado, a Análise de Componentes Principais foi usada para otimizar os parâmetros extraídos dos padrões nas imagens, antes destas serem introduzidas no classificador. Com este método o sucesso na taxa de classificação para a biblioteca de imagens KTH-TIPS e um conjunto de 10.000 permutações diferentes foi incrementado –em média– de 72.28% para 84.25%.

Palavras chave: Parâmetros de Haralick, Classificação de Texturas, Processamento de Imagens, Análise espectral, PCA.

*Electronic address: mbvalentin@cbpf.br
†Electronic address: debom@cbpf.br
‡Electronic address: mpa@cbpf.br

1. INTRODUCTION

Images carry a lot of information about the system represented on them. A part of all this information is directly visible to a person –for instance, colors or grayscale tones–, and it is this part that allow us to understand what we are seeing on that image, and ergo the system that is represented on them. This part of information directly visible to us is, sometimes, only a little slice of the total amount of information contained in the image. For example, it is easy for a person to see shapes, objects, regions, spatial arrangements and differentiate colors or grayscale values on the captured scene. However, another part of the total information may not be visible directly, and so the use of texture algorithms help to extract, from any image, information that may be hidden to the naked eye.

The extraction of texture information has shown positive results when used for characterizing and estimate parameters of any image [1], [2], [3], [4], [5], [6] and [7]. Each parameter will depend strongly on the type of image to be processed (i.e. color-images and grayscale-images will demand a different kind of feature extractors), and so the feature-extraction method will have to fit every case requirements.

In some applications [8], [9], [10] and [11] colored images have too much information that may not be very relevant for the desired application, so grayscale images can be used instead, reducing the amount of information to be processed and, by doing this, simplifying the following processing steps. For both colored and grayscale images, Haralick features –as will be shown later in this work- represent a good choice for texture extraction.

Even though some of these features share some similarities, each one of them will provide information about a very specific property of the image. When it comes to classification, it is expected that each analyzed class –group of images– will have similar textural patterns, while these patterns should be different from a different class. When this is true classes are distinguishable for the used textural extraction method and so the classification process of new images that belong to an unknown class relies on the comparison of these features and the average features for each one of the classes.

This process of classification has, obviously, a strong correlation with the features used. One might think, wrongly, that the more features used, the more information can be extracted from the image and, therefore, the better the classification process may be. Some of these features may be very similar in some cases for images that represent different objects. This similarity of features between two different groups may lead to misclassifications when using automated algorithms and, therefore, to increase the error. Finding the group of features that, for a given group of images and classes, allow to obtain the best performance in classification (the lowest number of misclassifications) can be a deeply time-consuming and inefficient task, considering that the number of possible combinations of features increases, usually, by a power-law.

In this work we aim to describe a complete workflow for texture features and classification for a set of previously known textures. We review some usual techniques in image and texture processing and introduce nontraditional features for the classification problem. We discuss the choice and the motivation for the features, evaluating them using the Principal Component Analysis method.

2. HARALICK TEXTURE FEATURES

Since these features were proposed on 1973 by R. M. Haralick, K. Shanmugam and I. Dinstein [12], several articles have been published using them for image-processing techniques. From breast cancer detection [13], subcellular structures detection [14], to Grain food detection [15], Haralick Features –also called Haralick Parameters- have been used widely for pattern recognition, properties detection, images analyzing and image classification.

On his original paper, Haralick described a set of 14 features that may extract different properties from an image, which could be then used as input parameters for a classifier. Some of these features are related to intuitive properties of the image, such as contrast or homogeneity, which can be intuitively interpreted. However, others have no obvious interpretation.

In any case, Haralick Features calculations are based on a probability 2D matrix called co-occurrence matrix[1], instead of the image itself. This matrix is defined over an image as the distribution of co-occurring values at a given offset. For a given I image with n,m dimensions and Δx and Δy (horizontal and vertical offsets, respectively) values, this matrix can be obtained by (1), being, by definition, a square matrix.

$$C_{\Delta x,\Delta y}(i,j) = \sum_{p=1}^{n}\sum_{q=1}^{m}\begin{cases}1 & I(p,q)=i \ \ AND \ \ I(p+\Delta x,q+\Delta y)=j \\ 0 & otherwise\end{cases} \quad (1)$$

In a more simple way, this matrix is a measure of how many times a certain pair of pixels appears on the image at a given distance (offsets) one from each other. It represents a distribution of co-occurrence events of repetition, meaning that every cell of this matrix represents how many times in the original image a pixel with value i (vertical index of co-occurrence matrix of the cell) was next to a pixel with value j (horizontal index of co-occurrence matrix of the cell), being both of them separated a distance of Δx pixels in the vertical direction of the image and Δy pixels in the horizontal direction.

In order to understand this matrix, imagine a 6x6 example binary image I_{6x6} shown at (2). In this case, non-zero val-

§Electronic address: marcelo@cbpf.br
¶Electronic address: elisangela@cbpf.br
**Electronic address: maury.duarte@petrobras.com.br

[1] The co-occurrence matrix for grayscale images is usually called Gray Level Co-occurrence Matrix, or GLCM

ues mean high intensity values, while zero values could be understood as low intensity pixels.

$$I_{6x6} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{2}$$

Now, if we wanted to know the co-occurrence of values for horizontal 0° pairs (in other words, for a pixel to be followed in the horizontal direction by another pixel), the offset values should be given by (3).

$$\left\{ \begin{array}{l} \Delta x = 0 \\ \Delta y = 1 \end{array} \right\} \tag{3}$$

Obtaining the co-occurrence matrix for this example image would be as simple as applying (2) and (3) on (1). In this case the result would led to the distribution of 0° horizontal combinations of pairs in the image (2). Being this image binary, there are only two possible values that each pixel can assume: 0 or 1. Therefore, there only exist four possibilities of pair-combination in this case, which are: a 0 value followed by another 0 value, a 0 value followed by a 1 value, a 1 value followed by a 0 value and a 1 value followed by another 1 value. The number of times that each one of these events happen over the image will be represented by each cell in the co-occurrence matrix. The co-occurrence matrix for I_{6x6} image would be given by (4).

$$C_{0,1} = \begin{bmatrix} 9 & 4 \\ 4 & 13 \end{bmatrix} \tag{4}$$

Analyzing the previous matrix it is possible to see that, for instance, there are 9 occurrences of a 0 value pixel be followed by another 0 value pixel in the original image, while there are only 4 occurrences of a 1 value pixel be followed by a 0 value pixel or vice-versa. From this matrix it can deduced that, in the example image used in this case, it is more likely to find a 1 value pixel followed by another 1 value pixel than any other combination of pixels.

On the other hand, binary images usually contain less information than grayscale ones –and even less than color images–. The Gray Level of a grayscale picture is a source of a great amount of information that could be interesting to extract from the image. Now let us imagine that it is necessary to process an 8-bit image, which gray-tone ranges between a minimum value of 0 for low-intensity pixels, and a maximum value of 255 for high-intensity pixels.

For most image applications from which all these features are going to be calculated, it can be imagined that it is much more likely to find a pixel with a certain grayscale value followed by another pixel with, approximately, the same value. So if, for instance, a pixel has a value of 234, it is much more probable to find a pixel with a value between 230 and 240 on its side, than one with a 0 value.

Analyzed images are often representations of physical phenomena, and so the calculation of the GLCM for all 255 possible values could be considered unnecessary. In fact, for any grayscale image, with any dimensions, the total number of possible combinations of pixel pairs is 65536. Following the reasoning exposed before, in most applications it is no necessary, for instance, to distinguish between the number of pixels with a 234 value followed by a 235 value, and the number of pixels with 233 value followed by a 234 value. In many cases these small variations may represent noise, so different pixel levels could be added into one. By doing this, the total amount of possible combinations is reduced, optimizing the calculation of Haralick Parameters and, so, minimizing the time needed to process the images. The number of grayscale groups, which are called gray levels, is a very important parameter that may have a major impact on the final result of subsequent processes and therefore should be selected carefully.

To fully understand the process of resizing the original image with the user-defined gray levels consider a grayscale image I_{GS} given by (5).

$$I_{GS} = \begin{bmatrix} 254 & 232 & 251 & 210 & 205 & 248 \\ 10 & 50 & 90 & 66 & 16 & 2 \\ 208 & 131 & 120 & 4 & 251 & 66 \\ 141 & 232 & 9 & 205 & 210 & 227 \\ 60 & 135 & 90 & 189 & 23 & 27 \\ 5 & 48 & 84 & 210 & 98 & 0 \end{bmatrix} \tag{5}$$

Now, assuming that 3-gray levels[2] –which here have been called “A”, “B” and “C”– are defined a new image I_K can be described by (6). Thus, the original image can be rewritten, considering (6), as shown in (7).

$$I_K(i,j) = \begin{cases} A & 0 \leq I_{GS}(i,j) \leq 85 \\ B & 86 \leq I_{GS}(i,j) \leq 170 \\ C & 171 \leq I_{GS}(i,j) \leq 255 \end{cases} \tag{6}$$

$$I_{GS} = \begin{bmatrix} C & C & C & C & C & C \\ A & A & B & A & A & A \\ C & B & B & A & C & A \\ B & C & A & C & C & C \\ A & B & B & C & A & A \\ A & A & A & C & B & A \end{bmatrix} \tag{7}$$

This new labeled image (7) allows to calculate the GLCM in a faster and more efficient way, considering that for an image with 3 gray levels (7) the total amount of possibilities of pairs in a grayscale image is only 9, much less than the original 65536 total possibilities for the original 255 gray level image (5). The GLCM for the 3 gray level image, for the same 0° horizontal offset considered in the binary image example, is shown in (8).

[2] There are several ways to obtain these levels, however the one used in this paper and the example shown above is to simply divide the total gray range in as many groups as new gray levels are desired.

$$C'_{0,1} = \begin{bmatrix} 6 & 2 & 3 \\ 3 & 2 & 2 \\ 3 & 2 & 7 \end{bmatrix} \quad (8)$$

It is possible to notice that, in this case (8), if a random horizontal pair is picked, it is highly likely that the first pixel will have a value between 171 and 255 ("C") and will be followed by a pixel with the same value, or that it will have a value between 0 and 85 ("A") and will be followed by a pixel with the same value.

It is very important to understand how the Offset parameters introduced earlier (Δx and Δy) influence on the calculation of the GLCM. As mentioned earlier in this section, these parameters describe the distance between the two pixels compared to form the GLCM. The relation between these two parameters describes the direction angle θ of an imaginary vector that joins that pair of pixels. Considering that an image is usually represented as a matrix which rows are indicated by index i –which grows downwards– and which columns are indicated by index j –which grows rightwards–, the direction angle for a given pair of Offset parameters can be obtained by (9). This angle between a specific pair of pixels in the image is illustrated in Fig. 1.

$$\theta = \tan\left(\frac{\Delta x}{\Delta y}\right) \quad (9)$$

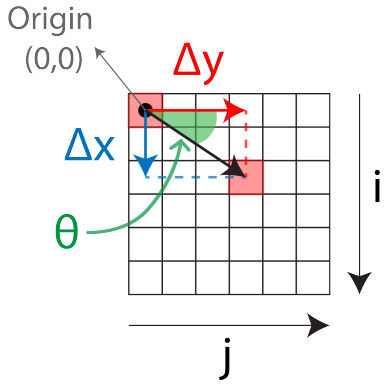

Figure 1: Representation of a discrete image and direction angle between a given pair of pixels.

It is possible to calculate an image GLCM for any pair of offset values desired. Usually, it is more interesting to analyze pairs of pixels at short distance in an image, rather than pairs of pixels which are very far one from each other. In general terms, the smaller the offset values are, the more precise the GLCM will be.

As the images used in this work have reduced size (200x200 pixels), offset parameters will have a maximum value of 1. Taking this into account, only 8 direction angles are possible. These angles are shown in Fig. 2.

As it can be seen in Fig.2, only 4 of these directions are worth to consider, while the other 4 are redundant. For instance, the co-occurrences that a 0° offset will obtain will be the same that a 180° offset might obtain. This same principle can be applied to the other 6 directions, so opposite angles (rotated 180° one from each other) generate the same co-occurence matrices. Due to notation and simplicity the only direction angles considered in this paper are 0°, 45°, 90° and 135°. The relation of these 4 angles and the Offset parameters, for a sample Image, considering 8 graylevels, is shown in Fig. 3.

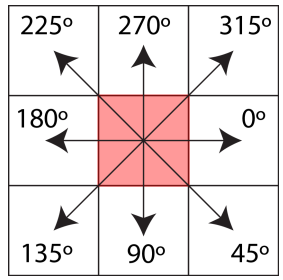

Figure 2: Illustration of possible direction angles for the GLCM when offset parameters are set between 0 and 1.

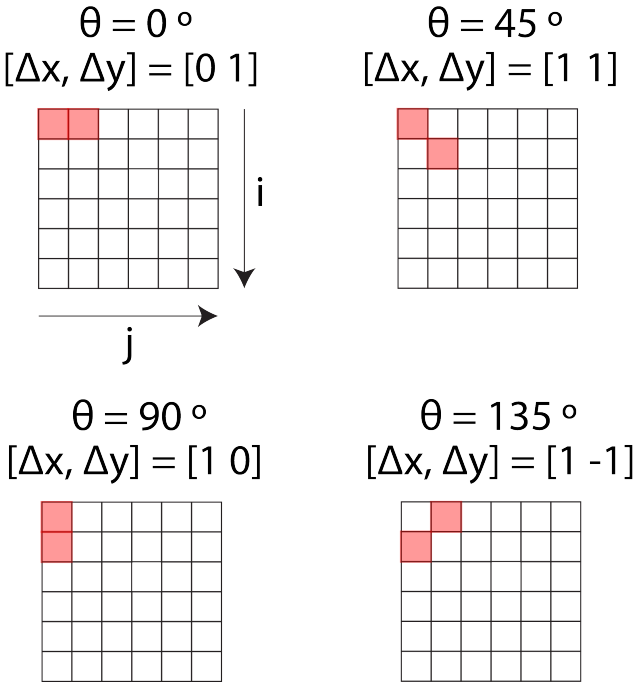

Figure 3: Illustration of 4 Direction used for the calculation of co-occurrence matrices for a given sample image.

From equation (1) it can be easily deduced that each one of these 4 angle directions will produce a different co-occurrence matrix. Rather than studying which direction is better to consider for every given image, Haralick et al. themselves exposed on their paper that 4 co-occurrence matrices should be calculated, each one of them for each possible direction considered before. Then all features could be extracted for each one of these co-occurrence matrices, generating a total of 4 different sets of features. At the end, the value of a single feature could be calculated as the average of all 4 co-occurrence matrices values for that feature.

In order to illustrate this process, the very well-known and used image of Lenna Söderberg has been processed in order

to obtain all 4 GLCM matrices for the 4 directions considered, using 64 gray levels, shown in Fig. 3. It is possible to observe on Fig. 4 how similar all four GLCMs actually are. It is also interesting to know that the more the GLCM has a standing-out diagonal –like in Fig. 4 case– the more correlated the image is. This means, basically, that the original image was not a random set of values, but an image with slowly changes in its grayscale values.

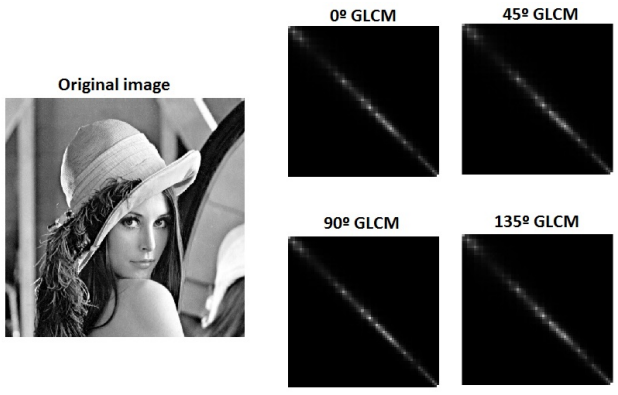

Figure 4: Calculation of GLCM matrices for all 4 main directions (0°, 45°, 90° and 135°) for a sample grayscale picture of Lenna Söderberg.

GLCM matrices can lead to a probability distribution function by the process of normalizing. A GLCM can be understood as the pairs co-occurrence histogram. Thus, in order to obtain the probability distribution function of pairs it is necessary to divide each cell value by the total sum of all values, according to (10). The total sum of all values is a constant and does not depend on the GLCM itself, but only on the size of the original image (m and n).

$$p(i,j) = \left(\frac{1}{\sum_{k=1}^{m} \sum_{l=1}^{n} C_{\Delta x,\Delta y}(k,l)} \right) C_{\Delta x,\Delta y}(i,j) \qquad (10)$$

After this normalization is complete, a set of 4 different probability distributions is generated, one for each GLCM calculated previously. These probability distributions p(i,j) are the basis for the calculation of the image Haralick Features.

Haralick et al. defined a total of 14 different features which were proposed to characterize textures. From these original 14 features 13 of them have been used in this work[3]. These 13 Haralick features –whose equations can be found in Appendix A– are:

1. Energy (Angular Second Moment).
2. Contrast.
3. Correlation.
4. Sum of Squares: Variance.
5. Local Homogeneity (Inverse Difference Moment).
6. Sum Average.
7. Sum Variance.
8. Sum Entropy.
9. Boltzmann Entropy.
10. Difference Variance.
11. Difference Entropy.
12. Information of Correlation 1.
13. Information of Correlation 2.

As mentioned before in this work, Haralick suggested to obtain 4 different GLCMs, each one of them for each direction considered, and therefore calculate all desired features for each one of these GLCMs. Haralick et al. themselves explained in their original paper that in order to avoid co-occurrence matrices anisotropy the average values as well as the range (difference between maximum and minimum) values per feature for all 4 GLCMs. In other words, they proposed that, for any given image, GLCMs should be calculated for each direction. These 4 GLCMs would allow to obtain all features introduced before (Energy, Contrast, etc.), for each one of these directions. After that, all values could be averaged in order to obtain the average and range values of each feature for all 4 directions, aiming to break anisotropy of features. This leads to 6 different groups of features, shown in Fig. 5, which were used on this work. Considering that, a total of 78 Haralick parameters are extracted from every input image.

GLCM 0º	GLCM 45º	GLCM 90º
GLCM 135º	Average	Range

Figure 5: Different types of GLCMs and GLCM-related values considered in this work for extracting Haralick features.

3. OTHER TEXTURAL FEATURES

Several parameters have been proposed since Haralick et al. proposed their original features. In June 2007, M. Linek et al. used some Haralick-based features to find patterns in resistivity borehole images in order to classify the rocks in the wall of the drilled borehole [16]. An application of these parameters can be seen in [17]. Some of these features can extract very interesting information about the images, and the following ones were also considered in this work –see equations in Appendix B–:

14. Max. Probability

[3] The 14th feature, Maximal Correlation Coefficient, was not used in this work, due to its computational cost, so the algorithm used for the classification process could be faster

15. Cluster Shade

16. Cluster Prominence

Apart from these features, in 2004 M. Albuquerque et al. showed that Tsallis Entropy can be used instead of commonly used Boltzmann Entropy in thresholding segmentation techniques, getting more accurate results [18]. Thus, it was considered that the use of Tsallis Entropy might help to improve the success of the classifier.

Tsallis Statistics, also known as q-Statistics, is a type of Thermodynamics Statistics based on non-Extensive Entropy –Tsallis Entropy–. Several systems in physics cannot be correctly or precisely described by the classical Boltzmann-Gibbs Statistics due to non-extensivity or non-additivity of themselves. In these cases, Tsallis Statistics may provide better results than Boltzmann-Gibbs Statistics.

As C. Tsallis proposed in 2000, Entropic Non-extensivity may be a good parameter in order to quantify the complexity of a system [19]. Even non-extensivity being a property strongly related with entropy, there are some methods to estimate whether a system may be complex or simple.

Among all parameters used to consider the complexity of a system, two of them have been used in image processing before: the fractal dimension [20], [21], [22] and the Lyapunov Exponent [23].

The fractal dimension of an image and its validity were studied by P. Soille [24] in 1996. In this work, the fractal dimension of the image corresponds to the Hausdorff dimension MATLAB code implemented by A. Costa [25]. This dimension is a measure of how similar an image is to subdivisions of itself, for a given scale factor or, roughly speaking, it is a measure of how an image is composed by scaled versions of itself.

On the other hand, A. Wolf et al. proposed a method for calculating the Lyapunov Exponent for time-series data, on 1985 [26], and so did H. D. I. Abarbanel et al. in 1992 for observed data [27]. This exponent is a measure of the ergodicity of a physical system [28]. A system with a positive value of the Lyapunov Maximum exponent for a given dimension will cause that system not to converge (for an infinite time), and so it could be defined as a complex and sometimes random system; on the other hand, a null value for this exponent for a given dimension indicates that the system, for that dimension, has a tendency to converge in a closed orbit[4].

The purpose of this work is not to quantify the complexity of an image by this parameters, nor the ergodicity or chaotic behaviour. Thus, the features extracted here are not suitable for calculating the complexity of the image, neither the results shown in this work may be interpreted as a measure of these properties. The method proposed by A. Wolf for calculating the Lyapunov Exponent was modified in order to fit the classification requirements that a textural feature should have. Therefore, these parameters –Fractal Dimension and the Modified Lyapunov Exponent– are used here only as textural parameters due to the divergent values that they might have for different images, allowing to use them as classifying parameters and not, again, as a true measure of image complexity. These extra parameters are:

17. Tsallis Entropy (defined in Appendix C)

18. Fractal Dimension

19. Modified Lyapunov Exponent.

Parameters #14, #15 and #16 depend on the probability distribution, therefore will be calculated for each one of the 6 different GLCMs configurations explained before. On the other hand, features #18 and #19 are meant to be extracted from the original image, so they will not depend on the GLCM configurations.

Thus, considering all 19 parameters shown in this section, a total of 104 parameters (features #1 to #17 per 6 GLCM configurations plus features #18 and #19) will be obtained for each image. These parameters, after extracted from the original images, will be used as input data for the classifier.

An illustration of the configuration of all the features obtained for a given image is shown in Fig. 6.

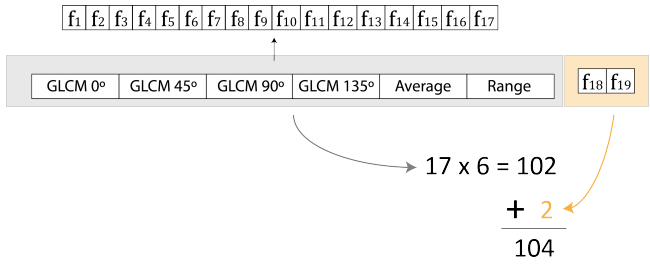

Figure 6: Illustration of all 104 parameters extracted from each processed image.

4. NAIVE BAYES CLASSIFICATION METHOD

When it comes to classifying data, several methods and techniques can be used. Some of them might be more sophisticate –such as Neural Networks–, while others may be based on simpler algorithms –like the one used in this work–. In that context, Naive Bayes classifiers are a group of simple probabilistic classifiers based on the bayesian Theorem.

These classifiers assume that all features –properties that define the class of each classification– are independent one from each other and do not interfere between them. Even though this assertion may not be precise for several cases where properties such as volume, contrast or homogeneity do actually depend one from each other for the studied object, several authors have shown that Naive Bayes is, after all, more robust and efficient that they were supposed to be, even when the used features do have strong dependence and correlation one with each other [29], [30].

For this reason, along with their simplicity, this kind of classifiers is one of the most popular ones [31], [32], [33], [34].

[4] For instance, a system with a closed orbit must have null values for, at least, some of their maximum lyapunov exponents, meaning that its movement is not chaotic, but defined and not ergodic –If the system is left even for infinite time not all possible locations will be covered, but only those positions contained on its orbit–.

In order to describe how Naive Bayes classifiers work, let us define a set of features extracted from an object which class is unknown $X = (x_1, x_2, \ldots, x_n)$. Each one of these features define one specific property of the object (for instance, one might represent its contrast, other one its entropy, etc.). On the other hand, let us assume that there exist two previously-defined classes, which will be referred as "A" and "B" in this example. It is possible to define the probability that the unclassified vector X belongs to any class C_k is given by (11).

$$p(C_k|X) = p(x_1, x_2, \ldots, x_n) = \frac{p(C_k)\, p(X|C_k)}{p(X)} \tag{11}$$

Now, as introduced before, the key assumption that bayesian classifiers make is that all features are independent one from each other and, so, they do not interfere between them. This assumption allow us to redefine the joint probability between $p(X|C_k)$ as (12), and so the joint probability for the unclassified vector X to belong to an specific class as (13).

$$p(X|C_k) = p(x_1|C_k)\, p(x_2|C_k) \ldots p(x_n|C_k) \tag{12}$$

$$p(C_k|X) = p(C_k) \prod_{i=1}^{n} p(x_i|C_k) \tag{13}$$

Following equation (13) we can see that, basically, the total probability of X data to belong to each class C_k is proportional to the product of individual probabilities for each feature to belong to that certain class. The decision rule, most times, is to simply assign the data X to the class that obtained the greatest probability or, in other words, the class which had the highest value for the product of individual features probabilities. This decision rule –which has also been used in this work– is shown in (14).

$$CLASS = \underset{k \exists 1,\ldots,K}{\arg\max} p(C_k) \prod_{i=1}^{n} p(x_i|C_k) \tag{14}$$

Now, there exists as many kinds of bayesian classifiers as ways there exists of calculating the joint probability $p(x_i|C_k)$. In this work the bayesian classifier used assumes that the probability distribution of samples –features extracted from the images– follows a gaussian distribution. For this kind of distribution the joint probability can be calculated as shown in (15)

$$p(x_i|C_k) = \frac{1}{\sqrt{2\pi\sigma_{k,i}}} e^{\frac{-\left(x_i - \mu_{k,i}\right)^2}{2\sigma_{k,i}^2}} \tag{15}$$

As it can be seen in equation (15), it is necessary to know the mean and standard deviation for each feature and each class –independently– in order to calculate the joint probability. Coming back to the previous example of groups "A" and "B", the statistical parameters that should be extracted from all features and classes in order to be able to calculate the joint probabilities using (15) would be: the mean value of group "A" for each feature in X, $\mu_A = (\mu_{A1}, \mu_{A2}, \ldots \mu_{An})$; the standard deviation for group "A" for each feature in X, $\sigma_A = (\sigma_{A1}, \sigma_{A2}, \ldots \sigma_{An})$; the mean value of group "B" for each feature in X, $\mu_B = (\mu_{B1}, \mu_{B2}, \ldots \mu_{Bn})$; and the standard deviation for group "B"' for each feature in X, $\sigma_B = (\sigma_{B1}, \sigma_{B2}, \ldots \sigma_{Bn})$. With these values, the joint probability of each feature to belong to a certain class can be evaluated and, therefore, the product of all of them will give a measure of the fitness of unclassified data for that certain class.

These statistical parameters can be obtained only from data that has been already classified. In other words, the classification process relies on the incorporation of previously classified data. For instance, it is impossible to classify oranges and apples if somebody has never seen or heard about apples or oranges at all.

Thus, the first step in the classification process must be to train the classifier. Data has to be collected and manually classified. Once all these measurements are introduced into the classifier, all statistical parameters of each feature are calculated and so new data can be classified according to these measurements. As the statistical parameters depend on the number of measurements, it is important to see that Bayesian classifiers can only be constructed if the initial training data has, at least, two measurements for each group.

The training step represents the most vital part of the process and initial data has to be used for the training process very carefully. The classification groups will depend strongly on the parameters measured and so the classification data should be reviewed and examined in order to determine its fitness in that particular case. For example, both oranges and apples can be round and have a similar diameter, so these features by themselves will not allow to build a robust and consistent 'apple-orange' classifier; instead, it would be more useful to use other features which are significantly different one from each other for every group, such as, in this case, color.

5. IMAGE FILTERING AND SPECTRAL ANALYSIS

Misclassifications, in bayesian classifiers, are often due to similarities between features of two different classes. The values of these features between classes might be very close one to each other, or similar in average or deviation values, confusing the classifier and generating errors on the classification process. In order to increase the classification success for this dataset and the set of features introduced in the previous sections a filtering process over original images has been proposed.

According to Signal Processing Theory [35], all the information of a discrete signal can be reproduced and extracted if the Shannon-Nyquist Theorem of minimum sample frequency is obeyed. For any image this theorem accomplishment is ensured by the company that manufactured the camera (in this case, an *Olympus C-3030ZOOM* Camera). The observation of the image spectrum is a good way to understand the grayscale (or color, if dealing with colored images)

information contained on that image.

As any other kind of spectrum, information contained inside an image spectrum can be divided, roughly, into three categories: High frequency, Mid frequency and Low Frequency. For a grayscaled image, high frequencies on the spectrum are usually interpreted as noise or big variances in the image grayscale, which usually means that the image has borders or edges. In the other hand, low frequencies indicate low variances in the image, which usually means that the image has areas without borders.

When a filter is applied to an image, its spectrum is modified in a way that some of its properties are attenuated, leading to a magnification of the rest of the properties –when compared to the ones attenuated-. For instance, edge detection filters can be understood, basically, as high-pass filters that only allow high frequencies –edges and borders– to remain on the image, erasing or attenuating low frequencies, and obtaining a processed image where only edges are conserved. This behavior is illustrated in Fig. 7, where an original image from the KTH-TIPS dataset –Aluminum Foil 1– has been processed first by the edge-detector Canny filter and by a 5x5 Box Normalized Mean filter (18) –separately–, and their respective spectra have been obtained.

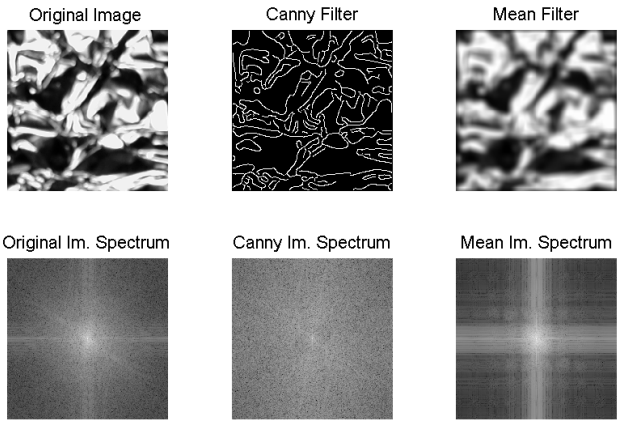

Figure 7: Different filtered images and their respective spectrum. (Left Top) Original Image without filters and (Left Bottom) its spectrum. (Mid Top) Original image with Canny –edge detection– filter and (Mid Bottom) its spectrum. (Right Top) Original image with mean filter and (Right Bottom) its spectrum.

Now, we could consider that edge detection filters magnify high frequencies –edges and borders– of any image, while average filters magnify low frequencies. Any textural feature extracted from an image processed by these two filters separately is expected to be very different.

On the other hand, even a filtered image –like the case shown in Fig. 7– represents the same physical object –in the case of the previous image, an Aluminum Foil–. The extraction of texture features for all three cases –original image, high-frequency filtered image and low-frequency filtered image– would increase the number of features extracted for each image, while these features may differ strongly between classes. If original images between two classes differ, it is expected that filtered images between these two same classes will clearly differ even more.

With this hypothesis in mind, the classification method has been repeated extracting all 104 features for three different images representing the same object: original image, original image filtered with an edge-detector filter and original image filtered with a Mean filter[5].

This configuration results in the extraction of a total amount of 312 features and is illustrated in Fig. 8.

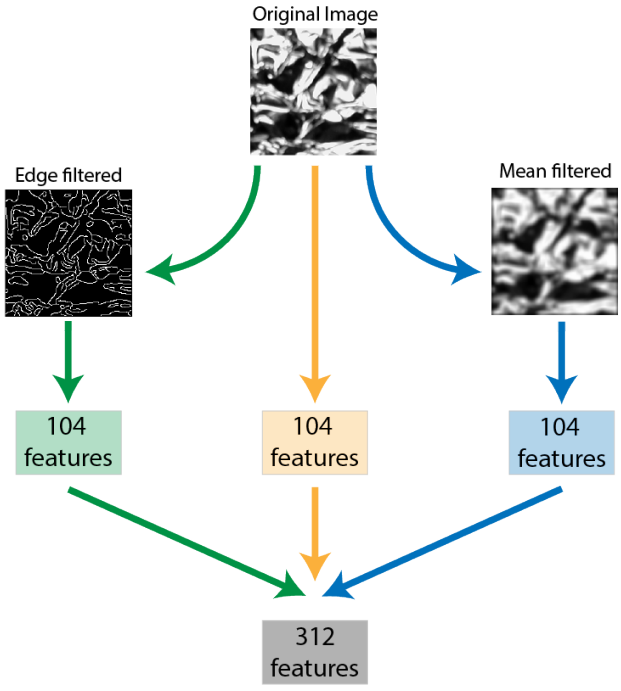

Figure 8: Illustration of the configuration used for extracting 104 features for each type of processed image (Original, Edge-filtered image and Mean-filtered image).

Obtaining the mean-filtered image can be achieved by convoluting the original image and the average filter[5] [21].

$$h = \frac{1}{5^2}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{16}$$

There exist plenty filters that allow to detect edges of any image. However, Canny filters have proved to be, over the last thirty years, one of the most suitable and effective when it comes to finding edges in images [36], [37], [38]. In this work this filter has been used for obtaining the edge-detected images. A more precise definition and explanation about the Canny filter can be read in John Canny′s original article [39] and also in a later review [40].

[5] The mean filter used on this illustration is the same used on the classification algorithm: a 5x5 Box normalized Mean Filter

6. TEXTURE CLASSIFICATION

6.1. Texture Classification Algorithm

The algorithm used for the classification process used in this work is illustrated on Fig. 9 and Fig. 10. All codes have been implemented in MATLAB. First of all the 810 images are imported, along with their correspondent class. Then, all images are filtered with the Canny and Mean filters, independently. After the filtering process, the texture features are extracted for each image (original and filtered ones), obtaining a matrix of 810 rows (original images) and 312 columns (104 features per type of image).

Once all textures have been extracted it is defined which filters will be considered. As there are 3 different images (original, canny-filtered and mean-filtered) there are 7 valid possibles, depending on the combination and use of these 3 kinds of images.

When the kind of images to be used in the classification process has been selected, all selected images are processed through a Principal Component Analysis (PCA) of the features. A PCA is a type of computational procedure that calculates a set of parameters for each input variable in a way that, when multiplied by the original data, all input features will become uncorrelated one from each other [41], [42]. As the extracted features proposed by Haralick et al. might be very correlated one with each other, this procedure is essential to guarantee the classification success. In this work the MATLAB default PCA function has been used.

After the Features matrix are processed with the PCA –becoming more uncorrelated than the original features– the dataset is randomly rearranged and then split into two groups: a training subset –which will train the classifier– and a testing subset –which will allow us to validate the classification process–. In this work the size used for the training subset was 40% of the original set of images –324 images–, while, obviously, the 60% left was used for testing the classifier.

As mentioned in section 4, the bayesian classifier used in this work is the gaussian one. In this case, when all 324 images with their respective features are obtained after the PCA, they should be introduced into the classifier, in order to train it. There is one point, however, worth mentioning: Naive Bayes classifiers only work properly if all features variance within any given class is greater than 0. If a class has zero variance for a certain feature all measurements are identical and, therefore, the standard deviation of the sample data is also zero. When calculating the joint probability for a certain feature and a certain class, the gaussian naive bayes assumes that the standard deviation of the sample will be greater than zero (See 15). Thus, all features with zero variance have to be ignored in order to be able to classify images with this method.

Once non-positive variance features have been suppressed, the features matrix can be introduced into the classifier and train in. After this, the features extracted from the testing subset can be introduced into the classifier, obtaining a prediction of class for each set of features. The calculation of the classification success can be achieved by simply comparing these predicted classes with the original ones from the imported set of images.

In this work this procedure has been repeated for a total of 10,000 different permutations for each different combination of filtering processes used.

6.2. About the KTH-TIPS Dataset

As mentioned before, the process of creation and configuration of classifiers depends on the essence of the system to be classified itself, and so all the features and training data should be suitable for that specific case.

The images used as samples for training and testing our classifier are part of dataset called KTH-TIPS that were firstly used by E. Hayman in 2004 [43] and shortly after that became available for public use. Since then this library of images has been widely used, as long as others, as examples of textures for image processing, analyzing, filtering and, of course, classification[44], [45], [46].

This dataset provides a total of 810 images, divided in 10 different classes, each one of them for a different material, with 81 pictures for each class. The materials, and therefore the classes –with their abbreviations–, found in this dataset are:

1. Sandpaper (SD)
2. Aluminum Foil (AL)
3. Styrofoam (SY)
4. Sponge (SP)
5. Corduroy (CY)
6. Linen (LI)
7. Cotton (CT)
8. Brown Bread (BB)
9. Orange Peel (OP)
10. Cracker Biscuit (CR)

For each class a total of 81 pictures were taken using 9 different scales (close or far to the object), 3 different angles in reference to the object (left, frontal and right) and 3 different illumination directions (frontal, from top and from aside). A sample of these textures can be seen in Fig. 11.

This group of unique combinations of illuminations, directions and scales increases the reliability of the dataset. As they were mainly thought to be used as training data for classifiers, the most different cases these pictures contemplate, the most reliable the classifier would get and so, theoretically, the better its predictions would be.

All the data and images used in this work for training the classifier and testing it came from this database. In order to remove the possible dependence and correlation between one picture and the one that was taken in sequence, they all have been mixed randomly, so when the dataset is divided into a training subset and a testing subset all possible cases are contemplated.

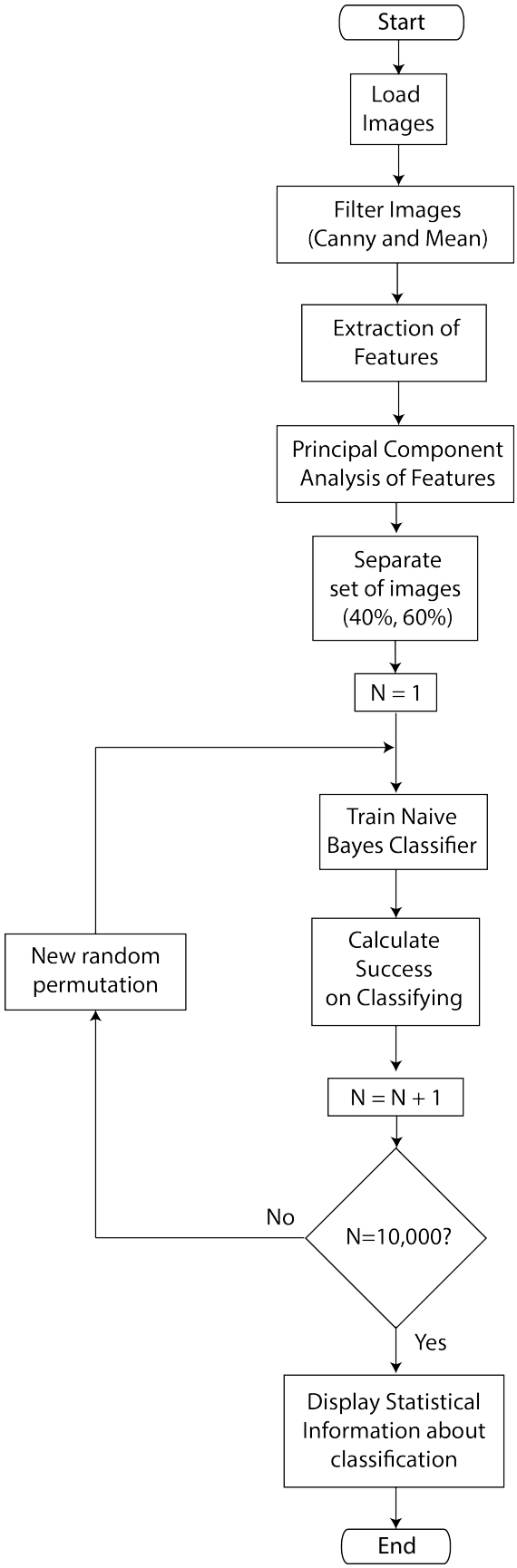

Figure 9: Organigram of the algorithm used in this work.

A more precise and extensive description of this database was written by its creators themselves and can be found at the website of the Royal Institute of Technology of Sweden (KTH) [47].

7. CLASSIFICATION RESULTS

Using the concepts explained on the previous sections, the classification success variation due to the use of filters on the images has been quantified. In order to do that, a set of 40% of the KTH-TIPS images have been used for training the Naive Bayes Classifier while the remaining 60% have been used for testing its success.

The classification process has been evaluated for both filtering methods separately in order to quantify the increase of classification for each kind of filter. On the other hand, the effect of PCA in the classification success was also evaluated and quantified. All different cases tested in the following points are shown in Table I.

Table I: Tested cases depending on the use of filters and PCA.

	Original Images	Canny Filtered	Mean Filtered	PCA
Case 1	Yes	No	No	No
Case 2	No	Yes	No	No
Case 3	Yes	Yes	No	No
Case 4	No	No	Yes	No
Case 5	Yes	No	Yes	No
Case 6	No	Yes	Yes	No
Case 7	Yes	Yes	Yes	No
Case 8	Yes	No	No	Yes
Case 9	No	Yes	No	Yes
Case 10	Yes	Yes	No	Yes
Case 11	No	No	Yes	Yes
Case 12	Yes	No	Yes	Yes
Case 13	No	Yes	Yes	Yes
Case 14	Yes	Yes	Yes	Yes

7.0.1. Case 1

First of all, when the Naive Bayes classifier is trained with 40% of the original textures, using all 104 features proposed in section 2 and section 3, without any filter nor PCA process, the mean success of the classifier in a total of 10,000 tests[6] is 72.28%, with a maximum value of 78.60% and a standard deviation of 1.97%. This result, for all the tested permutations, can be seen in Fig. 12.

The previous result means that, in average, about 30% of the classifier predicted classes are wrong. The average num-

[6] In each test a different and random permutation of images has been used, in order to avoid any possible correlation between the classification success and the order in which the images were used in the algorithm.

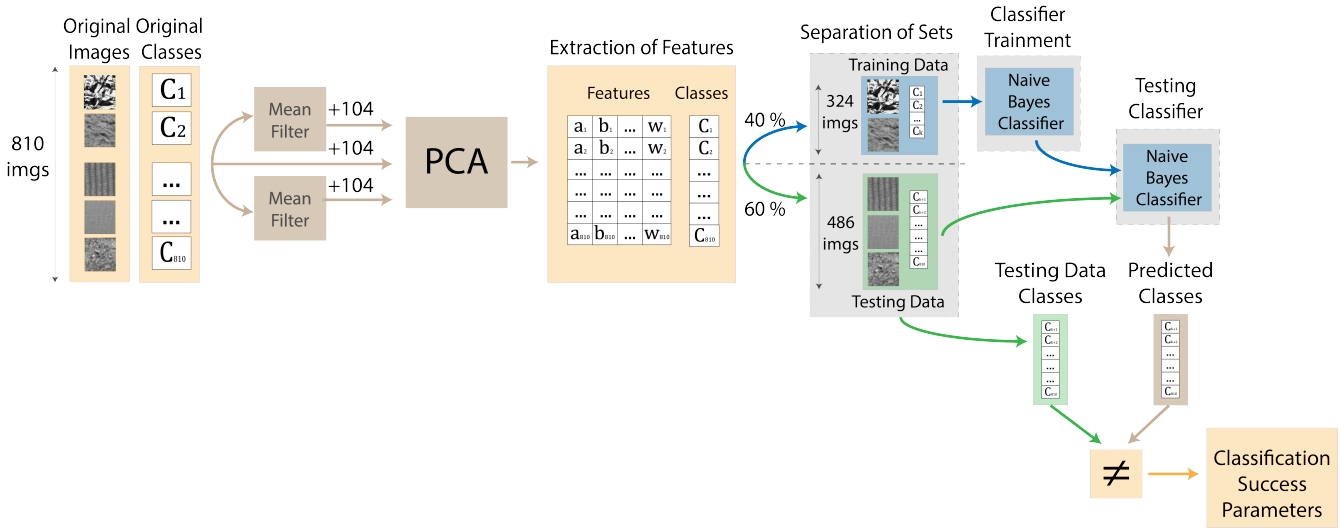

Figure 10: Illustration of the classification algorithm used in this work.

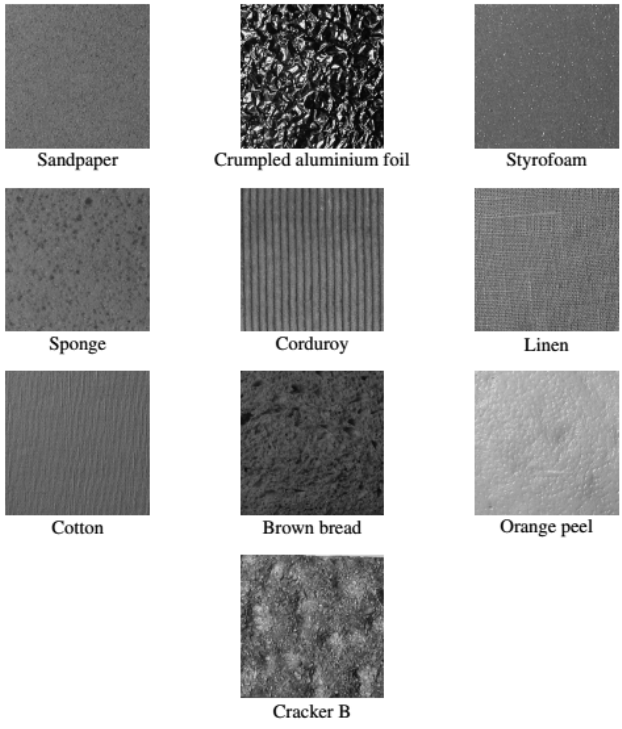

Figure 11: Sample images from all 10 different classes in the KTH-TIPS dataset. This image was extracted from [47] and all rights belong to the authors.

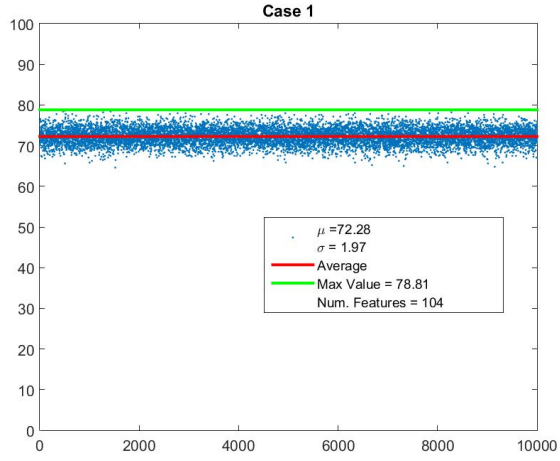

Figure 12: Classification success for 10,000 different permutations for Case 1, a training subset of 40% and a testing subset of 60%.

Table II: Misclassifications (in percent) for Case 1.

	BB	CY	CT	CR	LI	OP	SD	SP	SY
AL	0,0	0,4	0,9	0,4	0,0	0,3	0,0	0,0	0,0
BB		1,3	0,4	12,7	1,2	2,8	0,6	0,2	0,0
CY			2,5	4,5	11,7	0,6	0,3	0,0	0,0
CT				0,8	19,3	1,5	0,5	0,0	0,0
CR					5,8	4,3	3,9	0,1	0,0
LI						0,0	3,5	0,0	0,0
OP							2,1	2,8	5,2
SD								7,1	1,8
SP									0,1

ber of misclassifications in this case is 135. The relation between misclassifications and classes can be seen in Table II.[7]

As shown on Table II, the classifier has problems when it comes to distinguish between Cotton and Linen and between Brown Bread and Cracker Biscuits These results are easy to understand, as each one of the components of each pair of misclassifications is actually very similar to each other (Cotton and Linen are actually similar, and so are Brown Bread and Cracker Biscuits), even to our eyes.

[7] On this table permutations have been considered as equal cases, so for example, Aluminum Foil being misclassified as Cracker has been considered equivalent to Cracker being misclassified as Aluminum Foil.

7.1. Quantification of the Effect of Filtering on Classification

7.1.1. Case 2

When the Naive Bayes classifier is trained with 40% of the original textures, using all 104 features proposed in section 2 and section 3, without the PCA process, using only the canny-filtered images the mean success of the classifier in a total of 10,000 tests [6] is 52.07%, with a maximum value of 58.44% and a standard deviation of 1.63%. This result, for all the tested permutations, can be seen in Fig. 13.

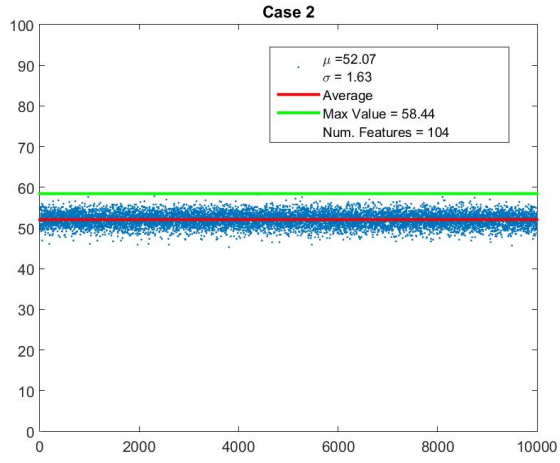

Figure 13: Classification success for 10,000 different permutations for Case 2, a training subset of 40% and a testing subset of 60%.

The previous result means that, in average, almost 50% of the classifier predicted classes are wrong. The average number of misclassifications in this case is 233. The relation between misclassifications and classes can be seen in Table III[7].

Table III: Misclassifications (in percent) for Case 2.

	BB	CY	CT	CR	LI	OP	SD	SP	SY
AL	3,9	0,9	0,0	3,8	0,0	7,5	1,7	4,3	2,2
BB		0,5	0,4	3,2	0,9	10,2	0,5	3,7	0,1
CY			3,4	0,9	1,0	1,2	1,6	0,3	0,4
CT				0,5	7,7	0,0	2,9	1,1	0,1
CR					0,1	0,5	0,8	6,9	7,2
LI						0,4	1,9	0,1	0,2
OP							0,1	0,2	0,1
SD								0,6	12,0
SP									3,8

7.1.2. Case 3

When the Naive Bayes classifier is trained with 40% of the original textures, using all 104 features proposed in section 2 and section 3, without the PCA process, using both original and the canny-filtered images the mean success of the classifier in a total of 10,000 tests [6] is 77.01%, with a maximum value of 83.33% and a standard deviation of 1.79%. This result, for all the tested permutations, can be seen in Fig. 14.

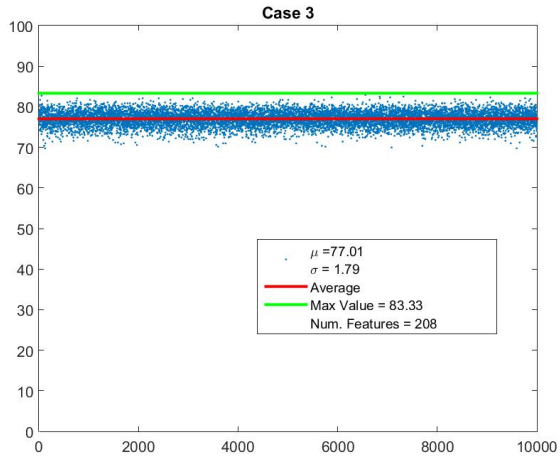

Figure 14: Classification success for 10,000 different permutations for Case 3, a training subset of 40% and a testing subset of 60%.

The previous result means that, in average, more than 25% of the classifier predicted classes are wrong. The average number of misclassifications in this case is 111. The relation between misclassifications and classes can be seen in Table IV[7].

Table IV: Misclassifications (in percent) for Case 3.

	BB	CY	CT	CR	LI	OP	SD	SP	SY
AL	0,0	0,3	1,8	2,4	0,0	0,6	0,0	0,0	0,0
BB		1,4	0,3	9,4	1,7	9,1	0,7	3,4	0,0
CY			5,6	4,0	7,4	1,3	0,1	0,0	0,0
CT				0,8	17,1	2,0	0,8	0,0	0,0
CR					0,6	3,5	5,8	0,6	0,0
LI						0,6	2,5	0,0	0,0
OP							2,4	3,3	2,8
SD								2,5	4,7
SP									0,4

7.1.3. Case 4

When the Naive Bayes classifier is trained with 40% of the original textures, using all 104 features proposed in section 2 and section 3, without the PCA process, using only the mean-filtered images the mean success of the classifier in a total of 10,000 tests [6] is 80.14%, with a maximum value of 85.60% and a standard deviation of 1.47%. This result, for all the tested permutations, can be seen in Fig. 15.

The previous result means that, in average, almost 20% of the classifier predicted classes are wrong. The average number of misclassifications in this case is 97. The relation between misclassifications and classes can be seen in Table V[7].

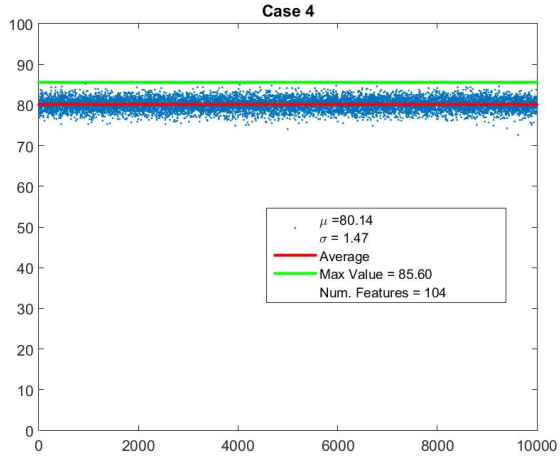

Figure 15: Classification success for 10,000 different permutations for Case 4, a training subset of 40% and a testing subset of 60%.

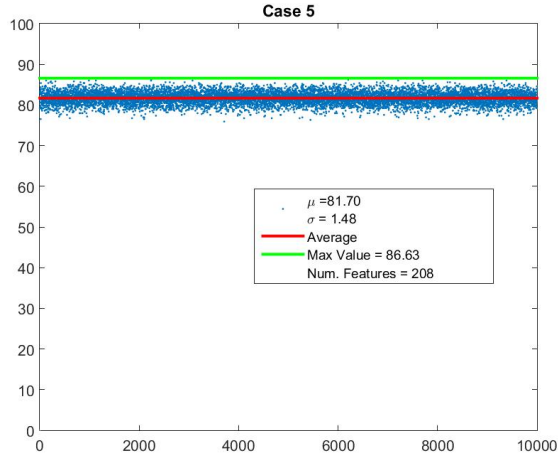

Figure 16: Classification success for 10,000 different permutations for Case 5, a training subset of 40% and a testing subset of 60%.

Table V: Misclassifications (in percent) for Case 4.

	BB	CY	CT	CR	LI	OP	SD	SP	SY
AL	0,0	0,0	0,0	0,1	0,0	0,3	0,0	0,0	0,0
BB		3,7	0,7	5,9	3,9	1,2	0,0	1,8	0,0
CY			4,0	0,6	12,1	0,0	0,0	0,4	0,0
CT				0,1	17,1	2,2	6,8	0,3	0,2
CR					0,0	1,0	3,8	1,4	0,8
LI						0,0	8,4	0,0	0,0
OP							2,6	3,6	4,1
SD								9,3	1,9
SP									1,5

Table VI: Misclassifications (in percent) for Case 5.

	BB	CY	CT	CR	LI	OP	SD	SP	SY
AL	0,0	0,0	0,1	0,8	0,0	0,2	0,0	0,0	0,0
BB		3,4	0,9	5,0	3,6	2,0	0,3	0,5	0,0
CY			3,2	1,2	16,4	0,2	0,0	0,3	0,0
CT				0,1	22,8	2,5	2,9	0,3	0,0
CR					0,1	1,4	5,6	0,4	0,7
LI						0,0	2,6	0,0	0,0
OP							3,6	3,9	5,2
SD								6,5	2,1
SP									1,0

7.1.4. *Case 5*

When the Naive Bayes classifier is trained with 40% of the original textures, using all 104 features proposed in section 2 and section 3, without the PCA process, using both original and mean-filtered images the mean success of the classifier in a total of 10,000 tests [6] is 81.70%, with a maximum value of 86.63% and a standard deviation of 1.48%. This result, for all the tested permutations, can be seen in Fig. 16.

The previous result means that, in average, almost 18% of the classifier predicted classes are wrong. The average number of misclassifications in this case is 89. The relation between misclassifications and classes can be seen in Table VI[7].

7.1.5. *Case 6*

When the Naive Bayes classifier is trained with 40% of the original textures, using all 104 features proposed in section 2 and section 3, without the PCA process, using both canny and mean filtered images the mean success of the classifier in a total of 10,000 tests [6] is 82.33%, with a maximum value of 87.04% and a standard deviation of 1.48%. This result, for all the tested permutations, can be seen in Fig. 17.

The previous result means that, in average, more than 15% of the classifier predicted classes are wrong. The average number of misclassifications in this case is 86. The relation between misclassifications and classes can be seen in Table VII[7].

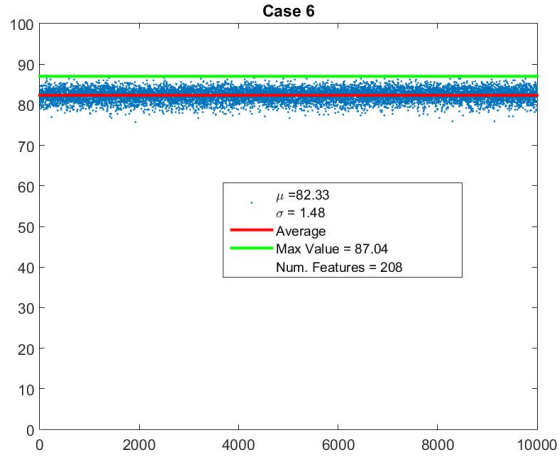

Figure 17: Classification success for 10,000 different permutations for Case 6, a training subset of 40% and a testing subset of 60%.

Table VII: Misclassifications (in percent) for Case 6.

	BB	CY	CT	CR	LI	OP	SD	SP	SY
AL	0,5	0,4	0,0	1,7	0,0	1,7	0,0	0,0	0,1
BB		6,6	1,1	5,6	4,3	2,1	0,6	3,3	0,0
CY			6,8	0,7	8,8	0,2	0,1	0,0	0,0
CT				0,2	16,0	2,7	6,4	0,0	0,3
CR					0,0	1,4	4,7	0,1	0,7
LI						0,0	4,6	0,0	0,0
OP							2,7	4,2	3,3
SD								2,3	5,1
SP									0,3

Table VIII: Misclassifications (in percent) for Case 7.

	BB	CY	CT	CR	LI	OP	SD	SP	SY
AL	0,0	0,2	0,2	1,6	0,0	0,2	0,0	0,0	0,0
BB		5,1	1,1	5,5	4,2	3,7	0,7	2,1	0,0
CY			6,0	1,8	12,7	0,2	0,0	0,0	0,0
CT				0,3	19,4	2,9	2,7	0,0	0,0
CR					0,0	2,5	5,9	0,1	0,7
LI						0,0	2,2	0,0	0,0
OP							3,2	4,2	3,8
SD								2,5	3,7
SP									0,2

7.1.6. Case 7

When the Naive Bayes classifier is trained with 40% of the original textures, using all 104 features proposed in section 2 and section 3, without the PCA process, using original images and both canny and mean filtered images the mean success of the classifier in a total of 10,000 tests [6] is 83.22%, with a maximum value of 87.65% and a standard deviation of 1.30%. This result, for all the tested permutations, can be seen in Fig. 18.

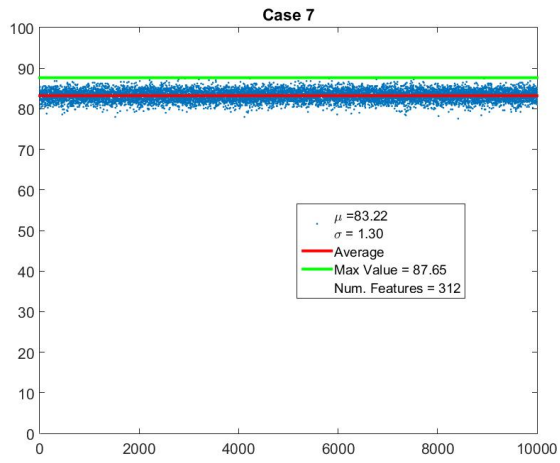

Figure 18: Classification success for 10,000 different permutations for Case 7, a training subset of 40% and a testing subset of 60%.

The previous result means that, in average, more than 15% of the classifier predicted classes are wrong. The average number of misclassifications in this case is 81. The relation between misclassifications and classes can be seen in Table VIII[7].

7.2. Quantification of the Effect of PCA on Classification

7.2.1. Case 8

When the Naive Bayes classifier is trained with 40% of the original textures, using all 104 features proposed in section 2 and section 3, processing the features with the PCA algorithm, using only the original images, without any filter, the mean success of the classifier in a total of 10,000 tests [6] is 82.35%, with a maximum value of 89.09% and a standard deviation of 1.90%. After the PCA it was found that the total number of suitable uncorrelated features for this case was 87. This result, for all the tested permutations, can be seen in Fig. 19.

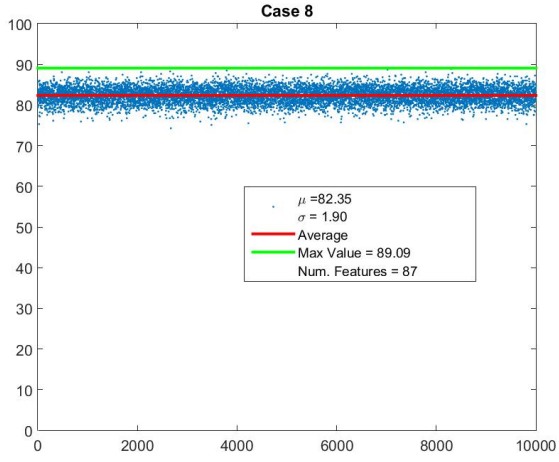

Figure 19: Classification success for 10,000 different permutations for Case 8, a training subset of 40% and a testing subset of 60%.

The previous result means that, in average, more than 15% of the classifier predicted classes are wrong. The average number of misclassifications in this case is 87. The relation between misclassifications and classes can be seen in Table IX[7].

Table IX: Misclassifications (in percent) for Case 8.

	BB	CY	CT	CR	LI	OP	SD	SP	SY
AL	0,2	0,1	1,0	0,0	0,1	3,0	0,0	0,0	0,0
BB		0,4	3,0	5,0	0,1	3,0	0,2	4,0	0,0
CY			7,1	2,6	5,1	0,6	1,2	1,2	0,0
CT				4,7	17,0	1,7	1,5	0,9	0,1
CR					0,7	2,4	8,9	2,9	0,3
LI						0,8	0,0	0,0	0,6
OP							3,3	3,5	5,1
SD								3,7	3,4
SP									0,4

7.2.2. Case 9

When the Naive Bayes classifier is trained with 40% of the original textures, using all 104 features proposed in section 2 and section 3, processing the features with the PCA algorithm, using only the canny-filtered images, the mean success of the classifier in a total of 10,000 tests [6] is 58.90%, with a maximum value of 65.43% and a standard deviation of 1.88%. After the PCA it was found that the total number of suitable uncorrelated features for this case was 70. This result, for all the tested permutations, can be seen in Fig. 20.

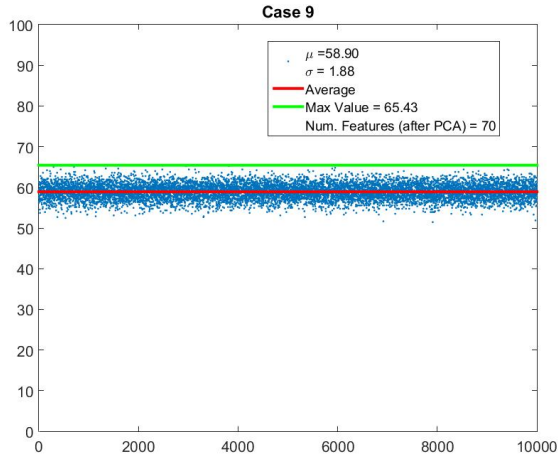

Figure 20: Classification success for 10,000 different permutations for Case 9, a training subset of 40% and a testing subset of 60%.

The previous result means that, in average, more than 40% of the classifier predicted classes are wrong. The average number of misclassifications in this case is 200. The relation between misclassifications and classes can be seen in Table X[7].

Table X: Misclassifications (in percent) for Case 9.

	BB	CY	CT	CR	LI	OP	SD	SP	SY
AL	4,5	2,0	1,4	4,4	0,6	8,0	2,8	3,0	2,6
BB		0,1	0,1	1,8	0,0	5,7	0,3	4,1	0,1
CY			4,1	0,2	3,7	2,9	0,1	0,0	0,1
CT				0,9	7,8	1,3	4,3	0,0	0,2
CR					0,0	0,4	0,3	6,3	5,0
LI						1,5	0,0	0,0	0,0
OP							0,9	0,8	0,1
SD								1,0	10,2
SP									6,6

7.2.3. Case 10

When the Naive Bayes classifier is trained with 40% of the original textures, using all 104 features proposed in section 2 and section 3, processing the features with the PCA algorithm, using both original and canny-filtered images, the mean success of the classifier in a total of 10,000 tests [6] is 83.28%, with a maximum value of 89.71% and a standard deviation of 1.80%. After the PCA it was found that the total number of suitable uncorrelated features for this case was 157. This result, for all the tested permutations, can be seen in Fig. 21.

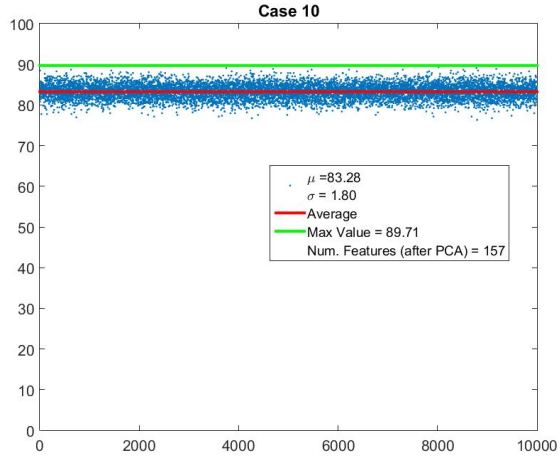

Figure 21: Classification success for 10,000 different permutations for Case 10, a training subset of 40% and a testing subset of 60%.

The previous result means that, in average, more than 15% of the classifier predicted classes are wrong. The average number of misclassifications in this case is 81. The relation between misclassifications and classes can be seen in Table XI[7].

Table XI: Misclassifications (in percent) for Case 10.

	BB	CY	CT	CR	LI	OP	SD	SP	SY
AL	0,0	0,0	1,0	0,0	0,1	2,8	0,0	0,0	0,0
BB		1,7	2,5	5,2	0,1	3,9	1,1	5,8	0,1
CY			10,3	4,2	5,1	0,6	0,5	0,4	0,0
CT				2,6	16,2	2,3	0,4	0,0	0,1
CR					0,6	1,7	7,6	1,8	0,5
LI						1,7	0,0	0,0	0,5
OP							4,4	3,3	5,7
SD								1,1	4,1
SP									0,0

7.2.4. Case 11

When the Naive Bayes classifier is trained with 40% of the original textures, using all 104 features proposed in section 2 and section 3, processing the features with the PCA algorithm, using only the mean-filtered images, the mean success of the classifier in a total of 10,000 tests [6] is 79.63%, with a maximum value of 86.01% and a standard deviation of 1.60%. After the PCA it was found that the total number of suitable uncorrelated features for this case was 86. This result, for all the tested permutations, can be seen in Fig. 22.

The previous result means that, in average, about 20% of the classifier predicted classes are wrong. The average

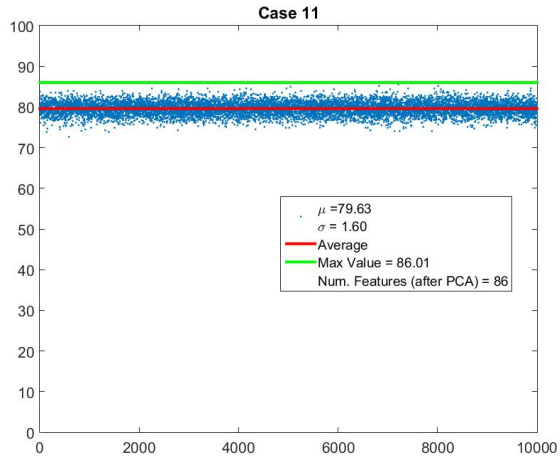

Figure 22: Classification success for 10,000 different permutations for Case 11, a training subset of 40% and a testing subset of 60%.

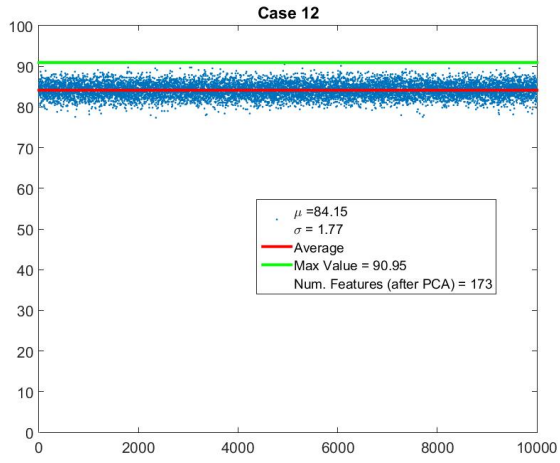

Figure 23: Classification success for 10,000 different permutations for Case 12, a training subset of 40% and a testing subset of 60%.

number of misclassifications in this case is 99. The relation between misclassifications and classes can be seen in Table XII[7].

Table XII: Misclassifications (in percent) for Case 11.

	BB	CY	CT	CR	LI	OP	SD	SP	SY
AL	2,6	1,6	0,5	3,2	1,5	4,5	0,0	0,9	0,0
BB		1,6	0,1	3,5	0,3	0,8	0,9	2,8	0,1
CY			4,5	2,6	8,3	0,8	2,6	0,9	0,0
CT				0,5	8,9	2,1	11,5	1,2	2,9
CR					0,0	2,2	2,6	1,4	0,4
LI						1,4	4,7	0,1	0,0
OP							2,0	3,6	2,1
SD								4,2	2,9
SP									0,9

7.2.5. Case 12

When the Naive Bayes classifier is trained with 40% of the original textures, using all 104 features proposed in section 2 and section 3, processing the features with the PCA algorithm, using both original and mean-filtered images, the mean success of the classifier in a total of 10,000 tests [6] is 84.15%, with a maximum value of 90.95% and a standard deviation of 1.77%. After the PCA it was found that the total number of suitable uncorrelated features for this case was 173. This result, for all the tested permutations, can be seen in Fig. 23.

The previous result means that, in average, about 15% of the classifier predicted classes are wrong. The average number of misclassifications in this case is 77. The relation between misclassifications and classes can be seen in Table XIII[7].

Table XIII: Misclassifications (in percent) for Case 12.

	BB	CY	CT	CR	LI	OP	SD	SP	SY
AL	1,6	1,8	0,1	3,9	0,4	6,3	0,0	0,3	0,4
BB		1,3	3,0	4,8	1,1	4,0	0,6	2,5	0,3
CY			7,7	2,4	4,7	0,5	0,9	0,4	0,0
CT				1,0	17,2	3,2	2,1	0,7	0,3
CR					0,1	0,9	6,7	0,2	0,1
LI						0,2	0,0	0,8	0,7
OP							2,6	3,6	6,3
SD								1,1	2,9
SP									0,2

7.2.6. Case 13

When the Naive Bayes classifier is trained with 40% of the original textures, using all 104 features proposed in section 2 and section 3, processing the features with the PCA algorithm, using only the canny and mean-filtered images, the mean success of the classifier in a total of 10,000 tests [6] is 81.91%, with a maximum value of 88.68% and a standard deviation of 1.73%. After the PCA it was found that the total number of suitable uncorrelated features for this case was 156. This result, for all the tested permutations, can be seen in Fig. 24.

The previous result means that, in average, almost 20% of the classifier predicted classes are wrong. The average number of misclassifications in this case is 88. The relation between misclassifications and classes can be seen in Table XIV[7].

7.2.7. Case 14

Lastly, when the Naive Bayes classifier is trained with 40% of the original textures, using all 104 features proposed in section 2 and section 3, processing the features with the PCA algorithm, using original images along with the canny and mean-filtered ones, the mean success of the classifier in

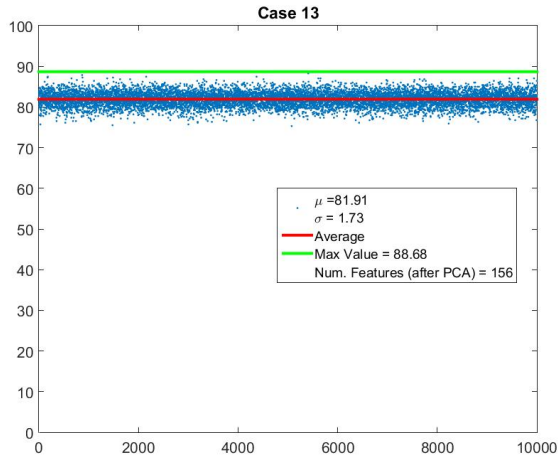

Figure 24: Classification success for 10,000 different permutations for Case 13, a training subset of 40% and a testing subset of 60%.

Table XIV: Misclassifications (in percent) for Case 13.

	BB	CY	CT	CR	LI	OP	SD	SP	SY
AL	3,5	0,7	0,0	4,1	0,0	5,1	0,9	1,5	0,8
BB		0,9	0,1	5,5	0,0	1,0	1,0	4,0	0,0
CY			6,7	1,5	7,6	2,9	0,2	0,6	0,0
CT				0,1	12,7	2,3	7,4	0,2	0,6
CR					0,5	2,9	4,7	0,4	0,1
LI						1,9	1,4	0,0	0,0
OP							3,6	3,2	3,5
SD								2,1	3,2
SP									0,3

a total of 10,000 tests [6] is 84.24%, with a maximum value of 90.12% and a standard deviation of 1.78%. After the PCA it was found that the total number of suitable uncorrelated features for this case was 243. This result, for all the tested permutations, can be seen in Fig. 25.

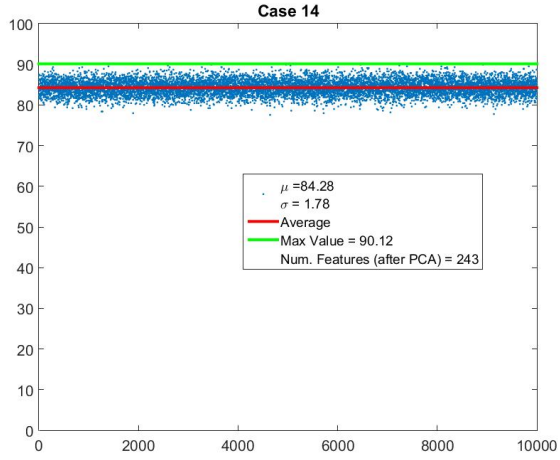

Figure 25: Classification success for 10,000 different permutations for Case 14, a training subset of 40% and a testing subset of 60%.

The previous result means that, in average, about 15% of the classifier predicted classes are wrong. The average number of misclassifications in this case is 88. The relation between misclassifications and classes can be seen in Table XVI[7].

Table XV: Misclassifications (in percent) for Case 14.

	BB	CY	CT	CR	LI	OP	SD	SP	SY
AL	1,8	0,3	1,0	4,5	0,0	4,6	0,0	1,7	0,5
BB		1,0	1,4	6,1	0,1	2,9	0,4	2,3	0,4
CY			8,7	2,0	6,2	1,2	1,4	0,6	0,4
CT				0,2	17,2	5,9	1,2	0,1	0,0
CR					1,1	2,4	4,6	0,5	0,1
LI						0,5	0,1	0,3	0,3
OP							2,9	2,3	6,3
SD								1,6	2,7
SP									0,1

7.3. Comparison of Results

Finally, Table XV compares the classification results for all cases while Fig. 26 shows the different success in classifying for each case (blue and green bars), along with the number of features used (yellow area).

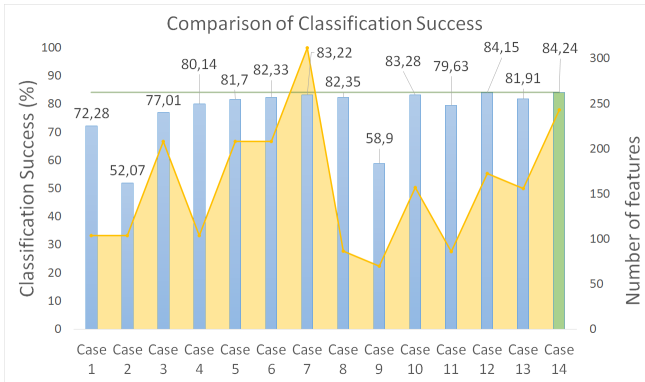

Figure 26: Comparison Chart between Misclassification for Original Images only (blue bars) and filtered images (red bars).

When analyzing the classification success results in Table XV and Fig. 40 it is possible to notice that:

1. The Canny filter, when used combined with the original images, without PCA, increased the classification success in about 5% –in comparison to the classification success for the original images–; and less than 1% when the features were processed by the PCA. On the other hand, when the canny filter images were used on their own, without the original images, they decreased the classification success in almost 20% for both cases –with and without PCA–.

2. The Mean filter, when used on its own, without the original images, increased the classification success in about 8% –in comparison to the classification success for the original images–; but decreased that success

Table XVI: Comparison of the classification success for all cases.

	μ^a	σ^b	Max. Success	N.F.^{c}	Misc.^{d}	P.M^e
Case 1	72.28	1.97	78.60	104	135	31
Case 2	52.07	1.63	58.44	104	233	42
Case 3	77.01	1.79	83.33	208	111	32
Case 4	80.14	1.47	85.60	104	97	29
Case 5	81.70	1.48	86.63	208	89	32
Case 6	82.33	1.48	87.04	208	86	33
Case 7	83.22	1.30	87.65	312	81	30
Case 8	82.35	1.90	89.09	87	86	37
Case 9	58.90	1.88	65.43	70	200	38
Case 10	83.28	1.80	89.71	157	81	34
Case 11	79.63	1.60	86.01	86	99	40
Case 12	84.15	1.77	90.95	173	77	42
Case 13	81.91	1.73	88.68	156	88	38
Case 14	84.24	1.78	90.12	243	76	42

[a] Average Success.
[b] Success Standard Deviation.
[c] Number of Features (after PCA, if used).
[d] Average number of misclassifications.
[e] Number of Misclassified Pairs. This is a measure of the dispersion of errors among the different classes during classification.

in about 3% when the features were processed by the PCA. Additionally, when the mean filter images were used along with the original images, they increased the classification success in almost 10%, without PCA, and almost 2%, with PCA.

3. When both Mean and Canny filters were combined, without the original images, the classification success was increased in about 10% –in comparison to the classification success for the original images–, without PCA; however, it was decreased in almost 1% when the features were processed by the PCA. Finally, when the both filters were used along with the original images, the classification success was increased in almost 11%, without PCA, and about 2%, with PCA.

4. The PCA process increased significantly the classification success for all cases except two: Case 4 and Case 6. The average increase in this success due to PCA was 5.5%.

Analyzing Table XV it is also possible to notice that:

1. 8 combinations of classes (among all 45 possible) cause 50% of the total misclassifications committed in the process, on their own.

2. 2 combinations of classes (among all 45 possible) cause almost 25% of the total misclassifications commited in the process, on their own. These combinations are: Cotton and Linen, and Corduroy and Linen.

8. CONCLUSIONS

We presented in this work a work-flow for increasing the classification success for bayesian classifiers by using image filters and Principal Component Analysis for optimizing the number and values of textural features. The results shown in the previous sections allow us to conclude that:

1. The best results were obtained when using both Canny and Mean filtered images along with the original images and processing the features using the PCA (Case 14). In this case the classification success achieved almost 85%, in average, and more than 90% for certain permutations.

2. The use of PCA increased –in average– the classification success in almost all cases, reducing the number of features for each analyzed case, and therefore reducing the total computational cost of obtaining these parameters.

3. When used on their own, the Canny filtered images decreased dramatically the classification result (Case 2 and Case 9). Thus, this filter should not be considered for image classification when used alone. This result may be result of the images noise. The presence of high-frequency noise modifies the information contained in an image [48]. High-pass filters, like Canny filters, may magnify the effect of this noise and, therefore, decrease the classification success. In further studies a reduction of this noise in the original images is expected to improve these results.

4. The use of the Mean filter along with the original images, with or without PCA (Case 5 and Case 12), also increased the classification result. Even though this increase was lower than the increased produced for Case 14, in further applications it should be taken into account the computational cost due to the Canny-filtering process and compared to the classification success. The number of features in Case 12, after the PCA process, was reduced to 173, while in Case 14 the number of these features was 243. The difference in classification success between these two cases was only 0.09%. Thus, Case 13 could be considered a lower-cost higher-performance method than Case 14.

5. The use of simple filters –like the ones used in this work– increased the performance of the classifier in average and maximum values for cases (Case 3, Case 5, Case 7, Case 10, Case 12 and Case 14), although the performance increase when combining original images and filtered lays inside one sigma. It could be interesting to study the effect of different filters on this performance. For instance, more sophisticate mean filters, like gaussian filters, may lead to better results.

6. For certain permutations, the maximum classification success by using methods in Cases 12 and 14 reached almost 91%.

On the other hand, the implementation of these methods with more advanced types of classifiers, like Neural Networks, has not been tested. However, it is also expected that these methods may increase the classification success rates and performance, allowing to reach higher levels of success than the ones showed in this paper.

Acknowledgement

This work was made possible by cooperation agreement between CENPES/PETROBRÁS and CPBF and was supported by CARMOD thematic funding for Researches in Carbonates.

Bibliography

[1] Marcos William da Silva Oliveira, Nubia Rosa da Silva, Antoine Manzanera, and Odemir Martinez Bruno. Feature extraction on local jet space for texture classification. *Physica A: Statistical Mechanics and its Applications*, 439:160 – 170, 2015.

[2] Degang Xu, Xiao Chen, Yongfang Xie, Chunhua Yang, and Weihua Gui. Complex networks-based texture extraction and classification method for mineral flotation froth images. *Minerals Engineering*, 83:105 – 116, 2015.

[3] Arvind R. Yadav, R.S. Anand, M.L. Dewal, and Sangeeta Gupta. Multiresolution local binary pattern variants based texture feature extraction techniques for efficient classification of microscopic images of hardwood species. *Applied Soft Computing*, 32:101 – 112, 2015.

[4] Suganya Ramamoorthy, R. Kirubakaran, and Rajaram Siva Subramanian. Texture Feature Extraction Using MGRLBP Method for Medical Image Classification. In Suresh, LP and Dash, SS and Panigrahi, BK, editor, *ARTIFICIAL INTELLIGENCE AND EVOLUTIONARY ALGORITHMS IN ENGINEERING SYSTEMS, VOL 1*, volume 324 of *Advances in Intelligent Systems and Computing*, pages 747–753, 2015. International Conference on Artificial Intelligence and Evolutionary Algorithms in Engineering Systems (ICAEES), Noorul Islam Univ, Noorul Islam Ctr Higher Educ, Kumaracoil, INDIA, APR 22-23, 2014.

[5] Hong Li, Xieping Xu, Buer Qi, Nan Bao, Yaonan Zhang, Hang Sun, Liwei Yu, and Yan Kang. An Effective Feature Extraction Method on Mammograms: A Band Shaped Texture Analysis Based on Iris Filter. *JOURNAL OF MEDICAL IMAGING AND HEALTH INFORMATICS*, 4(5):787–792, OCT 2014.

[6] D. Abraham Chandy, J. Stanly Johnson, and S. Easter Selvan. Texture feature extraction using gray level statistical matrix for content-based mammogram retrieval. *MULTIMEDIA TOOLS AND APPLICATIONS*, 72(2):2011–2024, SEP 2014.

[7] Chao Peng, Jian-Ming Zheng, Xu-Bo Li, Yan-Chao Song, and Jiao-Jiao Shi. Feature extraction on machined surface texture image of tool wear based on fractional brown motion. In Shahhosseini, AM, editor, *DESIGN, MANUFACTURING AND MECHATRONICS (ICDMM 2015)*, pages 706–714. Hebei Univ; Beijing Technol & Business Univ; Chengdu Univ, 2015. International Conference on Design, Manufacturing and Mechatronics (ICDMM), Adv Sci Technol & Ind Res Ctr, Wuhan, PEOPLES R CHINA, APR 17-18, 2015.

[8] Satrajit Mukherjee, Bodhisattwa Prasad Majumder, Aritran Piplai, and Swagatam Das. An Adaptive Differential Evolution Based Fuzzy Approach for Edge Detection in Color and Grayscale Images. In Panigrahi, BK and Suganthan, PN and Das, S and Dash, SS, editor, *SWARM, EVOLUTIONARY, AND MEMETIC COMPUTING, PT I (SEMCCO 2013)*, volume 8297 of *Lecture Notes in Computer Science*, pages 260–273, 2013. 4th International Conference on Swarm, Evolutionary, and Memetic Computing (SEMCCO), SRM Univ, Chennai, INDIA, DEC 19-21, 2013.

[9] Daniel Merkel, Eckard Brinkmann, Joerg C. Kaemmer, Miriam Koehler, Daniel Wiens, and Karl-Michael Derwahl. Comparison Between Various Color Spectra and Conventional Grayscale Imaging for Detection of Parenchymal Liver Lesions With B-Mode Sonography. *JOURNAL OF ULTRASOUND IN MEDICINE*, 34(9):1529–1534, SEP 2015.

[10] Jing Hu, Daoliang Li, Qingling Duan, Guifen Chen, and Yeiqi Han. Texture Extraction and Analysis by Statistical Methods for Fish Species Classification. *SENSOR LETTERS*, 11(6-7, SI):1110–1114, JUN-JUL 2013.

[11] Harun Gunes, Elif Nisa Unlu, and Ayhan Saritas. CT versus grayscale rib series for the detection of rib fracture. *AMERICAN JOURNAL OF EMERGENCY MEDICINE*, 33(10):1515–1516, OCT 2015.

[12] R.M. Haralick, K. Shanmugam, and Its'Hak Dinstein. Textural features for image classification. *Systems, Man and Cybernetics, IEEE Transactions on*, SMC-3(6):610–621, Nov 1973.

[13] BlairD. Fleet, Jinyao Yan, DavidB. Knoester, Meng Yao, Jr. Deller, JohnR., and ErikD. Goodman. Breast cancer detection using haralick features of images reconstructed from ultra wideband microwave scans. In Marius George Linguraru, Cristina Oyarzun Laura, Raj Shekhar, Stefan Wesarg, Miguel Angel Gonzalez Ballester, Klaus Drechsler, Yoshinobu Sato, and Marius Erdt, editors, *Clinical Image-Based Procedures. Translational Research in Medical Imaging*, volume 8680 of *Lecture Notes in Computer Science*, pages 9–16. Springer International Publishing, 2014.

[14] Michael V Boland, Mia K Markey, Robert F Murphy, et al. Automated recognition of patterns characteristic of subcellular structures in fluorescence microscopy images. *Cytometry*, 33(3):366–375, 1998.

[15] Neelamma K Patil, Virendra S Malemath, and Ravi M Yadahalli. Color and texture based identification and classification of food grains using different color models and haralick features. *International Journal on Computer Science and Engineering*, 3(12):3669, 2011.

[16] Margarete Linek, Matthias Jungmann, Thomas Berlage, Renate Pechnig, and Christoph Clauser. Rock classification based on resistivity patterns in electrical borehole wall images. *Journal of Geophysics and Engineering*, 4(2):171, 2007.

[17] Xiaofeng Yang, Srini Tridandapani, Jonathan J Beitler, S Yu David, Emi J Yoshida, Walter J Curran, and Tian Liu. Ultrasound glcm texture analysis of radiation-induced parotid-gland injury in head-and-neck cancer radiotherapy: an in vivo study of late toxicity. *Medical physics*, 39(9):5732–5739, 2012.

[18] M Portes de Albuquerque, IA Esquef, and AR Gesualdi Mello. Image thresholding using tsallis entropy. *Pattern Recognition Letters*, 25(9):1059–1065, 2004.

[19] Constantino Tsallis. Entropic nonextensivity: a possible measure of complexity. *Chaos, Solitons & Fractals*, 13(3):371–391, 2002.

[20] Lucas Correia Ribas, Diogo Nunes Goncalves, Jonatan Patrick Margarido Orue, and Wesley Nunes Goncalves. Fractal dimension of maximum response filters applied to texture analysis. *PATTERN RECOGNITION LETTERS*, 65:116–123, NOV 1 2015.

[21] Igor Pantic, Sanja Dacic, Predrag Brkic, Irena Lavrnja, Tomislav Jovanovic, Senka Pantic, and Sanja Pekovic. Discriminatory ability of fractal and grey level co-occurrence matrix methods in structural analysis of hippocampus layers. *JOURNAL OF THEORETICAL BIOLOGY*, 370:151–156, APR 7 2015.

[22] Alvaro G. Zuniga, Joao B. Florindo, and Odemir M. Bruno. Gabor wavelets combined with volumetric fractal dimension applied to texture analysis. *PATTERN RECOGNITION LET-*

TERS, 36:135–143, JAN 15 2014.

[23] Xiuling Liu, Haiman Du, Guanglei Wang, Suiping Zhou, and Hong Zhang. Automatic diagnosis of premature ventricular contraction based on Lyapunov exponents and LVQ neural network. *COMPUTER METHODS AND PROGRAMS IN BIOMEDICINE*, 122(1):47–55, OCT 2015.

[24] Pierre Soille and Jean-F Rivest. On the validity of fractal dimension measurements in image analysis. *Journal of visual communication and image representation*, 7(3):217–229, 1996.

[25] Alceu Costa. Hausdorff fractal dimension calculation using the box-counting method code for matlab. 2013.

[26] Alan Wolf, Jack B Swift, Harry L Swinney, and John A Vastano. Determining lyapunov exponents from a time series. *Physica D: Nonlinear Phenomena*, 16(3):285–317, 1985.

[27] Henry DI Abarbanel, Reggie Brown, and Matthew B Kennel. Local lyapunov exponents computed from observed data. *Journal of Nonlinear Science*, 2(3):343–365, 1992.

[28] C. Varsakelis and P. Anagnostidis. On the susceptibility of numerical methods to computational chaos and superstability. *Communications in Nonlinear Science and Numerical Simulation*, 33:118 – 132, 2016.

[29] Harry Zhang. The optimality of naive bayes. *AA*, 1(2):3, 2004.

[30] David J Hand and Keming Yu. Idiot's bayesânot so stupid after all? *International statistical review*, 69(3):385–398, 2001.

[31] Alper Kursat Uysal. An improved global feature selection scheme for text classification. *Expert Systems with Applications*, 43:82 – 92, 2016.

[32] Xiang Ji, Soon Ae Chun, Zhi Wei, and James Geller. Twitter sentiment classification for measuring public health concerns. *SOCIAL NETWORK ANALYSIS AND MINING*, 5(1), DEC 2015.

[33] Jie Lu, Khondaker A. Mamun, and Tom Chau. Pattern classification to optimize the performance of Transcranial Doppler Ultrasonography-based brain machine interface. *PATTERN RECOGNITION LETTERS*, 66:135–143, NOV 15 2015.

[34] Guozhong Feng, Jianhua Guo, Bing-Yi Jing, and Tieli Sun. Feature subset selection using naive Bayes for text classification. *PATTERN RECOGNITION LETTERS*, 65:109–115, NOV 1 2015.

[35] Paulo SR Diniz, Eduardo AB Da Silva, and Sergio L Netto. *Digital signal processing: system analysis and design*. Cambridge University Press, 2010.

[36] Ya-Hui Xiu and Wen-Qing Wu. A novel edge detection algorithm. In Shahhosseini, AM, editor, *DESIGN, MANUFACTURING AND MECHATRONICS (ICDMM 2015)*, pages 635–640. Hebei Univ; Beijing Technol & Business Univ; Chengdu Univ, 2016. International Conference on Design, Manufacturing and Mechatronics (ICDMM), Adv Sci Technol & Ind Res Ctr, Wuhan, PEOPLES R CHINA, APR 17-18, 2015.

[37] Daniel Tchiotsop, Beaudelaire Saha Tchinda, Rene Tchinda, and Godpromesse Kenne. Edge detection of intestinal parasites in stool microscopic images using multi-scale wavelet transform. *SIGNAL IMAGE AND VIDEO PROCESSING*, 9(1, SI):121–134, DEC 2015.

[38] Jaseema Yasmin and Mohamed Sathik. An Improved Iterative Segmentation Algorithm using Canny Edge Detector for Skin Lesion Border Detection. *INTERNATIONAL ARAB JOURNAL OF INFORMATION TECHNOLOGY*, 12(4):325–332, JUL 2015.

[39] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.

[40] Thomas Moeslund. Canny edge detection. *Laboratory of Computer Vision and Media Technology, Aalborg University, Denmark, http://www. cvmt. dk/education/teaching/f09/VGIS8/AIP/canny_09gr820. pdf*, 2009.

[41] Svante Wold, Kim Esbensen, and Paul Geladi. Proceedings of the multivariate statistical workshop for geologists and geochemists principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1):37 – 52, 1987.

[42] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459, 2010.

[43] Eric Hayman, Barbara Caputo, Mario Fritz, and Jan-Olof Eklundh. On the significance of real-world conditions for material classification. In *Computer Vision-ECCV 2004*, pages 253–266. Springer, 2004.

[44] Jin Xie, Lei Zhang, Jane You, and Simon Shiu. Effective texture classification by texton encoding induced statistical features. *PATTERN RECOGNITION*, 48(2):447–457, FEB 2015.

[45] Rakesh Mehta and Karen Egiazarian. Texture Classification Using Dense Micro-block Difference (DMD). In Cremers, D and Reid, I and Saito, H and Yang, MH, editor, *COMPUTER VISION - ACCV 2014, PT II*, volume 9004 of *Lecture Notes in Computer Science*, pages 643–658. Singapore Tourism Board; Omron; Nvidia; Garena; Samsung; Adobe; ViSenze; Lee Fdn; Morpx; Microsoft Res; NICTA, 2015. 12th Asian Conference on Computer Vision (ACCV), Singapore, SINGAPORE, NOV 01-05, 2014.

[46] Rouzbeh Maani, Sanjay Kalra, and Yee-Hong Yang. Noise robust rotation invariant features for texture classification. *PATTERN RECOGNITION*, 46(8):2103–2116, AUG 2013.

[47] Mario Fritz, Eric Hayman, Barbara Caputo, and Jan-Olof Eklundh. The kth-tips database, 2004.

[48] Rafael C Gonzalez and Richard E Woods. Digital image processing 3rd edition, 2007.

Appendix A: Haralick Features equations

All 13 original Haralick Features considered on this work can be calculated by the following equations:

1. Energy (Angular Second Moment), where N_g is the number of gray-levels:

$$f_1 = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j)^2 \tag{17}$$

2. Contrast:

$$f_2 = \sum_{n=0}^{N_g-1} n^2 \left(\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j) \right)_{|i-j|=n} \tag{18}$$

3. Correlation, where μ_x, μ_y, σ_x and σy are the mean and standard deviation for p_x and p_y, respectively:

$$f_3 = \left(\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} (ij) p(i,j) - \mu_x \mu_y \right) \left(\frac{1}{\sigma_x \sigma_y} \right) \tag{19}$$

where p_x and p_y are defined by:

$$p_x(i) = \sum_{j=1}^{N_g} p(i,j) \tag{20}$$

$$p_y(j) = \sum_{i=1}^{N_g} p(i,j) \tag{21}$$

4. Sum of Squares: Variance, where μ is the average value of the probability function $p(i,j)$:

$$f_4 = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} (i-\mu)^2 p(i,j) \tag{22}$$

5. Local Homogeneity (Inverse Difference Moment):

$$f_5 = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} \frac{1}{1+(i-j)^2} p(i,j) \tag{23}$$

6. Sum Average:

$$f_6 = \sum_{i=2}^{2N_g} [i \cdot p_{x+y}(i)] \tag{24}$$

where p_{x+y} is defined by:

$$p_{x+y}(k) = \left(\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j) \right)_{i+j=k} \tag{25}$$

7. Sum Variance:

$$f_7 = \sum_{i=2}^{2N_g} (i - f_8)^2 \, p_{x+y}(i) \tag{26}$$

8. Sum Boltzmann Entropy (Sum Entropy):

$$f_8 = \sum_{i=2}^{2N_g} p_{x+y}(i) \cdot \log p_{x+y}(i) \tag{27}$$

9. Boltzmann Entropy (Entropy):

$$f_9 = -\left(\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j) \cdot \log p(i,j) \right) \tag{28}$$

10. Difference Variance:

$$f_{10} = Var(p_{x-y}) \tag{29}$$

where p_{x-y} is given by:

$$p_{x-y}(k) = \left(\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j) \right)_{|i-j|=k} \tag{30}$$

11. Difference Entropy:

$$f_{11} = -\sum_{i=0}^{N_g-1} p_{x-y}(i) \cdot \log[p_{x-y}(i)] \tag{31}$$

12. Information of Correlation 1:

$$f_{12} = \left(\frac{HXY - HXY1}{\max HX, HY} \right) \tag{32}$$

where HXY, HXY1, HX and HY are given by:

$$HXY = f_9 \tag{33}$$

$$HXY1 = -\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j) \cdot \log p_x(i) \cdot p_y(j) \tag{34}$$

$$HX = \sum_{i=1}^{N_g} p_x(i) \cdot \log p_x(i) \tag{35}$$

$$HY = \sum_{j=1}^{N_g} p_y(j) \cdot \log p_y(j) \tag{36}$$

13. Information of Correlation 2:

$$f_{13} = \sqrt{1 - e^{-2 \cdot (HXY2 - HXY)}} \tag{37}$$

where HXY2 is given by:

$$HXY2 = -\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} [p_x(i) \cdot p_y(j)] \cdot \log p_x(i) \cdot p_y(j) \tag{38}$$

Appendix B: Haralick-based features proposed by M. Linek et al.

All 3 Haralick-based Features proposed by M. Linek et al. which were considered on this work can be calculated by the following equations:

14. Max Probability (Mode):

$$f_{14} = \max p(i,j) \tag{39}$$

15. Cluster Shade, where N_g is the number of gray-levels:

$$f_{15} = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} (i - M_x + j - M_y)^3 \cdot p(i,j) \tag{40}$$

where M_x and M_y are given by:

$$M_x = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} i \cdot p(i,j) \tag{41}$$

$$M_y = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} j \cdot p(i,j) \tag{42}$$

16. Cluster Prominence:

$$f_{16} = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} (i - M_x + j - M_y)^4 \cdot p(i,j) \tag{43}$$

Appendix C: Extra Features.

The Tsallis Entropy for a given probability distribution function and a q-value can be calculated as follows. Some values of q may maximize little differences between pictures, in comparison to Boltzmann Entropy. Therefore, for this value of q, differences in Entropy for different images that for Boltzmann Entropy would not be detectable, may be magnified, leading to a more precise classification. In this work the value that maximizes the Entropy differences between images (for the KTH-TIPS dataset) has been found to be $q = 0.1$:

17. Tsallis Entropy:

$$f_{17} = -\sum_{i=1}^{N_g}\sum_{j=1}^{N_g}\frac{1}{q-1}\cdot p(i,j)^q \tag{44}$$