JDSSV

Journal of Data Science, Statistics, and Visualisation

September 2026, Volume VI, Issue VI. *doi: 10.52933/jdssv.v6i6.178*

Support Vector Machine-Based Visualization for a Multi-Instance Defect Classification Methodology in the Dissimilarity Space

Eduardo Villegas-Jaramillo
Universidad Nacional de Colombia

Jorge E. Hurtado
Universidad Nacional de Colombia

Mauricio Orozco-Alzate
Universidad Nacional de Colombia

Abstract

Automated visual inspection (AVI) systems for defect classification on the surface of objects involve sensing, preprocessing, representation, classification and post-processing. The first two stages produce either raw or preprocessed images, whereas the last one provides visualizations and actions based on the classification results. Regarding post-processing, this paper introduces a novel visualization technique—based on two statistically independent variables of a support vector machine (SVM), namely, the norm of the vector mapped by the SVM and the cosine of the angle between it and the polar vector of the separating hyperplane—which provides a hyperbolic visualization for an intuitive understanding of the defect classification results and an easy comparison against other classifiers. For the previous stages of the AVI system, we adopted a dissimilarity-based methodology, relying on multi-instance learning (MIL) and dissimilarity spaces, to deal with inexactly labeled examples and eliminate the need for explicit feature extraction. The proposed SVM-based visualization and the multi-instance methodology were evaluated with synthetic and real-world datasets, confirming the convenience of visually judging classification results, for an intuitive and interactive interpretation of the class label assignments, instead of only looking at small numerical

differences in classification performances which, for very small orders of magnitude, may turn difficult to understand. In addition, high classification accuracies (above 0.97) were obtained under several grid sizes for MIL representation via block decomposition and when using different dissimilarity measures for all the considered datasets. The approach is scalable and generalizable across different datasets, making it a robust alternative for AVI tasks.

Keywords: Automated visual inspection, defect classification, dissimilarity space, multi-instance learning, support vector machines, visualization.

---

1. Introduction

Many different automated visual inspection (AVI) systems, aimed at supporting quality assessment in industrial production processes, have been proposed over the years. Among them, those for detecting defects in the surface of objects (Zheng et al. 2021) stand out, either planar as in flat steel products (Luo et al. 2020) or convex as in fruits and other round shaped groceries (Soltani Firouz and Sardari 2022). Machine learning in these systems allow to support repetitive and labor-intensive tasks that were traditionally only performed by humans without the assistance of visual inspection technologies. In such a way, the quality of the products is better guaranteed in the sense that AVI systems do not suffer from human factors such as boredom, fatigue, inexperience or variability due to rotation of personnel. However, these AVI systems are typically only based on numerical outputs, lacking a visualization that provides a graphical feedback for an intuitive interpretation of the classification results.

Most AVI systems are composed by the following stages: (i) sensing, (ii) preprocessing, (iii) representation, (iv) classification and v) post-processing. The first stage corresponds to the acquisition of raw data: typically an image of the product. In the second stage, which is optional, the raw images are subjected to operations such as filtering and segmentation, as well as transformed into spectra, contours or histograms. In the third stage, descriptive attributes—often called features—are measured from the preprocessed images, to be used afterwards as entries of a vector that represents the image in a vector space (also known as feature space). Instead of measuring features to each raw or preprocessed image, it has also been proposed to compare each one directly against a set of other representative images by using an appropriate dissimilarity measure; in that way, each image would be represented as a vector of dissimilarities that, in turn, can be understood as a point in a vector space: the so-called dissimilarity space (Duin and Pekalska 2012). The fourth stage involves the design and implementation of a decision rule that assigns a class label (e.g., Defective or Non-defective) to the vectorial representation of a previously unseen image, ensuring that it was not used during the design of the decision rule. Finally, an interpretation of the classification results might be carried out; for instance providing a detailed report of the confidences for the assigned class labels or a visualization for an intuitive understanding by the industrial practitioners. Actions such as the robotic acceptance/rejection of the inspected object are also often included in this last stage.

The decision rule—mentioned above in the fourth stage of an AVI system—is learned from the data and, therefore, requires a set of labeled examples (i.e., a training set $\mathcal{T}$). In the case of images, the available class labels in the training set may be only provided for the images as a whole or, alternatively, being accompanied with metadata indicating the specific location of the defects; for instance with coordinates of bounding boxes or binary segmentation masks. The latter scenario is said to be strongly supervised and, in contrast, the former one is known as a classification problem with inexact supervision, which is a special case of the so-called weakly supervised learning (Zhou 2018). The inexactly supervised problem has been successfully addressed by using an approach called multi-instance learning (MIL) (Herrera et al. 2016), which has also been applied to build AVI solutions, e.g., in (Mera et al. 2016) and (Ge et al. 2020).

According to Jha and Babiceanu (2023), two categories of AVI solutions can be distinguished, namely, traditional systems and modern ones. The first category refers to systems whose representation stage requires an explicit extraction of features that are typically based on geometric properties, statistical attributes or other descriptors for each image, such that a classification rule can be trained and applied in the feature space. The most common classification rules in traditional AVI systems include simple but competitive ones based either on distances—e.g., the well-known nearest neighbor rule (1-NN)—or on probabilities, such as the Bayesian classifiers; alternatively, more sophisticated classifiers are also available; for instance, support vector machines (SVM) and artificial neural networks. On the other hand, the second category of AVI systems comprise the relatively recent developments in deep learning (DL) which, in contrast with the traditional solutions, do not require an explicit feature extraction stage; in other words, the deep neural networks implicitly perform feature extraction within their many hidden layers, whereas the decision for the class label assignment is taken in the last layer. However, there has been a growing trend in parsimonious machine learning (Divasón et al. 2023) advocating to reconsider simpler (traditional) solutions, such that the Occam's razor principle is observed.

Lastly, consider the fifth stage of an AVI system (post-processing). An important aspect in this stage is applying a visualization tool aimed at exploring both the general structure of the dataset and the particular location of each test point (taken from a test set $\mathcal{S}$) when it is seen within the cloud of training examples. Indeed, classification results are more easily understood by looking at visualizations than by examining small differences in numerical results such as, for instance, posterior probabilities whose values might only be different from each other in very small orders of magnitude. For this purpose, the most popular visualization tools range from classic well-known techniques—e.g., Principal Component Analysis (PCA) and Multidimensional Scaling (MDS)—to other modern ones as Locally Linear Embedding (LLE) and t-distributed Stochastic Neighbor Embedding (t-SNE). In spite of the proven helpfulness of these techniques, they are not free from limitations and challenges, including linear restrictions in the classic techniques and the need for careful parameter optimization for the modern methods. Therefore, a simple but powerful visualization technique is desired, hopefully based on information that is already available after applying the classification rule and without requiring additional complicated computations, but just simple arithmetic and trigonometric operations. This is precisely the advantage of our proposed visualization method.

According to the five stages of an AVI system as they were explained above, as well as considering the trend of parsimonious machine learning and the need for a simple but powerful visualization of the classification results, we revisited three successful approaches from the pre-DL era—namely, MIL, dissimilarity spaces and SVMs—and propose a novel visualization based on the coordinate plane (*norm*, *cosine of the angle*) of the SVM-based classification function, which operates as the final stage of a dissimilarity-based and multi-instance methodology for defect classification in object surfaces and assumes an inexactly supervised scenario. In the methodology, neither geometric nor deep features are extracted to represent the images or their parts, but they are just decomposed into blocks that, in turn, are preprocessed as histograms to be compared using simple dissimilarity measures under a MIL approach. Each image is then represented as a vector in a dissimilarity space, in which an SVM is used as classifier and provides the above-mentioned coordinates for visualizing the classification results.

The remaining part of this paper is organized as follows. The essential concepts of MIL, dissimilarities between bags, the dissimilarity space, and SVMs are presented in Sect. 2. Afterwards, the dissimilarity-based and multi-instance methodology is explained in Sect. 3, including a description of the key contribution on the SVM-based visualization of the classification process. Results using one synthetic and two real-world datasets are shown and discussed in Sect. 4. Our concluding remarks are provided in Sect. 5. Finally, a list of the symbols and notation, as well as exemplar images of the datasets and the block decomposition procedure for MIL are presented in Appendices A and B, respectively.

2. Methods

2.1. Multi-instance learning

MIL is an inexactly supervised learning approach that differs from the traditional supervised classification paradigm in the nature of the learning examples. In the traditional paradigm, each example X is represented by a single fixed-length feature vector $\boldsymbol{x}$ whereas, in MIL, each X is represented by a set $\mathcal{X}$ (termed as a bag) of feature vectors (called instances). In other words, each example (a bag) contains one or more instances, thereby the name multi-instance learning. Class labels in MIL are provided for the bags, but not for the individual instances. Furthermore, not all instances are necessarily relevant and, consequently, there may be instances within a bag that do not provide any information about its class or that are even more related to other bag classes than to the one they belong, potentially leading to confusion (Amores 2013).

In image-based recognition, as it is the typical case in AVI systems, a bag represents an entire image; whereas the instances represent parts of it, for example regions of the image found by a segmentation algorithm or simply blocks resulting from a decomposition of the image according to the application of a regular division grid (Waqas et al. 2024). If all the images in a dataset have the same size, the strategy of block decomposition generates the same number n of instances per bag; so, in that case $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$.

The most common assumption in MIL is that a bag is said to be positive if at least one of its instances is positive and, conversely, the bag is labelled as negative if none of its instances are positive (Foulds and Frank 2010). For MIL-based AVI systems, positive means *defective* and negative means *non-defective*. Several specialized approaches have been proposed to solve the MIL problem (Herrera et al. 2016), including an approach to compute dissimilarities between bags (Cheplygina et al. 2015), either for finding the closest training bag to a given test bag (that is, applying the 1-NN rule between bags) or for building a dissimilarity space where any other classifier can be applied. The essentials of both dissimilarities between bags and the dissimilarity space are given below in the subsequent sections.

2.2. Dissimilarities between bags

Let us consider two bags ($\mathcal{X}$ and $\mathcal{Y}$) and an instance-level distance d, such as the Chernoff or the Jeffrey–Matusita distances which are typical choices when the instances to be compared are histograms. Moreover, for the sake of simplicity, let us also consider that both bags contain n instances. Since bags can be conceptually understood as sets of points, any distance measure between sets can be used to estimate the dissimilarity between bags. According to Cheplygina et al. (2015), the Hausdorff distance (also known as the maxmin distance) and its variations can be applied. Their formulations are:

$$d_{\text{maxmin}}(\mathcal{X},\mathcal{Y}) = \max_{1\leq i\leq n} \min_{1\leq j\leq n} d(\boldsymbol{x}_i,\boldsymbol{y}_j), \quad d_{\text{minmin}}(\mathcal{X},\mathcal{Y}) = \min_{1\leq i\leq n} \min_{1\leq j\leq n} d(\boldsymbol{x}_i,\boldsymbol{y}_j),$$
$$d_{\text{meanmin}}(\mathcal{X},\mathcal{Y}) = \frac{1}{n}\sum_{i=1}^{n} \min_{1\leq j\leq n} d(\boldsymbol{x}_i,\boldsymbol{y}_j), \quad d_{\text{meanmean}}(\mathcal{X},\mathcal{Y}) = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} d(\boldsymbol{x}_i,\boldsymbol{y}_j).$$

Other possibilities to compute the dissimilarities between bags include distances between distributions, such as the Mahalanobis distance, the earth movers distance and the Cauchy–Schwarz divergence (Tax et al. 2011; Cheplygina et al. 2015). Nonetheless, it has been found that the original formulation of the Hausdorff distance is a good dissimilarity measure in most cases. Once the dissimilarities between bags are computed, a classification decision can be made either by simply applying the 1-NN rule or building a dissimilarity space.

2.3. The dissimilarity space

In pattern recognition tasks, an object X is traditionally represented as a feature vector $\boldsymbol{x}$, whose dimension corresponds to the number of features that are measured to the object. Therefore, in a linear algebra sense, $\boldsymbol{x}$ lies in a vector space: the feature space. Since features are measured to each object independently from the others, the feature representation is said to be absolute. However, it is sometimes not clear which features are the relevant ones for discrimination purposes, so a direct comparison between the raw objects might be easier or even better than extracting features. Such a direct comparison can be carried out by using an appropriate measure between pairs of objects, allowing to quantify how dissimilar they are. In that case, the traditional way for taking a decision was using the 1-NN rule; i.e., assigning to an object from the test set $\mathcal{S}$ the class label of its closest (less dissimilar) example from the training set $\mathcal{T}$.

As an alternative, Duin and Pekalska (2012) proposed the so-called dissimilarity space: a vector space in which the dimensions do not correspond to features but to dissimilarities from the objects to a number of selected examples known as representation objects. In contrast with the feature-based representation, the dissimilarity-based one is said to be relative. More formally, let us consider a set $\mathcal{R}$ of representation objects (called the representation set) and a dissimilarity measure d. An object X in the dissimilarity space is then represented as a vector $\boldsymbol{x}_{\text{diss}} = [d(X, R_1), \ldots, d(X, R_q)]^\top$ where q is the number of representation objects $R_i \in \mathcal{R}$; moreover, $\mathcal{R} \subseteq \mathcal{T}$. In other words, given the sets $\mathcal{T}$ and $\mathcal{R}$, a dissimilarity representation is composed by a data-dependent mapping $D(\cdot, \mathcal{R}) : \mathcal{T} \to \mathbb{R}^q$, i.e., a mapping from $\mathcal{T}$ to the dissimilarity space. Any traditional classifier can be trained in that space and, afterwards, test objects from $\mathcal{S}$ can also be mapped there to be classified with the trained rule.

An approach that combines MIL and dissimilarities was presented in (Cheplygina et al. 2015), where each bag is represented by a vector of its dissimilarities with respect to other bags in a training set $\mathcal{T}$, such that the standard MIL problem is cast into a classification problem in a dissimilarity space. In spite that any classifier can be used in the dissimilarity space, these authors found that, on average, the SVM was the best performing one. For this reason, as well as according to the motivations explained in Sect. 1, we restricted ourselves to use the SVM for both classification and visualization. A brief description on the essential concepts of the SVM is given below.

2.4. Support vector machines

The SVMs have become one of the most popular and effective tools for classification tasks in pattern recognition (Du et al. 2024). They are based on the idea of finding a hyperplane that best separates the data into different classes. For linear separable classes, a hyperplane classification function reads

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b,$$

where $\boldsymbol{w}$ is the hyperplane vector, $\boldsymbol{x}$ the sample vector of dimensionality q (since, in our case, we will consider a vector in the dissimilarity space with $\mathcal{R} = \mathcal{T}$) and b the hyperplane offset. SVMs are based on a nonlinear projection of $\boldsymbol{x}$ onto a so-called feature space of dimensionality $Q \gg q$ by means of the mapping $\boldsymbol{x} \to \phi(\boldsymbol{x})$:

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \phi(\boldsymbol{x}) \rangle + b.$$

On this basis it is possible to build a kernel functional

$$K(\boldsymbol{x}, \boldsymbol{y}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle, \tag{1}$$

for classification, in which the mapping $\phi(\cdot)$ needs not be explicitly known, as in the polynomial Kernel of order m

$$K(\boldsymbol{x}, \boldsymbol{y}) = (\langle \boldsymbol{x}, \boldsymbol{y} \rangle + \theta)^m,$$

where θ is a parameter. As it is well known, the SVM classification function can be put in the form

$$f(\boldsymbol{x}) = \sum_{i=1}^{s} \alpha_i c_i K(\boldsymbol{x}, \boldsymbol{x}_i) + b,$$

where s is the number of support vectors, $\alpha_i > 0$ their corresponding Lagrange multipliers and c_i their class labels (equal to ± 1).

It is also possible to use other kernels such as the linear kernel, the radial basis function (rbf) kernel and the multi-layer perceptron kernel, which are among the most common ones. Further details about the latter are given in Sect. 3.2.

3. Methodology and Proposed Visualization

Our proposed SVM-based visualization corresponds to the last stage in a methodology that relies on MIL with dissimilarities; see Fig. 1. Inspired by the proposal from (Cheplygina et al. 2015), this methodology allows to cast the MIL problem into a conventional MIL classification problem, but in the dissimilarity space. However, in contrast with the proposal by these authors who have already explored MIL in the dissimilarity space, the methodology does not require a feature extraction in our case. The justification for having a completely dissimilarity-based approach is that it is more generalist or, in other words, it depends less on prior knowledge about which features to extract. Once the classification problem is solved by making use of an SVM, we take advantage of the previously computed information to present the results in a graphical form using a novel strategy based on SVMs allowing us to understand the results easily. This is opposed to the difficulty of observing the small differences among bare numerical results, such as posterior probabilities, whose results might only differ from each other by very small orders of magnitude.

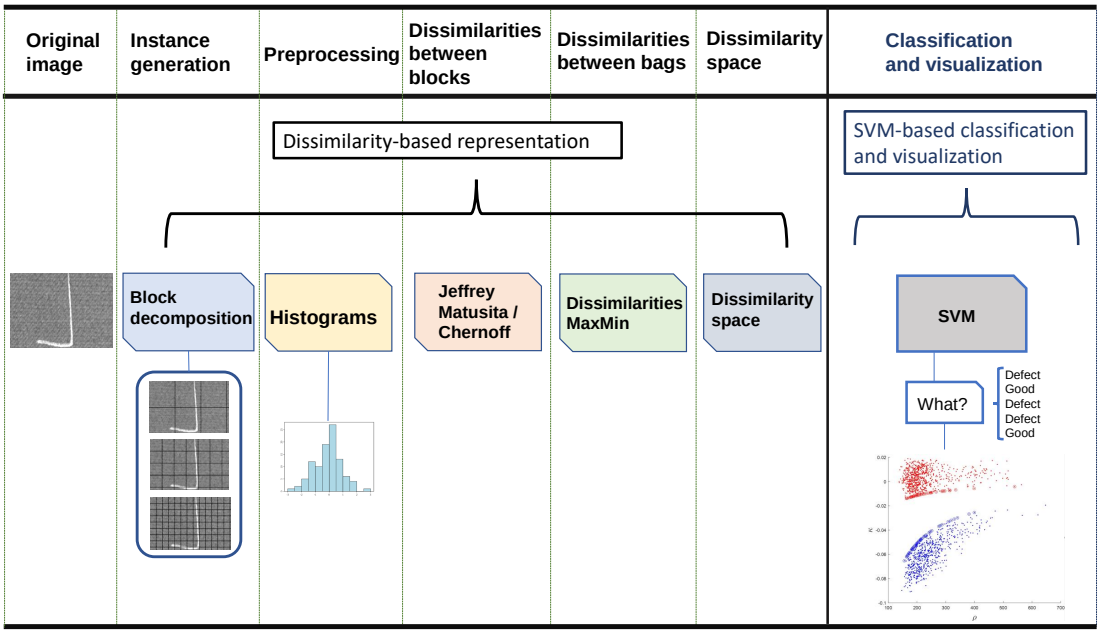


Figure 1: Schematic workflow diagram of the dissimilarity-based and multi-instance methodology, exhibiting the proposed SVM-based visualization as its last stage.

The methodology is dissimilarity-based because the instances (blocks) of the bags (images) will neither be represented as low-level feature vectors nor with deep features,

but will result from the direct comparison between the images using a measure that estimates how different two images or sub-images are based on their pixel histograms; that is, a dissimilarity measure that calculates the difference from simple preprocessed data (histograms) of the image blocks.

The diagram showing the proposed methodology can be seen in Fig. 1, which is decomposed into two parts: (a) the dissimilarity-based representation, and (b) the SVM-based classification and visualization.

3.1. Dissimilarity-based representation

Let X and Y be two arbitrary images of size $p \times l$. They can be either compared as entire images using any distance measure between matrices, or decomposed into blocks in order to follow a MIL approach. Deriving a dissimilarity and MIL-based representation for comparing X and Y involves the five stages explained below.

Instance generation. Given that image comparison can be done on the entire image or on blocks of the image, it is necessary to perform a block decomposition by dividing an image into a specific set of sub-images, according to a block size. Suppose that the image X has a size of $p \times l$ and that the desired block size for its decomposition is $u \times v$. The first thing we need to verify is that $u \leq p$ and $v \leq l$ because, if this is not met, the block would be wider or taller than the image (which is not permissible). Both conditions must be met; otherwise, the block would not fit within the image. The particular case $u = p$, $v = l$ corresponds to considering the entire image as a single block or, in other words, not decomposing into blocks but considering the entire image as a whole.

In block decomposition, different grid sizes are used so that, with a simple strategy, the resulting blocks can be directly compared based on either the raw sub-images, using the pixels that compose them, or on a simple preprocessing (for instance their pixel histograms). For this reason, regardless of the dimensions of the datasets used (see Sect. 4.1), grids half the original size (256×256 and 150×150) and finally a grid a quarter of the original size (128×128 and 75×75) were selected, covering different possibilities for defect classification and localization, resulting in 1, 4, and 16 blocks, respectively. Figure 10 in Appendix B shows examples of block decomposition for an image belonging to the DAGM C3 dataset.

Preprocessing. Having divided the images into blocks, the dissimilarity matrix between all the images must be calculated; for that purpose, computing the respective histograms from grayscale raw intensity values is required. Afterwards, the comparison of each image with respect to all the others is carried out by comparing their histograms. The images can be complete (without block decomposition) or built by its sub-images resulting from the block decomposition. As a result of this stage, images X and Y are structurally represented as bags of histograms: $\mathcal{X} = \{\boldsymbol{h}_X^{(1)}, \ldots, \boldsymbol{h}_X^{(n)}\}$ and $\mathcal{Y} = \{\boldsymbol{h}_Y^{(1)}, \ldots, \boldsymbol{h}_Y^{(n)}\}$, respectively.

Dissimilarities between histograms. Using image comparison functions such as the Chernoff and Jeffrey–Matusita dissimilarities (Mitchell 2010), which are based on the respective histograms, the resulting dissimilarity between each pair of images X and Y is calculated, as follows:

$$d_{\text{Chernoff}}\left(\widehat{\boldsymbol{h}_X^{(i)}}, \widehat{\boldsymbol{h}_Y^{(j)}}\right) = -\log\left(\max\left[\epsilon, \sum_{k=1}^{256}\left\{\widehat{\boldsymbol{h}_X^{(i)}}(k)\right\}^{\lambda}\left\{\widehat{\boldsymbol{h}_Y^{(j)}}(k)\right\}^{1-\lambda}\right]\right),$$

$$d_{\text{Jeffrey}--\text{Matusita}}\left(\widehat{\boldsymbol{h}_X^{(i)}}, \widehat{\boldsymbol{h}_Y^{(j)}}\right) = \sqrt{\sum_{k=1}^{256}\left\{\sqrt{\widehat{\boldsymbol{h}_X^{(i)}}(k)} - \sqrt{\widehat{\boldsymbol{h}_Y^{(j)}}(k)}\right\}^2},$$

where $\widehat{\boldsymbol{h}_X^{(i)}}(k)$ and $\widehat{\boldsymbol{h}_Y^{(j)}}(k)$ denote the k-th element of the normalized histograms, $\lambda \in [0, 1]$ is optimized and $\epsilon > 0$ is an arbitrarily small number that prevents numerical problems. Notice that normalizing the histograms is required since both d_{Chernoff} and $d_{\text{Jeffrey}--\text{Matusita}}$ are defined as dissimilarity measures between probability distributions.

Dissimilarities between bags. It is important to note that if the comparison of two images is performed under block decomposition, the Hausdorff distance formulation allows to obtain the overall dissimilarity between the two images, which is achieved by the max and min operations. That distance, as adapted for the case of bags of normalized histograms, takes the form:

$$d_{\text{maxmin}}(\mathcal{X}, \mathcal{Y}) = \max_{1\le i\le n}\min_{1\le j\le n} d\left(\widehat{\boldsymbol{h}_X^{(i)}}, \widehat{\boldsymbol{h}_Y^{(j)}}\right).$$

Dissimilarity space. The dissimilarity for each pair of images in the dataset is calculated, whether for complete images or for sub-images resulting from block decomposition using specific block sizes (such as 512×512, 256×256, or 128×128). The result is a dissimilarity matrix, where the value of the dissimilarity measure between two images X and Y corresponds to an entry in the matrix. If $\mathcal{R} = \mathcal{T}$, the resulting dissimilarity matrix is square. However, the dimensionality of the representation space can be reduced in order to find a balance between space dimension and performance. When a new data point arrives, it is represented as a vector of its dissimilarities against the selected objects from $\mathcal{R}$.

3.2. SVM-based classification and visualization

Once the training data is located in the dissimilarity space defined by the representation data, an SVM classifier is trained in this space, which will serve to classify test objects, as well as to perform the visualization procedure. Visualization of the SVM classification becomes possible if use is made of a kernel that can be expressed as a dot product of the nonlinear mapping vectors $\phi(\cdot)$ evaluated at different points, as in Eq. (1). Such is the case of the Polynomial Kernel

$$K(\boldsymbol{x}, \boldsymbol{y}) = (\langle \boldsymbol{x}, \boldsymbol{y}\rangle + \theta)^m,$$

where m, an integer, is the order of the kernel. In such a case, the classification function (1) has a weight vector $\boldsymbol{w}$ of the form

$$\boldsymbol{w} = \sum_{i=1}^{s} \alpha_i c_i \phi(\boldsymbol{x}_i)$$

and can be expressed as

$$f(\boldsymbol{x}) = |\boldsymbol{w}||\phi(\boldsymbol{x})| \cos\beta + b,$$

where β is the angle between vectors $\boldsymbol{w}$ and $\phi(\boldsymbol{x})$. With the notation $\rho = |\phi(\boldsymbol{x})|$, $\kappa = \cos\beta$, the classifier can be represented as a function of two variables ρ and κ in the form

$$f(\rho, \kappa) = |\boldsymbol{w}|\rho\kappa + b. \tag{2}$$

The relevance of this representation lies in the fact that variables ρ and κ have the strong property of being statistically independent, as the norm of vector $|\phi(\boldsymbol{x})|$ does not depend on the angle it forms with the polar vector $\boldsymbol{w}$, nor vice-versa. This fact makes them ideal for building a visualization tool of the SVM classification (as it will be shown in Sect. 4.3). Moreover, Eq. (2) indicates that the (ρ, κ) representation emerges naturally from the SVM classification function, as explained next.

For a given constant value f of the classification function, such as those defining the SVM margin $f(\rho, \kappa) = \pm 1$, the following hyperbolic function ensues:

$$\kappa = \frac{C}{\rho}, \tag{3}$$

with

$$C = \frac{f - b}{|\boldsymbol{w}|}. \tag{4}$$

This means that the support vector margins obey the hyperbolic law given by Eq. (4) with $C = (\pm 1 - b)/|\boldsymbol{w}|$. Generally speaking, the contours corresponding to constant values of the classification function (e.g., $f = \pm 2, \pm 3, ...$) are also hyperbolas that allow the visualization of strips of samples located between their values. On the basis of the known state of the training samples, labels such as damaged, safe, unsafe, ill, very ill, feasible etc., depending on the case, can then be assigned to the strips in order to make sense of the classification results in qualitative terms.

It must be noted that the proposed visualization procedure can be used for comparing the performance of different classifiers in a visual form. To this end, one simply uses the same coordinates ρ and κ calculated with the polynomial kernel, but the test point assumes the label and color given by each of the classifiers under examination. One can then obtain a representation of the right or wrong labels reported by each classifier and visually select the most adequate for practical purposes. Moreover, this SVM-based visual comparison provides not just a global sense of the differences between classifiers, but also a context for their assigned labels since test points are visualized along with the cloud of training examples.

4. Results and Discussion

4.1. Datasets and evaluation setup

Three datasets were selected in order to classify them into defective and non-defective categories, as well as to perform the respective graphical visualization of the classification. The first dataset, which is made up of synthetic images, originates from the German Society for Pattern Recognition (DAGM). The second dataset contains a collection of peanut candies, whereas the third comprises a set of lemons.

DAGM dataset

The DAGM dataset (Wieler et al. 2007) contains several subsets or classes consisting of synthetic images used to detect defects on textured surfaces. They were originally created for a contest at the 2007 Annual Symposium of the DAGM. Among them, *Class_3*(C3) was selected since it represents a particularly challenging dataset, thereby enabling the evaluation of the developed methodology. This class contains grayscale images composed by artificial backgrounds with a diffuse appearance, with and without defects, where the defects are particularly white blotch-like patterns (see Fig. 7 in Appendix B). The dataset comprises 1,150 images with a resolution of 512×512 pixels; 150 of them have defects whereas the remaining 1,000 images are defect-free.

Candy dataset

This dataset (Villegas-Jaramillo and Orozco-Alzate 2023) is composed by a collection of 400 color images corresponding to 200 peanut candies coated with colored chocolate, with two images per candy. The images in the dataset have a resolution of 512×512 pixels, and are balanced in number: 200 images have defects and the remaining 200 images are defect-free; see Fig. 8 in Appendix B.

Lemon quality dataset

This dataset (Emir 2022) is composed by a collection of 2076 color images of lemons with a resolution of 300×300 pixels, almost balanced so that 951 images have defects and the remaining 1125 images are defect-free; see Fig. 9 in Appendix B.

Evaluation setup

The executions with the three datasets were performed on a DELL server with 24 Intel® Xeon® CPU E5-2643 v3 @ 3.40GHz CPU (48 threads) and 128 GiB RAM. Rocky Linux version 9.5 as an operating system, where Matlab R2024b Update 2 (24.2.0.2773142) 64-bit was used as the support language. For the three datasets, the images are used without any preprocessing or normalization. The images are then decomposed based on the selected grid sizes, generating a list of the resulting sub-images.

4.2. Classification results

The classification of the images is performed by training an SVM with a polynomial kernel (hereafter polySVM), having an order $m = 2$ (this order was chosen as the

first non-linear option, providing a tradeoff between complexity and performance) and using the training images of each dataset, resulting in the coordinates of the support vectors, the weight and the bias, thus allowing the classification of test images for each combination of dataset, partition, and dissimilarity measure. To perform the classification, the same parameters for the polySVM classifier are used for both training and testing images, thereby allowing the calculation of metrics such as accuracy, TP-rate and FP-rate for the different executions.

As explained in Sect. 2.3, the representation set $\mathcal{R}$ can be chosen as a subset of the training set $\mathcal{T}$; however, as indicated in (Pekalska and Duin 2005), one may use the complete dissimilarity representation (that is, $\mathcal{R} = \mathcal{T}$) if that option is computationally tractable. Moreover, Duin et al. (2013) claim that using $\mathcal{R} = \mathcal{T}$ is particularly useful for the case of the SVM classifier. We have adopted that suggestion but, anyway, performed a series of executions for increasing sizes of $\mathcal{R}$ selected at random, which led us to confirm that the best classification results are obtained for the largest size of $\mathcal{R}$. Therefore, we report below the average classification performances of 25 repetitions for each dataset, randomly dividing it into training and test sets, with 70% allocated for training and the remaining 30% for testing and using the entire training set for representation. The largest grid size in each case corresponds to using the entire images without an effective block decomposition. For the other two grid sizes, the overall best results are highlighted in boldface.

A summary of the classification metrics for the evaluated dataset can be seen in Table 1; in addition, the same average results across the different data splits along with their corresponding standard deviations are shown in Appendix C. From this appendix, it can be seen that results are more stable as the grid size decreases; moreover, it is also interesting that standard deviations for the Lemon quality dataset are consistently lower for the Jeffrey–Matusita distance; in contrast, for the Candy dataset, the opposite behavior is observed in most of the cases. Consider firstly the case of the DAGM dataset, for which the 345 test images were classified. It can be highlighted that the lowest accuracy values are obtained when no decomposition of the images is used and that as the size of the grid becomes smaller, the results improve, finding the best ones when the grid size is 128×128 with values of 0.9957 for the Chernoff distance and 0.9951 for the Jeffrey–Matusita distance. Given the imbalance presented by the categories of this dataset, the values of TP-rate and FP-rate gain relevance, finding the best results for a TP-rate of 0.9876 and 0.9851 in Chernoff and Jeffrey–Matusita dissimilarity measures respectively, for a 128×128 grid size. Regarding the results for the FP-rate, values of 0.031 and 0.0033 in the same 128×128 grid size confirm that using grids of progressively smaller sizes improves the results in terms of these metrics.

Let us now examine the classification performances for the Candy dataset, where the 120 test images were classified and, similarly to the case of the DAGM dataset, the lowest accuracies are obtained when no decomposition of the images is used; nonetheless, the results improve when decomposition is applied, with the best results achieved with a grid size of 256×256, yielding values of 0.9847 for the Chernoff distance and 0.9720 for the Jeffrey–Matusita distance.

Finally, consider the classification metrics obtained for the Lemon quality dataset, where the 623 test images were classified. Similarly to the Candy and DAGM datasets, the lowest values are obtained when no decomposition of the images is used, and the

Table 1: Classification metrics for the evaluated datasets organized by grid size and dissimilarity measure (J–M is an abbreviation for Jeffrey–Matusita).

Grid Size	Metric	DAGM		Candy		Lemon	
		Chernoff	J–M	Chernoff	J–M	Chernoff	J–M
Large	TP-rate	0.8175	0.9663	0.8129	0.7553	0.9008	0.9223
	FP-rate	0.0425	0.0628	0.2004	0.2546	0.0905	0.0646
	Accuracy	0.9410	0.93916	0.8040	0.7477	0.9053	0.9293
Medium	TP-rate	0.9333	0.9539	0.9889	0.9684	0.9967	0.9980
	FP-rate	0.0093	0.0099	0.0187	0.0238	0.0017	0.0011
	Accuracy	0.9830	0.9853	**0.9847**	**0.9720**	**0.9976**	**0.9985**
Small	TP-rate	**0.9876**	**0.9851**	0.9797	0.9729	0.9948	0.9975
	FP-rate	**0.0031**	**0.0033**	0.0268	0.0300	0.0040	0.0011
	Accuracy	**0.9957**	**0.9951**	0.9760	0.9710	0.9954	0.9983

results improve when decomposition is applied, with the best results achieved with a grid size of 150×150, yielding values of 0.9976 for the Chernoff distance and 0.9985 for the Jeffrey–Matusita distance.

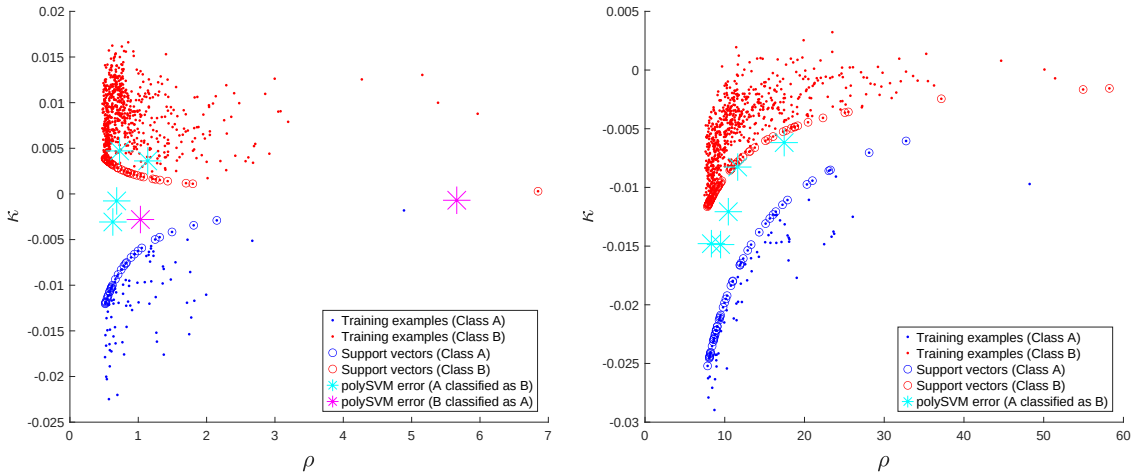
As a general observation, notice that the best results for the DAGM dataset are obtained for the smallest grid size; whereas, the best results for the other two datasets are obtained with a medium-sized grid. This behavior makes sense because the defects in the first dataset are very small (fitting within a cell of the smallest grid) and, in contrast, the defects in the Candy and Lemon datasets may cover a significant part of the objects; see again Figs. 8 and 9 in Appendix B. Another interesting finding is that the choice of dissimilarity measure matters greatly for larger grid sizes, but less for smaller ones; this can be explained by considering that smaller grid sizes imply more blocks and, therefore, the Hausdorff distance will choice the maximum and minimum distances among more values, potentially reducing the importance of the measure that is used at the instance level. Considering that the best classification results, as stated above, were obtained for the smallest and medium grid sizes, visualizations for the largest grid size are omitted in the subsequent section. Moreover, only visualizations corresponding to the last of the 25 repetitions performed for each evaluation are presented.

4.3. Visualization results

The SVM-based visualizations of the classification process for each of the three datasets are presented with the training set divided into its two distinct classes. Class A, representing the defective category, is depicted in blue, whereas Class B, representing the non-defective category, is shown in red. Objects in each class are visualized as points of their respective colors. In addition, the support vectors derived from the polySVM training are highlighted as points encircled by a ring of the corresponding color. The test set and its corresponding position on the graph are also displayed, indicating the class assignment of each test image. Misclassified instances are shown as larger asterisks ($*$), cyan for Class A, and purple for Class B, allowing observation of their proximity to or distance from the hyperplanes defined by the support vectors of each class.

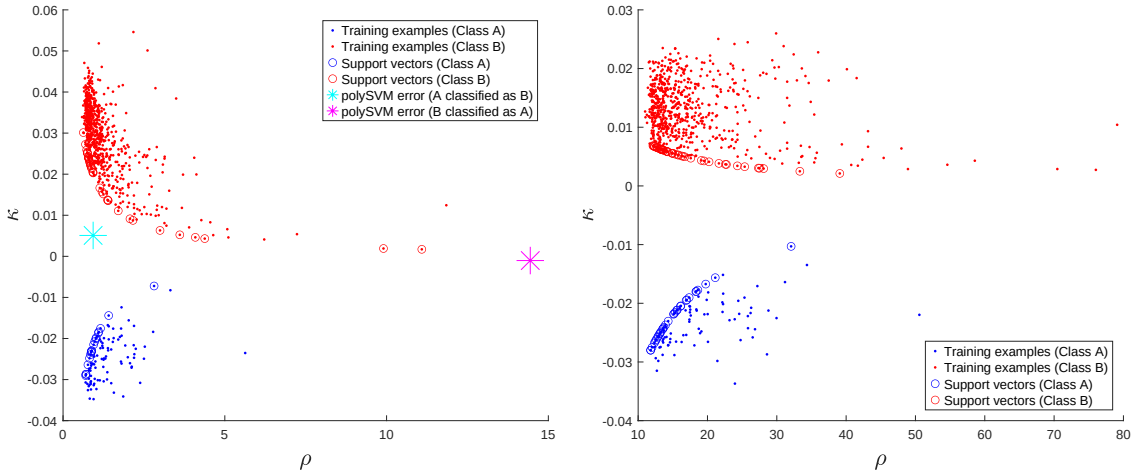
The following are the visualization results of the classification process for each of the three datasets:

Firstly, the visualizations for the DAGM dataset can be seen in Fig. 2, where the first row corresponds to the visualization of the classification results using the Chernoff and Jeffrey–Matusita dissimilarity measures, with a 256×256 grid size, where the Chernoff dissimilarity measure is on the left and the Jeffrey–Matusita dissimilarity measure is on the right. Analogously, visualizations obtained when decomposing the images with a 128×128 grid size are shown in the second row. Notice that there is a significant separation observed in Figs. 2(c) and (d), particularly in (d), where the two sets are clearly separated and there are no misclassified objects. In all panels of Fig. 2, the hyperbolic shape of the line connecting the support vectors, anticipated above by the comment to Eq. (3), can be appreciated.



(a) Image decomposed on a 256×256 grid with Chernoff distance.

(b) Image decomposed on a 256×256 grid with Jeffrey–Matusita distance.

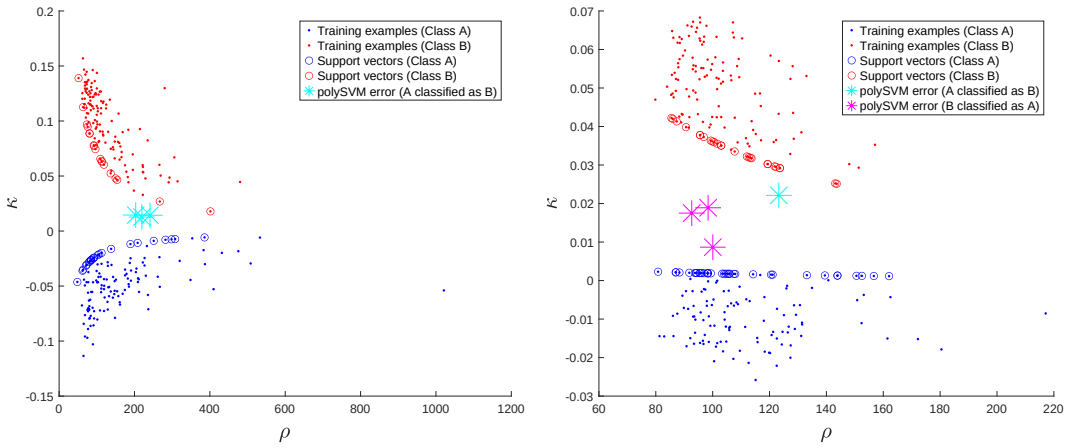


(c) Image decomposed on a 128×128 grid with Chernoff distance.

(d) Image decomposed on a 128×128 grid with Jeffrey–Matusita distance.

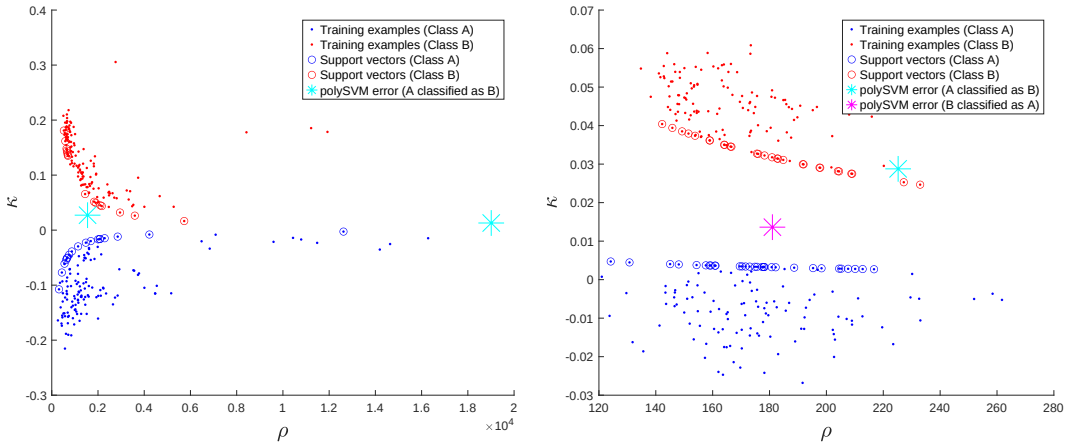
Figure 2: SVM-based visualization of the classification results for the DAGM dataset: the first row shows results using a 256×256 grid size, and the second row employs a 128×128 grid size.

Secondly, the visualization results for the Candy dataset are presented in Fig. 3. The first row shows the classification results obtained with a 256×256 grid size, whereas the second row depicts the results with a 128×128 grid size. The good results when using the smallest grid size and the Jeffrey–Matusita distance are noteworthy (see Fig. 3(d)), where classes are well separated and there are few classification errors; in contrast to additional errors observed in Figs. 3(a) and (b), where the largest partition is applied and, as a result, a slight performance decrease is observed for this dataset.



(a) Image decomposed on a 256×256 grid size with Chernoff distance.

(b) Image decomposed on a 256×256 grid size with Jeffrey–Matusita distance.



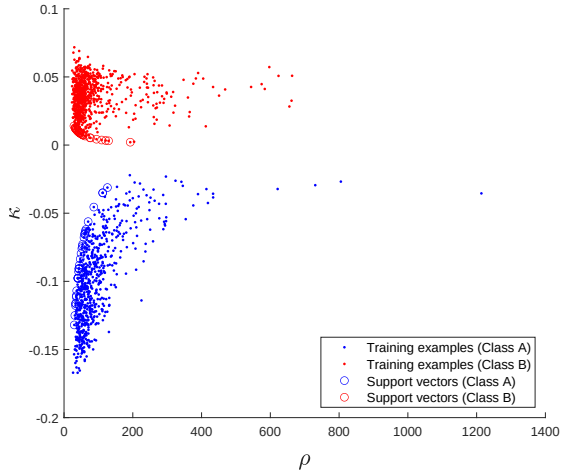
(c) Image decomposed on a 128×128 grid size with Chernoff distance.

(d) Image decomposed on a 128×128 grid size with Jeffrey–Matusita distance.

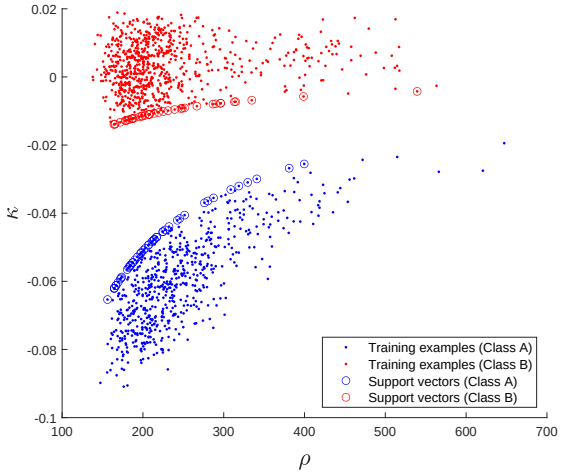
Figure 3: SVM-based visualization of the classification results for the Candy dataset: the first row shows results with a 256×256 grid size, and the second row employs a 128×128 grid size.

Finally, the visualizations for the Lemon quality dataset can be seen in Fig. 4, where in the first row, results obtained when decomposing the images with a 150×150 grid size are presented and, in the second row, those corresponding to a decomposition of the images with a 75×75 grid size. For this dataset, good visualization results are highlighted by using image decomposition with the 75×75 and 150×150 grids and the Jeffrey–Matusita distance (see Figs. 4(d) and (b)), and using the 150×150 grid size with Chernoff distance (see Fig. 4(a)), where classes are well separated, and there are

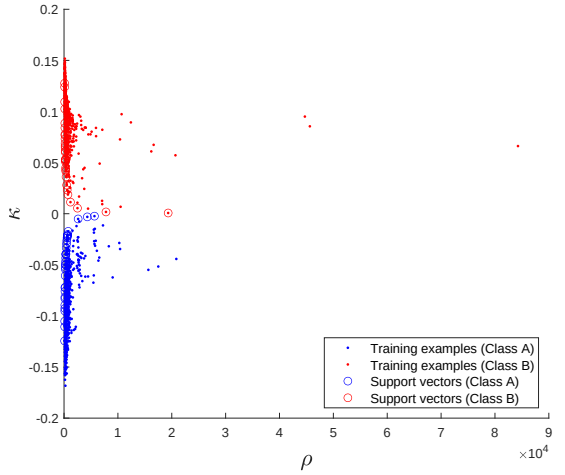
few classification errors.



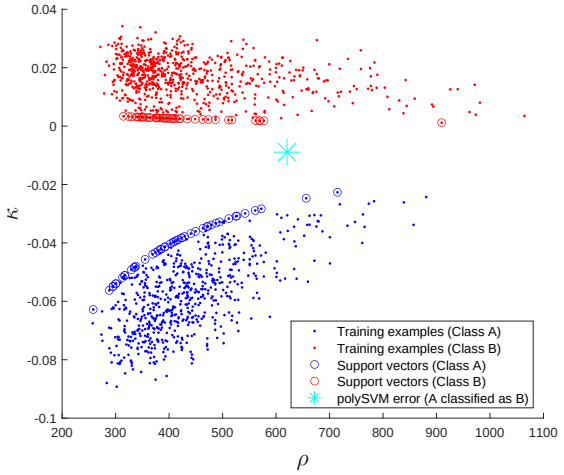
(a) Image decomposed on a 150×150 grid size with Chernoff distance.



(b) Image decomposed on a 150×150 grid size with Jeffrey–Matusita distance.



(c) Image decomposed on a 75×75 grid size with Chernoff distance.

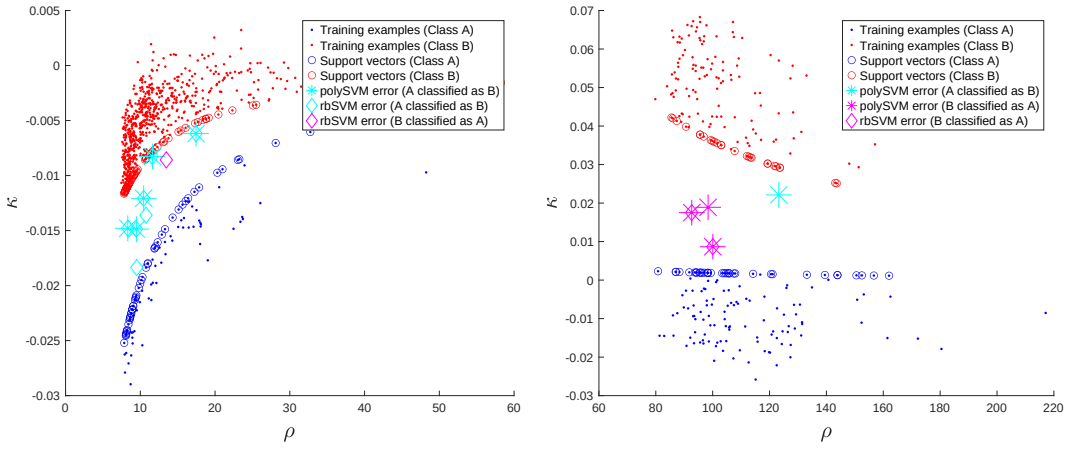


(d) Image decomposed on a 75×75 grid size with Jeffrey–Matusita distance.

Figure 4: SVM-based visualization of the classification results for the Lemon quality dataset: the first row shows results using a 150×150 grid size, and the second row employs a 75×75 grid size.

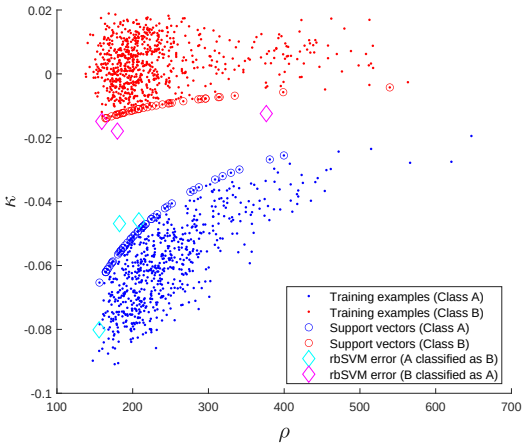
The results for the three datasets show that, in general, visualizations using the Chernoff dissimilarity measure produce concentrated data clouds on the left side of the (ρ, κ) representation, but exhibiting few points on the far part of the coordinate plane. In contrast, the data points in the visualizations using the Jeffrey–Matusita dissimilarity measure better cover the horizontal space of the (ρ, κ) plane and, as a result, the appreciation of the individual cases turns easier in those figures. Taking into account this behavior, below we restrict ourselves to using the Jeffrey–Matusita dissimilarity measure to illustrate the procedure described at the end of Sect. 3.2 for comparing classifiers; moreover, only the intermediate grid sizes were considered, such that the classifiers under comparison make some wrong class label assignments and, therefore, differences in their performances could be appreciated. The classifiers to be compared

were the polySVM and, without loss of generality, another SVM but using an rbf kernel (denoted in the figures below as rbSVM). The rbf kernel implicitly maps the data onto an infinite dimensional space (Olson 2008), allowing a good separability of the data points; as a result, it is typically one of the best performing kernels, particularly when its parameters (gamma and C) are tuned via cross-validation. Therefore, we compared the polySVM against an optimal rbSVM in terms of the best result for a cross-validation of its parameters. Results of the visual comparison are shown in Fig. 5.



(a) DAGM dataset.

(b) Candy dataset.



(c) Lemon quality dataset.

Figure 5: Visualizations in the (ρ, κ) coordinate plane for comparing classification results with two different SVMs (polySVM and rbSVM). The proposed methodology was evaluated using intermediate grid sizes and the Jeffrey–Matusita dissimilarity measure.

From Figs. 5(a) and 5(b), it can be noticed that both SVMs misclassified a number of the same test objects, particularly defective objects (class A) wrongly classified as non-defective (class B) in the case of the DAGM dataset. Conversely, coincident classification errors of non-defective test objects, wrongly assigned as defective, were observed for the Candy dataset. Finally, in Fig. 5(c), it is noteworthy that polySVM did not make classification errors for the Lemon quality dataset, whereas the rbSVM misclassified three objects from each category. The visualization tool can be easily used to inspect objects of interest by performing either an inverse search from the

(ρ, κ) coordinates to the original image files or, alternatively, via an interactive visual interface that locates and displays the corresponding image after clicking on a selected point. Notice that the (ρ, κ) values are unique for each object and, therefore, they allow a bijective identification useful for developing an interactive AVI system.

This capability is illustrated in Fig. 6 by showing the three objects that were misclassified by rbSVM but correctly classified by polySVM, cf. Fig. 5(c). Another advantage of the proposed visualization method is that, in contrast to other techniques such as PCA and MDS, it does not involve a dimensionality reduction. Moreover, the SVM-based visualization is unique in the sense that is not computed in terms of an additional iterative optimization procedure, but it is just based on the same training process of the SVM classifier itself. On the other hand, intrinsic limitations of the proposed method are: i) the restriction to visualization in only two dimensions and, ii) the high time and space complexity demanded for computing the square dissimilarity matrix when scaling to very large datasets.

Figure 6: Examples of the Lemon quality dataset, that were wrongly assigned to the *Defective* category (first three images) or to the *Non-Defective* category (last three images) by the rbSVM, but that were correctly classified by the polySVM.

5. Conclusions

This study presented a novel dissimilarity-based methodology for defect classification and visualization, leveraging the principles of multi-instance learning, dissimilarity spaces, and support vector machines. The proposed methodology eliminates the need for explicit feature extraction by representing images through their dissimilarities to other images, thus addressing challenges associated with feature selection and interpretability in classification tasks. The main contribution to highlight is the proposal of a simple but effective visualization of the classification results, based on the plane (norm, cosine of the angle) of the SVM, which naturally emerges from the mathematical formulation of this powerful classifier.

The evaluation, conducted on synthetic and real-world datasets (DAGM, Candy, and Lemon quality), demonstrated the robustness of the methodology. The following concluding remarks about the results can be highlighted:

High Classification Accuracy: The use of dissimilarity measures such as Chernoff and Jeffrey–Matusita distances, combined with block decomposition, significantly improved classification accuracy. Notably, small to moderate grid decompositions (e.g. 128×128 or 150×150) improved the separability of defective and non-defective classes.

Visualization Capability: The integration of SVM with a novel visualization approach based on a hyperbolic representation (parameters ρ and κ) offers an intuitive and

interpretable way to explore classification results. This visualization effectively highlighted the separation between classes and provided insights into classifier behavior.

Scalability and Generalization: By adopting a dissimilarity-based approach, the methodology proved to be generalizable across datasets with varying characteristics, such as grayscale and color images, and diverse defect patterns.

Future work will focus on extending the methodology to multi-class classification scenarios and exploring its integration with deep learning techniques to balance simplicity with the powerful representation capabilities of neural networks. Further studies are also needed to understand the implications, if any, of the high correlation between dimensions (dissimilarities) related to similar objects on the geometry of the SVM, as well as to characterize any resulting bias on the proposed visualization method.

Acknowledgments

Support from Universidad Nacional de Colombia - Sede Manizales is acknowledged.

Generative AI Statement

The authors declare that AI tools have not been used neither for developing the proposed methodology nor for writing this paper.

References

Amores, J. (2013). Multiple instance classification: Review, taxonomy and comparative study. *Artificial Intelligence*, 201:81–105, DOI: 10.1016/j.artint.2013.06.003.

Cheplygina, V., Tax, D. M. J., and Loog, M. (2015). Multiple instance learning with bag dissimilarities. *Pattern Recognition*, 48(1):264–275, DOI: 10.1016/j.patcog.2014.07.022.

Divasón, J., Ceniceros, J. F., Sanz-Garcia, A., Pernia-Espinoza, A., and Martinez-de-Pison, F. J. (2023). PSO-PARSIMONY: A method for finding parsimonious and accurate machine learning models with particle swarm optimization. Application for predicting force–displacement curves in T-stub steel connections. *Neurocomputing*, 548:126414, DOI: 10.1016/j.neucom.2023.126414.

Du, K.-L., Jiang, B., Lu, J., Hua, J., and Swamy, M. N. S. (2024). Exploring kernel machines and support vector machines: Principles, techniques, and future directions. *Mathematics*, 12(24), ISSN: 2227-7390, DOI: 10.3390/math12243935.

Duin, R. P. W. and Pekalska, E. (2012). The dissimilarity space: Bridging structural and statistical pattern recognition. *Pattern Recognition Letters*, 33(7):826–832, DOI: 10.1016/j.patrec.2011.04.019.

Duin, R. P. W., Pekalska, E., and Loog, M. (2013). Non-Euclidean dissimilarities: Causes, embedding and informativeness. In *Similarity-Based Pattern Analysis and Recognition*, Advances in Computer Vision and Pattern Recognition, pages 13–44. Springer, London, UK, DOI: 10.1007/978-1-4471-5628-4_2.

Emir, Y. (2022). Lemon quality dataset. https://www.kaggle.com/datasets/yusufemir/lemon-quality-dataset.

Foulds, J. and Frank, E. (2010). A review of multi-instance learning assumptions. *The Knowledge Engineering Review*, 25(1):1–25, DOI: 10.1017/s026988890999035x.

Ge, C., Wang, J., Wang, J., Qi, Q., Sun, H., and Liao, J. (2020). Towards automatic visual inspection: A weakly supervised learning method for industrial applicable object detection. *Computers in Industry*, 121:103232, DOI: 10.1016/j.compind.2020.103232.

Herrera, F., Ventura, S., Bello, R., Cornelis, C., Zafra, A., Tarragó, D. S., and Vluymans, S. (2016). *Multiple Instance Learning - Foundations and Algorithms*. Springer, Cham, Switzerland, DOI: 10.1007/978-3-319-47759-6.

Jha, S. B. and Babiceanu, R. F. (2023). Deep CNN-based visual defect detection: Survey of current literature. *Computers in Industry*, 148:103911, DOI: 10.1016/j.compind.2023.103911 .

Luo, Q., Fang, X., Su, J., Zhou, J., Zhou, B., Yang, C., Liu, L., Gui, W., and Tian, L. (2020). Automated visual defect classification for flat steel surface: A survey. *IEEE Transactions on Instrumentation and Measurement*, 69(12):9329–9349, DOI: 10.1109/tim.2020.3030167.

Mera, C., Orozco-Alzate, M., Branch, J., and Mery, D. (2016). Automatic visual inspection: An approach with multi-instance learning. *Computers in Industry*, 83:46–54, DOI: 10.1016/j.compind.2016.09.002.

Mitchell, H. B. (2010). Image similarity measures. In *Image Fusion: Theories, Techniques and Applications*, chapter 14, pages 167–185. Springer Berlin Heidelberg, Berlin, Heidelberg, DOI: 10.1007/978-3-642-11216-4_14.

Olson, David L.and Delen, D. (2008). Support vector machines. In *Advanced Data Mining Techniques*, chapter 7, pages 111–123. Springer Berlin Heidelberg, Berlin, Heidelberg, DOI: 10.1007/978-3-540-76917-0_7.

Pekalska, E. and Duin, R. P. W. (2005). *The Dissimilarity Representation for Pattern Recognition: Foundations and Applications*, volume 64 of *Machine Perception and Artificial Intelligence*. World Scientific, Singapore, DOI: 10.1142/5965.

Soltani Firouz, M. and Sardari, H. (2022). Defect detection in fruit and vegetables by using machine vision systems and image processing. *Food Engineering Reviews*, 14(3):353–379, DOI: 10.1007/s12393-022-09307-1.

Tax, D. M. J., Loog, M., Duin, R. P. W., Cheplygina, V., and Lee, W.-J. (2011). Bag dissimilarities for multiple instance learning. In Pelillo, M. and Hancock, E. R., editors, *Similarity-Based Pattern Recognition*, volume 7005 of *LNCS*, pages 222–234. Springer, Berlin Heidelberg, DOI: 10.1007/978-3-642-24471-1_16.

Villegas-Jaramillo, E.-J. and Orozco-Alzate, M. (2023). Candy classification using convolutional neural networks, data augmentation and transfer learning: Application and a new public dataset. In *2023 IEEE 13th International Conference on Pattern Recognition Systems (ICPRS)*, pages 1–7, Guayaquil, Ecuador. IEEE, DOI: 10.1109/ICPRS58416.2023.10179072.

Waqas, M., Ahmed, S. U., Tahir, M. A., Wu, J., and Qureshi, R. (2024). Exploring multiple instance learning (MIL): A brief survey. *Expert Systems with Applications*, 250:123893, ISSN: 0957-4174, DOI: 10.1016/j.eswa.2024.123893.

Wieler, M., Hahn, T., and Hamprecht, F. A. (2007). Weakly supervised learning for industrial optical inspection [Data set]. 29th Annual Symposium of the German Association for Pattern Recognition (DAGM 2007), DOI: 10.5281/zenodo.12750201.

Zheng, X., Zheng, S., Kong, Y., and Chen, J. (2021). Recent advances in surface defect inspection of industrial products using deep learning techniques. *The International Journal of Advanced Manufacturing Technology*, 113(1):35–58, DOI: 10.1007/s00170-021-06592-8.

Zhou, Z.-H. (2018). A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, DOI: 10.1093/nsr/nwx106.

A. Symbols and Notation

A list of the symbols used for notation throughout the paper is presented in Table 2.

Table 2: List of symbols used for notation.

Objects	X, Y	Arbitrary objects without any vectorial representation; for instance, raw images
	R	Representation object
Vectors	$\boldsymbol{x}, \boldsymbol{y}$	Vectors representing objects X and Y, respectively, either in a feature space or in a dissimilarity space
	$\boldsymbol{x}_{\text{diss}}$	Vector in the dissimilarity space (only used when distinction from $\boldsymbol{x}$ in the feature space is required).
	$\boldsymbol{w}$	Hyperplane vector
	$\boldsymbol{h}_X^{(i)}$	Histogram of the i-th block in object X
Sets and bags	$\mathcal{R}$	Representation set
	$\mathcal{S}$	Test set
	$\mathcal{T}$	Training set
	$\mathcal{X}, \mathcal{Y}$	Bags of instances
Dimensions	q	Dimension of $\boldsymbol{x} \in \mathbb{R}^q$
	Q	Dimension of $\boldsymbol{x} \in \mathbb{R}^Q$, where $Q \gg q$
	$p \times l$	Size of input images
	$u \times v$	Block size
	s	Number of support vectors
	n	Number of instances in the bag $\mathcal{X}$
	m	Order of the kernel
Functions and mappings	d	Dissimilarity measure
	f	Classification function
	ϕ	Mapping
	K	Kernel functional
	D	Dissimilarity mapping
Scalars and angles	ρ	Norm of $\phi(\boldsymbol{x})$
	κ	$\cos\beta$
	b	Hyperplane offset
	α	Lagrange multiplier
	β	Angle between vectors $\boldsymbol{w}$ and $\phi(\boldsymbol{x})$
	θ	Kernel parameter
	c	Class label
	C	Support vector margin

B. Datasets and Block Decomposition Procedure

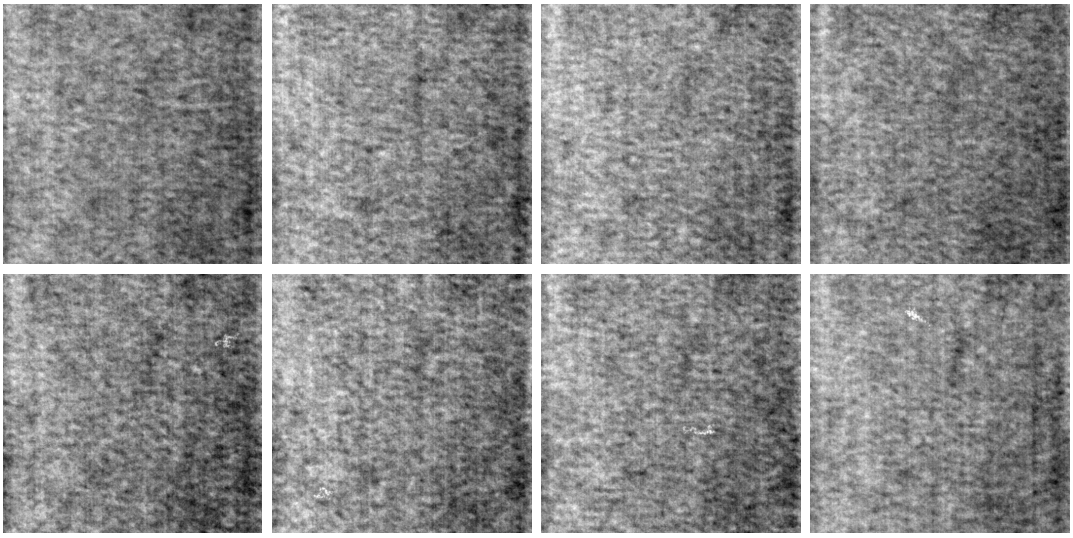
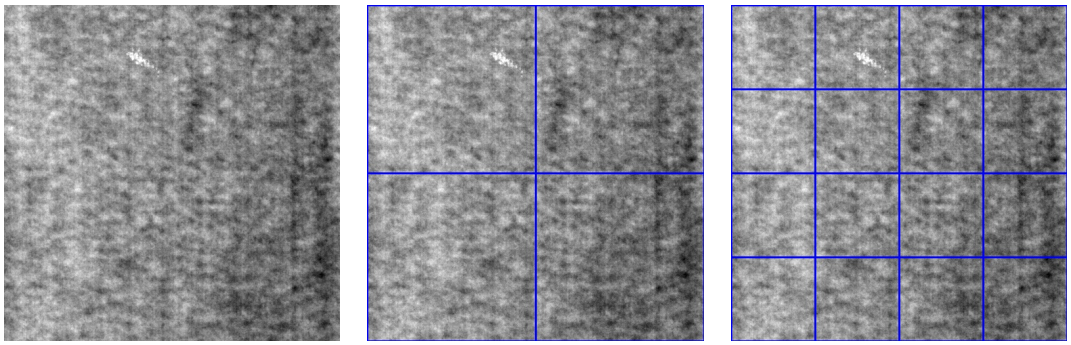


Figure 7: Examples from the DAGM dataset, labeled as *Non-Defective* in the first row, and *Defective* in the second row.

Figure 8: Examples of peanut candies dataset, labeled as *Non-Defective* in the first row, and *Defective* in the second row.

Figure 9: Examples of Lemon quality dataset, labeled as *Non-Defective* in the first row, and *Defective* in the second row.



(a) Original whole image: Bag composed by a single instance

(b) Grid decomposition of 256×256 for a bag of 4 instances

(c) Grid decomposition of 128×128 for a bag of 16 instances

Figure 10: Defective image belonging to the DAGM C3 dataset, without decomposition in (a), with decomposition into blocks of size 256×256 in (b) and with decomposition into blocks of size 128×128 in (c).

C. Results with Variability

Table 3: Classification metrics for the DAGM dataset, organized by grid size and dissimilarity measure. Standard deviation computed from 25 repetitions.

Grid	Metric	Chernoff	Jeffrey–Matusita
512×512	TP-rate	0.8175 ± 0.0570	0.9663 ± 0.0360
	FP-rate	0.0425 ± 0.0070	0.0628 ± 0.0080
	Accuracy	0.9410 ± 0.0085	0.9391 ± 0.0080
256×256	TP-rate	0.9333 ± 0.0360	0.9539 ± 0.0295
	FP-rate	0.0093 ± 0.0050	0.0099 ± 0.0045
	Accuracy	0.9830 ± 0.0065	0.9853 ± 0.0050
128×128	TP-rate	0.9876 ± 0.0170	0.9851 ± 0.0190
	FP-rate	0.0031 ± 0.0030	0.0033 ± 0.0040
	Accuracy	0.9957 ± 0.0040	0.9951 ± 0.0045

Table 4: Classification metrics for the Candy dataset, organized by grid size and dissimilarity measure. Standard deviation computed from 25 repetitions.

Grid	Metric	Chernoff	Jeffrey–Matusita
512×512	TP-rate	0.8129 ± 0.0400	0.7553 ± 0.0545
	FP-rate	0.2004 ± 0.0500	0.2546 ± 0.0505
	Accuracy	0.8040 ± 0.0365	0.7477 ± 0.0450
256×256	TP-rate	0.9889 ± 0.0150	0.9684 ± 0.0185
	FP-rate	0.0187 ± 0.0210	0.0238 ± 0.0225
	Accuracy	0.9847 ± 0.0120	0.9720 ± 0.0170
128×128	TP-rate	0.9797 ± 0.0230	0.9729 ± 0.0185
	FP-rate	0.0268 ± 0.0200	0.0300 ± 0.0235
	Accuracy	0.9760 ± 0.0160	0.9710 ± 0.0155

Table 5: Classification metrics for the Lemon quality dataset, organized by grid size and dissimilarity measure. Standard deviation computed from 25 repetitions.

Grid	Metric	Chernoff	Jeffrey–Matusita
300×300	TP-rate	0.9008 ± 0.0155	0.9223 ± 0.0095
	FP-rate	0.0905 ± 0.0150	0.0646 ± 0.0115
	Accuracy	0.9053 ± 0.0105	0.9293 ± 0.0080
150×150	TP-rate	0.9967 ± 0.0045	0.9980 ± 0.0030
	FP-rate	0.0017 ± 0.0025	0.0011 ± 0.0015
	Accuracy	0.9976 ± 0.0030	0.9985 ± 0.0020
75×75	TP-rate	0.9948 ± 0.0045	0.9975 ± 0.0035
	FP-rate	0.0040 ± 0.0045	0.0011 ± 0.0015
	Accuracy	0.9954 ± 0.0025	0.9983 ± 0.0020

Affiliation:

Eduardo Villegas-Jaramillo
Universidad Nacional de Colombia - Sede Manizales
Departamento de Informática y Computación
km 7 vía al Magdalena
Manizales, 170003, Colombia
E-mail: ejvillegasj@unal.edu.co

Jorge E. Hurtado
Universidad Nacional de Colombia - Sede Manizales
Departamento de Ingeniería Civil
Carrera 27 # 64-60
Manizales, 170004, Colombia
E-mail: jehurtadog@unal.edu.co

Mauricio Orozco-Alzate
Universidad Nacional de Colombia - Sede Manizales
Departamento de Informática y Computación
km 7 vía al Magdalena
Manizales, 170003, Colombia
E-mail: morozcoa@unal.edu.co

Journal of Data Science, Statistics, and Visualisation https://jdssv.org/
published by the International Association for Statistical Computing
http://iasc-isi.org/

ISSN 2773-0689
September 2026, Volume VI, Issue VI *Submitted:* 2026-06-10
doi:10.52933/jdssv.v6i6.178 *Accepted:* 2026-08-17