Modern Computer Vision Techniques for X-Ray Testing in Baggage Inspection

Domingo Mery, *Member, IEEE*, Erick Svec, Marco Arias, Vladimir Riffo, Jose M. Saavedra, and Sandipan Banerjee

***Abstract*—X-ray screening systems have been used to safeguard environments in which access control is of paramount importance. Security checkpoints have been placed at the entrances to many public places to detect prohibited items, such as handguns and explosives. Generally, human operators are in charge of these tasks as automated recognition in baggage inspection is still far from perfect. Research and development on X-ray testing is, however, exploring new approaches based on computer vision that can be used to aid human operators. This paper attempts to make a contribution to the field of object recognition in X-ray testing by evaluating different computer vision strategies that have been proposed in the last years. We tested ten approaches. They are based on bag of words, sparse representations, deep learning, and classic pattern recognition schemes among others. For each method, we: 1) present a brief explanation; 2) show experimental results on the same database; and 3) provide concluding remarks discussing pros and cons of each method. In order to make fair comparisons, we define a common experimental protocol based on training, validation, and testing data (selected from the public $\mathbb{GDX}$ray database). The effectiveness of each method was tested in the recognition of three different threat objects: 1) handguns; 2) shuriken (ninja stars); and 3) razor blades. In our experiments, the highest recognition rate was achieved by methods based on visual vocabularies and deep features with more than 95% of accuracy. We strongly believe that it is possible to design an automated aid for the human inspection task using these computer vision algorithms.**

***Index Terms*—Baggage screening, deep learning, implicit shape model (ISM), object categorization, object detection, object recognition, sparse representations, threat objects, X-ray testing.**

I. Introduction

BAGGAGE inspection using X-ray screening is a priority task that reduces the risk of crime, terrorist attacks, and propagation of pests and diseases [1]. Security and safety screening with X-ray scanners has become an important process in public spaces and at border checkpoints [2]. However, inspection is a complex task because threat items are very difficult to detect when placed in closely packed bags, occluded by other objects, or rotated, thus presenting an unrecognizable view [3]. Manual detection of threat items by human inspectors is extremely demanding [4]. It is tedious because very few bags actually contain threat items, and it is stressful because the work of identifying a wide range of objects, shapes, and substances (metals, organic, and inorganic) takes a great deal of concentration. In addition, human inspectors receive only minimal technological support. Furthermore, during rush hour, they only have a few seconds to decide whether a bag contains any threat item or not [5]. Since each operator must screen many bags, the likelihood of human error becomes considerable over a long period of time even with intensive training. The literature suggests that detection performance is only about 80%–90% [6]. In baggage inspection, automated X-ray testing remains an open question due to: 1) *loss of generality*, which means that approaches developed for one task may not transfer well to another; 2) *deficient detection accuracy*, which means that there is a fundamental tradeoff between false alarms and missed detections; 3) *limited robustness* given that requirements for the use of a method are often met for simple structures only; and 4) *low adaptiveness* in that it may be very difficult to accommodate an automated system to design modifications of different specimens.

There are some contributions in computer vision for X-ray testing such as applications on inspection of castings, welds, food, cargos, and baggage screening [7]. For this paper, it is very interesting to review the advances in baggage screening that have taken place over the course of this decade. They can be summarized as follows. Some approaches attempt to recognize objects using a single view of mono-energy X-ray images (e.g., the adapted implicit shape model (ISM) based on visual codebooks [8] and adaptive sparse representations (XASR+) [9]) and dual-energy X-ray images (e.g., Gabor texture features [10], bag of words (BoWs) based [11], [12], and pseudo-color, texture, edge, and shape features [13]). More complex approaches that deal with multiple X-ray images have been developed as well. For the recognition of regular objects from mono-energy images, methods like data association [14], [15], and active vision [16], where a second-best view is estimated, have been explored. In the case of dual-energy imaging, visual vocabularies, and support vector machines (SVM) classifiers have been used, as shown in [17]. Progress also has been made in the area of computed tomography. For example, in order to improve the quality of CT images, metal artifact reduction and de-noising [18] techniques were suggested. Many methods based on 3-D features for 3-D object

Manuscript received August 7, 2016; accepted October 1, 2016. This work was supported by the Fondecyt from CONICYT, Chile, under Grant 1161314. This paper was recommended by Associate Editor K. Huang.

D. Mery, E. Svec, and M. Arias are with Department of Computer Science at Pontificia Universidad Católica de Chile, Santiago 7820436, Chile (e-mail: dmery@ing.puc.cl).

V. Riffo is with DIICC-Universidad de Atacama, Casilla 240 Chile.

J. M. Saavedra is with ORAND S.A., Santiago 8320180, Chile.

S. Banerjee is with the University of Notre Dame, Notre Dame, IN 46556 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2016.2628381

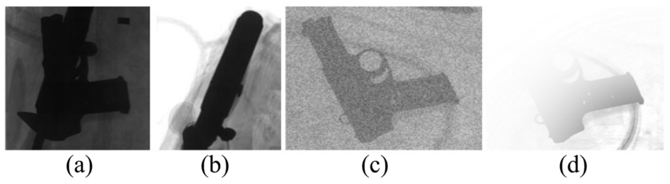


Fig. 1. Problems in recognition of a *gun*. (a) Occlusion. (b) Self-occlusion. (c) Noise. (d) Wrong acquisition.

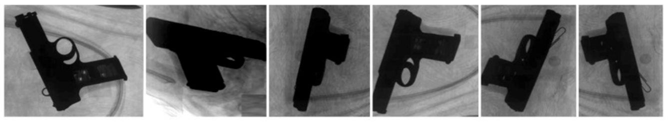


Fig. 2. Large variability within a *gun*. Some X-ray images of the same gun in different poses.

recognition have been developed (see rotation invariant feature transform and scale-invariant feature transform (SIFT) descriptors [19], 3-D visual cortex modeling 3-D Zernike descriptors and histogram of shape index [20]). There are contributions using known recognition techniques (see BoWs [21] and random forest [22]) as well. As we can see, the progress in automated baggage inspection is modest and very limited compared to what is needed because X-ray screening systems are still being manipulated by human inspectors. Automated recognition in baggage inspection is far from perfect given that the appearance of the object of interest can become extremely difficult to comprehend due to problems of (self-) occlusion, noise, acquisition, and clutter among others (as illustrated in Fig. 1). Furthermore, the large variability within an object sample depending on its points of view (e.g., top view and frontal view of a *gun* are very different as shown in Fig. 2).

The main difference between X-ray and photographic (optical) imaging is that an optical image is formed by light reflecting from an object (giving information about its surface), whereas an X-ray image is formed by irradiating the object with X-rays that pass through the object. The X-rays are attenuated according to absorption's law and the density of the structures of the object (giving information about its internal structure) [7]. Thus, an X-ray image consists of *shadows* from *transparent layers* that are superimposed and overlapped. In case an object that has a dense material, only its silhouettes are captured. On the other hand, in cases of an object that has relatively lesser dense material in presence of other objects, the image of the structure is so transparent that the objects behind or in front of it may be captured as well [23].

Despite these main differences, object recognition using optical images (in case the object to be recognized can be captured by an optical imaging system) and object recognition using X-ray imaging (in case the object to be recognized can be captured by an X-ray imaging system) share many similar problems, such as perspective imaging, geometric distortion, pose problems, (self-) occlusion, noise, and large intraclass variability among others. For this reason, we believe that algorithms based on modern computer vision techniques on optical images can be used for this general recognition task in X-ray testing. In addition, one could take advantage of the promising advances that have occurred in recent years in many computer vision applications with optical images, especially in object recognition [24].

Three-dimensional object recognition from 2-D images is a very complex task in computer vision in general, not only with X-ray images but also with conventional photographic images, given the infinite number of viewpoints, different acquisition conditions, and objects that are deformable, occluded, or embedded in clutter [25]. In certain cases, automated recognition is possible through the use of approaches focused on obtaining highly discriminative and local invariant features related to lighting conditions and local geometric constraints (see [26] for a good review and evaluation of descriptors including the well-known SIFT [27] and speeded up robust features [28] features) or texture features (see [29]). A test object can be recognized by matching its invariant features to the features of a model. Over the past decade, many approaches have been proposed in order to solve the problem of 3-D object recognition. Certain approaches focus on learning new features from a set of representative images (see visual vocabularies [30], ISMs [31], mid-level features [32], sparse representations [33], and hierarchical kernel descriptors [34]). For instance, Fisher vectors [35] and vector of locally aggregated descriptors [36] on SIFT features has been used successfully in recognition problems. In addition, sparse representation has been widely used in computer vision [37], [38]. In many computer vision applications, under the assumption that natural images can be represented using sparse decomposition, state of the art results have been significantly improved. However, these methods may fail when the learned features cannot provide a good representation of viewpoints that have not been considered in the representative images. Additionally, some approaches include multiple view models (see an interconnection of single-view codebooks across multiple views [39], a learned dense multiple view representation by pose estimation and synthesis [40], a model learned iteratively from an initial set of matches [41], a model learned by collecting viewpoint invariant parts [42], 3-D representations using synthetic 3-D models [43], and a tracking-by-detection approach [44]). These methods may fail, however, when objects have large intraclass variation. On the other hand, object recognition can be improved when color imaging (RGB) is used in conjunction with 3-D sensing technologies to include depth images (D), as shown in some approaches that use RGD-D sensors in [45]–[47]. Nevertheless, how to effectively combine multimodal information (color, texture, appearance, shape, and geometry) remains an open problem [45]. Other applications that include 3-D information can be found in inspection problems (see [48] that uses a laser range finder camera).

In recent years, "deep learning" has been successfully used in image and video recognition [49], [50]. The key idea is to replace *handcrafted* features with features that are *learned* efficiently using a hierarchical feature extraction approach. There are several deep architectures such as deep neural networks, convolutional neural networks (CNNs), energy based models, Boltzmann machines, deep belief networks, deep residual learning among others [50], [51]. CNNs, which were inspired by a biological model [52], has been established as a very

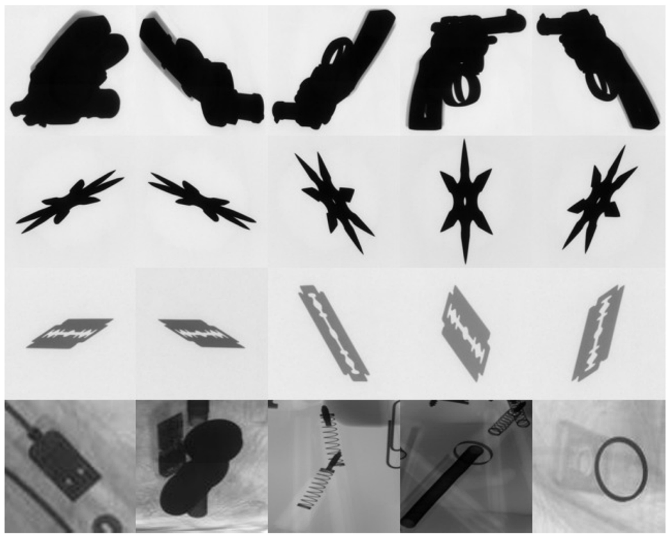


Fig. 3. Some training X-ray images used in our experiments. Each row represents a labeled class (handguns, shuriken, razor blades and others, respectively).

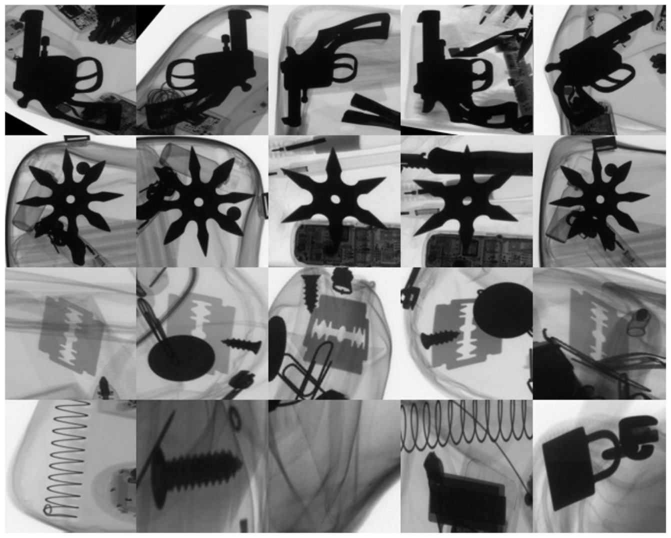


Fig. 4. Some testing X-ray images used in our experiments. Each row represents a labeled class (handguns, shuriken, razor blades and others, respectively).

powerful method for image recognition [53]. CNN replaces feature extraction and classification with a single neural network. CNN maps an input image $\mathbf{x}$ on an output vector $\mathbf{y} = f(\mathbf{x})$, where function f can be viewed as a sequence of convolutional functions $f_1, \ldots, f_L$, i.e., linear 2-D filters. The functions contain parameters $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_L)$ that can be discriminatively learned from example data $(\mathbf{x}_i, \mathbf{y}_i)$, for $i = 1, \ldots, n$, so that $\sum_n \ell(f(\mathbf{x}_i, \mathbf{w}), \mathbf{x}_i) \to \min$, where ℓ is a loss function. This optimization problem can be solved using the back-propagation approach [54].

This paper attempts to make a contribution to the field of object recognition in X-ray testing by evaluating different computer vision strategies that have been proposed in the last years. We tested ten approaches. They are based on BoWs, sparse representations, deep learning, and classic pattern recognition schemes among others. For each method, we present: 1) a brief explanation; 2) relevant references for further information; 3) experimental results on the same database; and 4) concluding remarks discussing pros and cons. In order to make fair comparisons, we define a common experimental protocol based on training, validation, and testing data (selected from the public $\mathbb{GDX}$ray [55] database). The effectiveness of each method was tested in the recognition of three different threat objects: 1) handguns; 2) shuriken (ninja stars); and 3) razor blades.

The rest of this paper is organized as follows. In Section II, the computer vision methods used in our experiments are briefly explained. In Section III, the experimental results are presented. Section IV concludes this paper.

II. Computer Vision Methods

In this section, we describe the computer vision techniques that we use for our X-ray testing experiments. Some of them have been developed by integrating well-known computer vision algorithms (see Sections II-A and II-B). Some of them have been already tested on X-ray images (see Sections II-C and II-D). Some of them are based on deep learning techniques (see Section II-E). Finally, we include some classic methods in computer vision as baseline (see Section II-F).

The task of each method is to recognize the object that is present in a cropped X-ray image. To this end we use labeled X-ray images from the database $\mathbb{GDX}$ray [55] for training and for testing purposes (see examples in Figs. 3 and 4, respectively). In our experiments, there are three classes: 1) Gun; 2) Shuriken; and 3) Blade (for handguns, ninja stars, and razor blades, respectively). We use an additional negative class (called Others) for training purposes in which none of the mentioned objects are present.

The explanations given in this section describe each method in two steps: 1) learning and 2) testing. In the first step, the algorithm learns a model from training images in a supervised way (meaning the labels of the class of each training image are known). In the second step, the trained algorithm is tested on new X-ray testing images that have not been used in the previous step. In order to measure the performance of the designed algorithm, the predicted and annotated labels are compared.[1]

A. Bag of Words

BoWs model is a well-known methodology that has been widely used in the computer vision community on optical images [56] and X-ray images [12]. This methodology commonly achieves a high performance by reducing the amount of features to just the most representatives ones.

Learning: In this stage, we design three independent binary classifiers (for the target classes: Gun, Shuriken, and Blade). In order to reduce the noise, the X-ray images are filtered using a Gaussian low-pass-filter. Afterwards, SIFT keypoints are detected [27]. SIFT descriptors of 128 elements and local

[1]In our experiments, we use an additional set of images for validation. This set is used to tune the parameters of the model only.

binary pattern (LBP) (rotation-invariant) features of 36 elements [57] are extracted centered in the location of the SIFT keypoints. For LBP the size of the window was $2s \times 2s$ pixels, where s is the scale determined by SIFT. Two visual dictionaries are defined using K-means with Euclidean distance [58]: one for SIFT descriptors and another for LBP features, in which the centroids of the clusters, i.e., the codewords, are stored in $\mathbf{X}_{\text{SIFT}}$ and $\mathbf{X}_{\text{LBP}}$ respectively. Thus, the extracted features (SIFT and LBP) are quantized into the corresponding nearest visual word of $\mathbf{X}_{\text{SIFT}}$ and $\mathbf{X}_{\text{LBP}}$. Each training image is represented by the concatenation of two histograms of visual words ($\mathbf{h}_{\text{SIFT}}$ and $\mathbf{h}_{\text{LBP}}$) that are computed by binning the quantized visual words. Finally, a random forest classifier [59] is trained for each target class in which training images of the class Others are to be considered.

Testing: Similarly to training stage, a concatenated histogram (from $\mathbf{h}_{\text{SIFT}}$ and $\mathbf{h}_{\text{LBP}}$) is computed for the testing image. The histogram is used as inputs of the three random forest classifiers. A score to each possible class is obtained, and the predicted class is the one with the highest score.

B. KNN-Based in Sparse Reconstruction Object Recognition

In this section, we describe KNN-based in sparse reconstruction object recognition (sparse KNN), a new method for X-ray testing based on sparse representations.

Learning: It consists of five steps.

1) For each image of training set, the object is segmented using an adaptive K-means clustering for grayscales images [60] and morphological transformations. This generates a mask in which SIFT keypoints [27] are extracted (obtaining matrix $\mathbf{F}_{1,i}$ of $128 \times r_i$ elements, where i is the number of the class, for $i = 1 \ldots 4$, and r_i is the number of keypoints in all training images of class i). The label of each extracted SIFT descriptor is stored in vector $\mathbf{d}_{\text{train}}$ (with $r = \sum r_i$ elements).
2) For each target class $j = 1, 2, 3$, an offline feature selection is done using sequential forward selection (SFS) [61], using the selected class j against all other, this is using all features with label $\mathbf{d}_{\text{train}} = j$ against features with label $\mathbf{d}_{\text{train}} \neq l$, and using as criterion method KNN with $k = 5$ and selecting $s = 50$ (from 128) features in each case (obtaining matrix $\mathbf{F}_{2,i,j}$ of $50 \times r_i$ elements, this corresponds to the features for i class using SFS feature selection for j class). The idea behind this feature selection, is mainly, to deal with our particular occlusion, selecting only gradients corresponding to the object and discarding background.
3) We calculate alpha feature, $\alpha = d/(wh)$ where d is the distance to the center of the image, w is image width, and h the height, maintaining keypoints relative position (obtaining matrix $\mathbf{F}_{3,i,j}$ of $51 \times r_i$).
4) For each matrix $\mathbf{F}_{3,i}$ a K-means algorithm [58] is calculated, then, the centroids of the clusters are used as our features and stored in a feature matrix $\mathbf{F}_{4,i,j}$ of $51 \times c_i$, where c_i is the number of centroids for class i; with this we reduce the dimensionality from duplicated or very similar keypoints.
5) For each class i the dictionary $\mathbf{D}_i$ is constructed as the concatenation of the resultant features matrices in step four, then $\mathbf{D}_i = [\mathbf{F}_{4,i,1} \quad \mathbf{F}_{4,i,2} \quad \mathbf{F}_{4,i,3}]$. Finally, label vector $\mathbf{dc}_{train,i}$ for features in $\mathbf{D}_i$, is calculated using the labels obtained from step four.

Algorithm 1 Soft-Voting Classification

```
for all $j$ do
    for all class $i \neq 4$ do
        $votes(i) = \sum_j d_{\theta,i} - distances(\hat{y}_j, i)$
if $std(votes) < d_0$ then
    $predictedClass(I_q) \leftarrow 4$
else
    $predictedClass(I_q) \leftarrow max(votes)$
```

Testing: In this stage, for testing image I_q feature extraction does not include object segmentation, extracting SIFT points directly from the image, selecting features with SFS and adding α feature, then, for each $\mathbf{y}_j$ vector resultant a vote is calculated as follows. A $\hat{\mathbf{y}}_i$ sparse reconstruction is made. For this, first, with $\mathbf{D}_i$ we find the sparse representation vector $\mathbf{x}$ of $\mathbf{y}_j$, then, the sparse reconstruction $\hat{\mathbf{y}}_j$ is calculated with $\mathbf{D}_i$ and $\mathbf{x}$ as explained in [62]. This sparse reconstruction includes information from one, two or many classes. Then, a threshold over the sparsity concentration index (SCI) is computed in order to evaluate how spread are its sparse coefficients [63], so the method can decide whether to continue processing the vector or classify it as the class Others. If the vector is not discarded, its vote is calculated normalizing the output of a KNN classifier trained with $\mathbf{D}_i$. Using the closest distance to a neighbor (k is not necessarily 1), we use a distance threshold d_θ for each class to determine if the sample is close enough to his neighbor or too far to take a clear decision, and as consequence, discarding this from votes. Finally, the predicted class will be selected as the one with the higher votes. Then, a threshold for the maximal distance d_0 is calculated by comparing this parameter with the standard deviation of the votes sample and used to detect untrusted set of votes, in witch case, I_q is classified as Others class. Soft voting algorithm is shown in Algorithm 1 in which $i = 4$ means the class Other.

In order to show the effectiveness of some modules our proposed sparse KNN, a baseline method called sparse KNN* was defined. This method corresponds to sparse KNN but excluding the SFS phase, therefore having a resultant feature vector of 129 dimensions (128 from SIFT and alpha feature); and using a binary voting system, where votes are "1" or "0" depending if it is classified or not as the given classifier class.

C. Adaptive Implicit Shape Model

Adaptive ISM (AISM) was presented originally in [8] for object recognition in baggage screening. AISM is based on the well-known ISM method [64] which was developed for recognition of object categories such as cars, people and animals in photographs. AISM adapted this methodology in order to detect object categories in single X-ray images that were acquired using an X-ray system.

Learning: The training stage is based on the creation of a visual vocabulary using *keypoints* and local visual descriptors. In this stage, a target object is represented using a visual vocabulary of parts (category-specific appearance codebook). Keypoints and their local visual descriptors are extracted automatically from all training images of the target object using the well-known SIFT approach [27]. Thus, an object category is characterized by estimating a visual vocabulary of the object parts and a measurement of their spatial distribution.

Testing: In the testing stage target objects are detected by searching similar visual words and similar spatial distributions. More details can be found in [8].

D. Adaptive Sparse Representations

An object recognition approach that has been tested in baggage screening called XASR+ was proposed in [9].

Learning: In the training stage, for each object of training dataset, many patches are extracted from its X-ray images in order to construct representative dictionaries. Each patch is described using some feature (e.g., grayvalues, LBPs [57], SIFT [27], etc.). In our experiments, we use as descriptor a combination of SIFT and LBP (rotation invariant version). A stop-list is used to remove very common words of the dictionaries [30].

Testing: In the testing stage, many test patches of the testing image are extracted, and for each test patch a dictionary is built concatenating the "best" representative dictionary of each object. Using this adapted dictionary, each test patch is classified following the sparse representation classification (SRC) methodology [63]. Finally, the testing image is classified by patch voting. Thus, XASR+ is able to deal with less constrained conditions including some contrast variability, pose, intraclass variability, size of the image, and focal distance. See more details in [9].

E. Deep Learning

Motivated by the tremendous success of deep learning, specially the convolutional neural network (CNN), as we mentioned in Section I, we present a strategy based on deep features (that are able to deal with noisy background) and a nearest neighbor classifier (that is able to deal with the potential risk of overfitting on the $\mathbb{GDX}$ray dataset).

Learning: Training CNN models from scratch with our own data did not yield good results. The models were found to be strongly overfitted on the training data, heavily biased toward the negative Others class. This can be attributed to the disparity in the distribution of the data samples among the four classes in training and the low number of available X-ray images. The strong bias toward the Others class resulted from the fact that it contained more than half of the image samples in the training set. The other approach we tried was to fine tune the CNN models by initializing training with a set of weights transferred from an already converged state of the same model, trained on the much larger ImageNet dataset [24]. However, this approach failed to overcome the problem of overfitting as well. Since training the networks with our own data did not yield beneficial results, we decided to use the trained CNN models as generic feature extractors instead. Thus, we used a CNN model that was previously trained with a large and highly-variable collection of (optical) images. Particularly, we use a CNN trained with the ImageNet dataset. We then take the responses of one of the hidden layer of the CNN model to be considered as the feature vector in our problem. To this end, we evaluate the two most popular CNN models related to ImageNet: AlexNet [53] and GoogleNet [65]. In our experiments, we use the same trained CNN models proposed by the authors of AlexNet and GoogleNet.[2]

Testing: The testing stage is very simple. Considering the high variability between training and testing dataset, trying to train a discriminative model may produce an overfitting effect. Therefore, instead of using a discriminative model we propose to use a simple nearest neighbor classifier (KNN). That means, that the label of an input image is the label of its nearest neighbor in the training dataset. To compute the nearest neighbor we could use the Euclidean distance (L_2). However, using this kind of distance may produce a *bursting effect*. That is, the final distance between two feature vectors can be biased by a few dimensions with high differences. To reduce this effect, we use the Hellinger function [66] instead of the standard L_2 function. This can be achieved if we normalize the feature vector taking the square root of each feature value and then transforming the whole vector to the unit. Therefore, before computing distances between feature vectors, we apply a square root normalization on each vector. Thus, in testing stage, deep features are extracted from the testing image and they are classified using KNN.

F. Baseline Methods

In this section, we describe briefly three classic computer vision methods that can be used for this task: SVM, AdaBoost, and SRC. They are uses as baseline methods in order to compare the performance. Four these three classifiers we follow the same methodology.

Learning: SIFT features are extracted for each image in training set, then with these features labeled, one model for each class is trained.

Testing: In testing stage, for each testing image SIFT keypoints are extracted, each keypoint is then classified used the trained model. These classifications are used as votes and with a threshold we determine the predicted class of the testing image.

III. Experimental Results

In this section, we present the evaluation protocol and the implementation details of each of the computer vision methods explained in the previous section. We also report and discuss the achieved results.

[2]The models are available at https://github.com/BVLC/caffe/wiki/Model-Zoo.

TABLE I
IMAGES OF $\mathbb{GDX}$RAY [55] USED IN OUR EXPERIMENTS

Set		Gun	Shuriken	Blade	Others
Training	Series	B0049	B0050	B0051	B0078
	Images	1–200	1–100	1–100	1-500
Validation	Series	B0079	B0080	B0081	B0082
	Images	1–50	1–50	1–50	1–200
Testing	Series	B0079	B0080	B0081	B0082
	Images	51-150	51-150	51-150	201-600

A. Experimental Protocol

In our experiments, there are three objects: 1) handguns; 2) shuriken (ninja stars); and 3) razor blades. Each category of objects defines a class (Gun, Shuriken, and Blade). Furthermore, there is a fourth class called Other for other objects and background. All X-ray images used in our experiments belong to the $\mathbb{GDX}$ray[3] database [55]. As shown in Table I, there are three different sets of images: 1) training; 2) testing; and 3) validation sets. For training, X-ray images of $\mathbb{GDX}$ray series B0049, B0050, B0051, and B0078 must be used for classes Gun, Shuriken, Blade, and Others, respectively. For validation, in case that a method has some parameters to be tuned, it is allowed to use the first 50 images of $\mathbb{GDX}$ray series B0079, B0080, and B0081 for Gun, Shuriken, and Blade, respectively, and the first 200 images of folder B0082 for Others. For testing, the last 100 images of $\mathbb{GDX}$ray series B0079, B0080, and B0081 for Gun, Shuriken, and Blade, respectively, and the last 400 images of folder B0082 for Others have to be used.

The $\mathbb{GDX}$ray dataset is specially challenging due to the high intraclass variability between training and testing images of positive classes (see some examples for guns, shuriken, and razor blades in Figs. 3 and 4 for training and testing, respectively). Indeed, training images of positive classes contain just the object with a clean background. In contrast, testing images corresponding to the these classes show a noisy background that may allow any discriminative model to classify them as the class Others.

In our experiments, we define two recognition tasks: 1) four-class classification and 2) detection of three threat objects.

1) Four-Class Classification: In the first problem, we have to design a classifier that is able to recognize the four mentioned classes: 1) Gun; 2) Shuriken; 3) Blade; and 4) Others. We define $m = 4$ as the number of classes. The classifier has to be trained using the trained data. The parameters of the classifier (if any) can be tuned using the validation only. The performance of the method must be reported using the testing data as follows. The elements of the $m \times m$ confusion matrix are defined as $C(i, j)$ for $i = 1 \ldots m$ and $j = 1 \ldots m$, where $C(i, j)$ means the number of images of class i (in the testing data) classified as class j. The accuracy of each class is defined as

$$\eta_i = \frac{C(i,i)}{\sum_{j=1}^{m} C(i,j)} \times 100. \tag{1}$$

The total accuracy is the average

$$\eta = \frac{1}{m}\sum_{i=1}^{m} \eta_i. \tag{2}$$

The $m + 1$ values $\eta_1 \ldots \eta_m$ and η must be reported.

2) Detection of Three Threat Objects (Three Binary Classifiers): In the second problem, we have to design three different detectors (binary classifiers) : 1) one for Gun; 2) one for Shuriken; and 3) one for Blade. For each detector there is a target (e.g., Shuriken for second detector). Each detector can be understood as a 2-class problem: one class (called the positive class) is the target, and the another class (called the negative class) is the rest. Similar to the previous problem, training data must be used to train the detectors, validation data can be used to tune the detectors' parameters (if any), and testing data have to be used to measure the final performance of the detectors. For the second detector (i.e., Shuriken), for example, in our database according to Table I, there are 100 images for the positive class and $200+100+500 = 800$ images for the negative class that can be used for training purposes. In this example, the validation can be performed using 50 images for the positive class and $50 + 50 + 200 = 300$ images for the negative class. Finally, for the testing of the second detector, there 100 images for the positive class and $100+100+400$ for the negative class. The performance must be given in terms of precision–recall (Pr, Re) considering all images of the testing set. The variables precision and recall are defined as follows:

$$\text{Pr} = \frac{\text{TP}}{\text{TP}+\text{FP}}, \quad \text{Re} = \frac{\text{TP}}{\text{TP}+\text{FN}} \tag{3}$$

where

True Positive (TP):	Number of targets correctly classified;
True Negative (TN):	Number of nontargets correctly classified;
False Positive (FP):	Number of nontargets classified as targets. The false positives are known as "false alarms" and "type I error";
False Negative (FN):	Number of targets classified as no-targets. The false negatives are known as "type II error."

Ideally, a perfect detection means all existing targets are correctly detected without any false alarms, i.e., $\text{Pr} = 1$ and $\text{Re} = 1$.

The values (Pr, Re) that maximizes the score $Q = \sqrt{\text{Pr} \times \text{Re}}$ must be reported. As average performance, we define

$$p = \frac{1}{3}\sum_{i=1}^{3} Q_i \times 100 \tag{4}$$

where $i = 1 \ldots 3$ means the classes Gun, Shuriken, and Blade, respectively.

[3]$\mathbb{GDX}$ray is a public database for X-ray testing with more than 20 000 images. The X-ray images included in $\mathbb{GDX}$ray can be used free of charge, for research and educational purposes only. Available at http://dmery.ing.puc.cl/index.php/material/gdxray/.

TABLE II
PERFORMANCE FOR FOUR-CLASS PROBLEM

	Gun	Shuriken	Blade	Others	Total
	η_1	η_2	η_3	η_4	η
BoW	97.0	92.0	82.0	87.0	90.0
Sparse KNN	97.0	99.4	91.9	90.6	94.7
Sparse KNN*	93.0	93.2	87.3	83.4	89.2
AISM	96.0	94.0	99.0	92.5	95.4
XASR+	91.0	99.8	71.0	88.3	87.5
GoogleNet	100.0	100.0	95.0	90.0	96.3
AlexNet	99.0	100.0	72.0	93.5	91.2
SVM	91.0	86.0	86.0	79.0	85.5
AdaBoost	87.0	86.0	84.0	60.0	79.3
SRC	79.0	83.0	52.0	80.0	73.5

TABLE III
PRECISION AND RECALL FOR EACH DETECTOR

	Gun		Shuriken		Blade		Total
	Pr	Re	Pr	Re	Pr	Re	p
BoW	0.65	0.84	1.00	0.92	1.00	0.97	89.4
Sparse KNN	0.99	0.97	1.00	0.99	0.97	0.94	97.7
Sparse KNN*	0.92	1.00	1.00	0.93	0.99	0.88	95.2
AISM	0.97	0.97	0.95	0.96	0.99	0.99	97.2
XASR+	0.92	0.88	0.69	1.00	0.78	0.97	86.7
GoogleNet	0.83	1.00	0.99	1.00	0.84	0.95	93.3
AlexNet	0.85	0.99	1.00	1.00	0.90	0.72	90.7
SVM	0.90	0.99	1.00	0.85	0.51	1.00	86.0
AdaBoost	1.00	0.87	1.00	0.86	0.98	0.87	92.8
SRC	0.75	1.00	0.80	1.00	0.45	0.98	80.8

B. *Results and Discussion*

In this section, we present the results obtained using the ten methods outlined in Section II in two recognition tasks: the four-class classification problem (see Section III-A1) and the problem of detection of three threat objects (see Section III-A2). The results are summarized in Tables II and III, respectively.

In the first recognition problem, the idea was to design a unique classifier that is able to recognize four classes (Gun, Shuriken, Blade, and Others). From Table II, we can see that there are five methods that achieved an accuracy $\eta \geq 90\%$ [see definition in (2)]. It is clear, however, that an accuracy around 95% or more is possible: see rows GoogleNet (96.3%), AISM (95.4%), and sparse KNN (94.7%). Moreover, these three computer vision techniques were able to recognize more than 90% of each class in the testing images. It is worth noting that the best performance was achieved by GoogleNet, an algorithm based on a convolutional neural net (CNN). This result is very interesting because, as we explained in Section II-E, the CNN model had been trained with optical images (and not with X-ray images). It can be noted that the features provided by GoogleNet seem to be better than those obtained from AlexNet. This behavior can be attributed to the fact that GoogleNet was thought to address the object detection problem while AlexNet, as simpler model, that was thought just for classification. Hence, GoogleNet could be more robust to noisy backgrounds which is the main problem of this paper. Nevertheless, as we can see in Table II, there are two methods that were able to achieve similar results to deep learning: sparse KNN and AISM. They were better than AlexNet, and respectively only 1.6% and 0.9% lower than GoogleNet.

In the second problem, the aim was to design three different and independent detectors: one for Gun, one for Shuriken, and another for Blade. In Table III, precision and recall for each detector and the average performance p [see definition in (4)] are shown. Similarly to four-class problem, we can see that there are several methods with a low performance. In addition, methods based on deep learning achieved between 90% and 94% only. Nevertheless, there are three methods that achieved a high performance ($p > 95\%$): see rows sparse KNN (97.7%), AISM (97.2%), and sparse KNN* (95.2%). Probably, the reason of this resulting performance is because the mentioned models have been learned from X-ray images (and not from optical images as in case of deep features).

If we analyze the performance of both experiments together, we can compute the average of the cumulative performance of each method (average of last column of Table II and last column of Table III). We can classify our methods into three groups by this analysis: 1) low; 2) moderate; and 3) high performance methods. It is clear that baseline methods like SVM, AdaBoost, and SRC belong to the first group because they achieved around 85% or lower. In the second group, where the performance was around 90 $\pm$ 3%, there were the methods sparse KNN*, AlexNet, BoW, and XASR+. Finally, in the third group, where the performance was around 95% or higher we have: AISM (with 96.3%), sparse KNN (with 96.2%), and GoogleNet (with 94.8%). In both experiments, we can observe that modern computer vision techniques based on learned representations, such as visual dictionaries, and deep features are able to deal with recognition problems in baggage inspection using X-ray images. This result is also consistent with other recognition problems using computer vision in optical images, where the best performance has been achieved by this kind of approaches.

We believe, that a CNN trained with a very large number of X-ray images (instead of optical images like GoogleNet and AlexNet) would lead to better results in X-ray testing. Moreover, as Section III-C shows, it is worth mentioning that the lowest computational time of testing stage was achieved by the deep learning methods.

C. *Practical Considerations*

In this section, we report the implementation details of each method explained in Section II. The computational time depends on the software implementation and the computer architecture. In order to present a reference, in this paper we give the details of the computational time for four-class problem.

1) BoWs: This model uses four parameters for each target class (Gun, Shuriken, and Blade):

$\theta = (\sigma, K_{\text{SIFT}}, K_{\text{LBP}}, n_{\text{tree}})$. Parameter σ is the standard deviation of the Gaussian low-pass-filter. Parameters K_{SIFT} and K_{LBP} are the number of clusters of the visual vocabularies $\mathbf{X}_{\text{SIFT}}$ and $\mathbf{X}_{\text{LBP}}$, respectively. Finally, parameter n_{tree} is the number of trees in the forest. Each parameter was tuned using exhaustive search in order to maximize the performance metrics in the validation set (for $\sigma = 0, 1 \ldots 8$; $K_{\text{SIFT}} = 100, 200 \ldots 400$; $K_{\text{LBP}} = 50, 100 \ldots 200$; and $n_{\text{tree}} = 1000, 2000 \ldots 4000$). The individual best parameters for each classifier were $\theta = (8, 100, 200, 2000)$, $(6, 200, 100, 4000)$, and $(2, 400, 100, 2000)$ for Gun, Shuriken, and Blade, respectively. The random forest uses Gini impunity [67] as homogeneity metric. The computational time was 30 min for the training stage and about 1 s per testing image on advanced micro device-processor FX(tm)-6300 Six Core, 3.5 GHz, 8GB RAM.

2) Sparse KNN: Main part of this method's accuracy relies on parameter and thresholds tuning. For SFS, s was selected by visual inspection in the curves $J_{\max}$ versus ns of each classifier and in order to maintain vectors of same size and a fair comparison among the results, we decided to use s as the same for each classifier. k is selected as the same k in our classifier, and this is done by exploration from $k = 1, 3, 5, 7, 9$. We use the SIFT implementation of from VLFeat library [68]. All SIFT parameters are used as default except `PeakThresh`, where `PeakThresh = 1`. This is to avoid selecting keypoints that are actually noise in the image. The threshold for the SCI (θ_{SCI}) was tuned using $\theta_{\text{SCI}} \in [0, 1]$ using 1 decimal precision. All distance threshold where iterated in the interval [10000, 100000] with jumps of 10 000 in a for loop, parameters for each class where selected at the same time, is meaningful to mention that the value for the Others class is 0, meaning that none $\hat{\mathbf{y}}_j$ classified as Others is affected. The d_0 parameter is also iterated using $d_0 \in [0, 100000]$ using intervals of 100. Finally, selected parameters of this section are: $s = 50$, $k = 5$, $\theta_{\text{SCI}} = 0.9$,[4] the distance thresholds for each target class where $d_{\theta,1} = 50000$, $d_{\theta,2} = 80000$, and $d_{\theta,3} = 90000$, for Gun, Shuriken, and Blade, respectively, and the threshold for the maximal distance was $d_0 = 21700$. As for methods implementations we use [60] for adaptive K-means, for sparse reconstruction we use SPAMS library from INRIA [62] and all other implementation are taken from Balu Toolbox [69]. For sparse KNN and sparse KNN*, the computational time was, respectively, 240 and 220 min for the training stage and about 20 s and 25 s per testing image on a Mac Mini Server OS X 10.10.1, processor 2.6 GHz Intel Core i7 with four cores and memory of 16GB RAM 1600 MHz DDR3.

3) AISM: In training stage, an implicit representation of each object is obtained, i.e., each object has different parts representing its shape. To achieve this, we used an agglomerative clustering, which clusters similar parts. The clustering process stops when a certain number of clusters is obtained. In our implementation, better results were achieved when the predefined number of clusters for each object category was set to 400. Then, the structure called "occurrence" is calculated for each cluster. The occurrence of cluster p, denoted as set $\mathbb{Z}_p$, for $p = 1, \ldots, 400$, contains all of the keypoints of the training images whose SIFT-descriptors are similar enough to the center of mass of each cluster. In step of *testing*, it was necessary to tune several parameters which are different threshold values and window sizes. See details in [8]. In this paper, we have tuned the threshold values using exhaustive search, in order to maximize the performance metrics in the validation set. Parameter θ_u is the minimum distance threshold allowed between all keypoints $\mathbf{f}_k$ stored in the training database and the keypoints test image $\mathbf{f}_t$, θ_B is the minimum number of keypoints of the same pose enclosed in the subwindows W_B, θ_m is the minimum number of candidates encloses in the subwindow W_m (whose size is not predefined), and if no subwindow W_m meets this condition, no potential target object is detected, W_F is the final size of the detection window. Where θ_u = (50.000, 30.000, 50.000), θ_B = (1, 3, 5), θ_m = (4, 11, 1), W_B = (175 × 175, 150 × 150, 100 × 100) [pixels] and W_F = (800 × 1.300, 820 × 820, 200 × 360) [pixels] for Gun, Shuriken, and Blade, respectively. The computational time for the training stage was 27 min, and about 8 s per testing image on an Intel Core i7-3537U CPU @ 2.00 GHz with four cores and memory RAM of 8 GB. The algorithms were implemented in MATLAB R2014a, 64-bit (win64).

4) XASR+: This has six parameters, θ = (Q, R, m, α, w, N_v). Parameters Q and R are the number of parent and child clusters, respectively, used in the dictionaries. Parameter m is the number of patches extracted in each X-ray image. Parameter α weights the appearance description and location of the patch. Parameter w is related to the size of the patch. Finally, N_v means the number of visual words of the dictionary that is used to construct the stop-list. See details in [9]. Each parameter was tuned using exhaustive search in order to maximize the performance metrics in the validation set (for $Q = 20, 40 \ldots 120$; $R = 20, 40 \ldots 120$; $m = 60, 80 \ldots 200$; $\alpha = 1, 2 \ldots 12$; $w = 12, 16 \ldots 32$; $N_v = 0, 100, \ldots 500$). The individual best parameters for each classifier were $\theta = (80, 40, 80, 10, 16, 400)$, $(80, 40, 100, 10, 20, 400)$, and $(80, 40, 60, 10, 12, 200)$ for Gun, Shuriken, and Blade, respectively. The computational time was 16 min for training stage and about 0.2 s per testing image on a Mac Mini Server OS X 10.10.1, processor 2.6 GHz Intel Core i7 with four cores and memory of 16GB RAM 1600 MHz DDR3.

5) Deep Learning: In order to provide robustness to rotations, we augmented the dataset by a factor of 12, producing 10 800 training images. We accomplished this by rotating each original training image by 12 angles computed in regular increments from 0° to 330°. For feature extraction from AlexNet, we use the *fc6* layer that experimentally showed better performance than the others (this layer was selected in other computer vision problems as well [70]). In the case of GoogleNet, we achieved a better performance using an inception layer for feature extraction. In particular, our best results were achieved using the layer called *inception_4b/output*. In the KNN approach, we use the nearest neighbor, i.e., $k = 1$. In this experiment, the convolutional networks were pretrained, that means we only need to extract the deep features of the training images: for AlexNet and GoogleNet the computational

[4]All SCI thresholds turn out to be the same result.

time for this task was 5.4 min. For the testing stage, the computational time per testing image was 40 ms for AlexNet and 67 ms for GoogleNet on a simple GPU (GT 730).

6) Baseline Methods: The details of the four classifiers used as baseline methods (SVM, AdaBoost, and SRC) are give. In the case of SVM, we use the LIBSVM library [71] using a linear kernel with default parameters; as for AdaBoost we use a AdaBoost.M2 classifier, with ten iterations and default parameters, using the implementation in the toolbox Balu [69]; for SRC we use a personal implementation based on the SPAMS library[5] with $\lambda = 0, 15$ and other parameters as default. For the baseline methods SVM, AdaBoost, and SRC, the computational time was, respectively, 8, 24, and 1 min for the training stage and about 0.35 s, 0.07 s, and 345 s per testing image on a Mac Mini Server OS X 10.10.1, processor 2.6 GHz Intel Core i7 with four cores and memory of 16GB RAM 1600 MHz DDR3.

IV. Conclusion

This paper attempts to make a contribution to the field of object recognition in X-ray testing by evaluating ten computer vision strategies. The four main contributions of this paper are the following.

1) We proposed a new dataset that can be used in computer vision techniques for X-ray testing in baggage inspection. We defined three threat objects: 1) guns; 2) ninja stars; and 3) razor blades and a negative class (classes Gun, Shuriken, Blade, and Others, respectively). Totally, the dataset has 1950 X-ray images. The dataset is public and can be used for free of charge, for research and educational purposes.
2) We defined an experimental protocol with 900 X-ray images for training, 350 for validation and 700 for testing purposes. In addition, two recognition problems were formulated: 1) four-class classification, in which a classifier must identify one of the four classes (Gun, Shuriken, Blade, and Others) and 2) three detections, in which three detectors must be designed in order to recognize the target classes (Gun, Shuriken, and Blade).
3) We proposed two new computer vision methods that have been developed by integrating well-known computer vision algorithms (see BoW in Section II-A and sparse KNN in Section II-B). In addition, we explained the rest of the methods briefly providing the relevant references for further information.
4) We implemented and evaluated ten computer vision techniques based on classic methods, BoWs, sparse representations, codebooks, and deep features. To the best knowledge of the authors, this is the first experiment in baggage inspection (and probably in X-ray testing) that uses deep learning.

In our experiments, the highest recognition rate was achieved by methods based on visual vocabularies and deep features with more than 95% of accuracy. We strongly believe that it is possible to design an automated aid for the human inspection task using these computer vision algorithms.

As future work, it is possible to consider the integration of shallow and deep learning, i.e., an ensamble of classifiers with different kind of features. In addition, we will train our own CNN using all the images from the $\mathbb{GDX}$ray database. We believe, that a CNN trained with X-ray images (instead of optical images) would lead to better results in X-ray testing. In addition, we will consider new threat objects in our dataset (e.g., knives and sizers).

[5]SPArse Modeling Software available on http://spams-devel.gforge.inria.fr.

References

[1] G. Zentai, "X-ray imaging for homeland security," in *Proc. IEEE Int. Workshop Imaging Syst. Tech. (IST)*, Chania, Greece, Sep. 2008, pp. 1–6.

[2] E. Parliament, "Aviation security with a special focus on security scanners," in *Proc. Eur. Parliament Resolut. (INI)*, Oct. 2012, pp. 1–10.

[3] A. Bolfing, T. Halbherr, and A. Schwaninger, "How image based factors and human factors contribute to threat detection performance in X-ray aviation security screening," in *HCI and Usability for Education and Work* (LNCS 5298), A. Holzinger, Ed. Heidelberg, Germany: Springer, 2008, pp. 419–438.

[4] A. Schwaninger *et al.*, "The impact of image based factors and training on threat detection performance in X-ray screening," in *Proc. 3rd Int. Conf. Res. Air Transp. (ICRAT)*, Fairfax, VA, USA, 2008, pp. 317–324.

[5] G. Blalock, V. Kadiyali, and D. H. Simon, "The impact of post-9/11 airport security measures on the demand for air travel," *J. Law Econ.*, vol. 50, no. 4, pp. 731–755, Nov. 2007.

[6] S. Michel *et al.*, "Computer-based training increases efficiency in X-ray image interpretation by aviation security screeners," in *Proc. 41st Annu. IEEE Int. Carnahan Conf. Security Technol.*, Ottawa, ON, Canada, Oct. 2007, pp. 201–206.

[7] D. Mery, *Computer Vision for X-Ray Testing*. Cham, Switzerland: Springer, 2015.

[8] V. Riffo and D. Mery, "Automated detection of threat objects using adapted implicit shape model," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 4, pp. 472–482, Apr. 2016.

[9] D. Mery, E. Svec, and M. Arias, "Object recognition in baggage inspection using adaptive sparse representations of X-ray images," in *Proc. Pac. Rim Symp. Image Video Technol. (PSIVT)*, Auckland, New Zealand, 2015, pp. 709–720.

[10] I. Uroukov and R. Speller, "A preliminary approach to intelligent X-ray imaging for baggage inspection at airports," *Signal Process. Res.*, vol. 4, no. 4, pp. 1–11, 2015.

[11] D. Turcsany, A. Mouton, and T. P. Breckon, "Improving feature-based object recognition for X-ray baggage security screening using primed visualwords," in *Proc. IEEE Int. Conf. Ind. Technol. (ICIT)*, Cape Town, South Africa, 2013, pp. 1140–1145.

[12] M. Baştan, M. R. Yousefi, and T. M. Breuel, "Visual words on baggage X-ray images," in *Computer Analysis of Images and Patterns*. Heidelberg, Germany: Springer, 2011, pp. 360–368.

[13] N. Zhang and J. Zhu, "A study of X-ray machine image local semantic features extraction model based on bag-of-words for airport security," *Int. J. Smart Sens. Intell. Syst.*, vol. 8, no. 1, pp. 45–64, 2015.

[14] D. Mery, "Inspection of complex objects using multiple-X-ray views," *IEEE/ASME Trans. Mechatronics*, vol. 20, no. 1, pp. 338–347, Feb. 2015.

[15] D. Mery, V. Riffo, I. Zuccar, and C. Pieringer, "Automated X-ray object recognition using an efficient search algorithm in multiple views," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Portland, OR, USA, 2013, pp. 368–374.

[16] V. Riffo and D. Mery, "Active X-ray testing of complex objects," *Insight Non Destruct. Testing Cond. Monitor.*, vol. 54, no. 1, pp. 28–35, 2012.

[17] T. Franzel, U. Schmidt, and S. Roth, "Object detection in multi-view X-ray images," in *Pattern Recognition*. Heidelberg, Germany: Springer, 2012, pp. 144–154.

[18] A. Mouton, G. T. Flitton, S. Bizot, N. Megherbi, and T. P. Breckon, "An evaluation of image denoising techniques applied to CT baggage screening imagery," in *Proc. IEEE Int. Conf. Ind. Technol. (ICIT)*, Cape Town, South Africa, 2013, pp. 1063–1068.

[19] G. Flitton, T. P. Breckon, and N. Megherbi, “A comparison of 3D interest point descriptors with application to airport baggage object detection in complex CT imagery,” *Pattern Recognit.*, vol. 46, no. 9, pp. 2420–2436, Sep. 2013.

[20] N. Megherbi, J. Han, T. P. Breckon, and G. T. Flitton, “A comparison of classification approaches for threat detection in CT based baggage screening,” in *Proc. 19th IEEE Int. Conf. Image Process. (ICIP)*, Orlando, FL, USA, 2012, pp. 3109–3112.

[21] G. Flitton, A. Mouton, and T. P. Breckon, “Object classification in 3D baggage security computed tomography imagery using visual codebooks,” *Pattern Recognit.*, vol. 48, no. 8, pp. 2489–2499, Aug. 2015.

[22] A. Mouton and T. P. Breckon, “Materials-based 3D segmentation of unknown objects from dual-energy computed tomography imagery in baggage security screening,” *Pattern Recognit.*, vol. 48, no. 6, pp. 1961–1978, Jun. 2015.

[23] R. A. Quinn and C. C. Sigl, *Radiography in Modern Industry*. Rochester, NY, USA: Eastman Kodak, 1980.

[24] O. Russakovsky *et al.*, “ImageNet large scale visual recognition challenge,” *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

[25] T. Poggio and S. Edelman, “A network that learns to recognize 3D objects,” *Nature*, vol. 343, no. 6255, pp. 263–266, 1990.

[26] K. Mikolajczyk and C. Schmid, “A performance evaluation of local descriptors,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1615–1630, Oct. 2005.

[27] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[28] H. Bay, T. Tuytelaars, and L. Van Gool, “SURF: Speeded up robust features,” in *Proc. 9th Eur. Conf. Comput. Vis. (ECCV)*, Graz, Austria, May 2006, pp. 404–417.

[29] L. Shen, C.-J. Jiang, and G.-J. Liu, “Satellite objects extraction and classification based on similarity measure,” *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 8, pp. 1148–1154, Aug. 2016.

[30] J. Sivic and A. Zisserman, “Efficient visual search of videos cast as text retrieval,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 4, pp. 591–606, Apr. 2009.

[31] B. Leibe, A. Leonardis, and B. Schiele, “Combined object categorization and segmentation with an implicit shape model,” in *Proc. Workshop Stat. Learn. Comput. Vis. (ECCV)*, Prague, Czech Republic, 2004, pp. 17–32.

[32] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce, “Learning mid-level features for recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, San Francisco, CA, USA, 2010, pp. 2559–2566.

[33] I. Tošić and P. Frossard, “Dictionary learning,” *IEEE Signal Process. Mag.*, vol. 28, no. 2, pp. 27–38, Mar. 2011.

[34] L. Bo, K. Lai, X. Ren, and D. Fox, “Object recognition with hierarchical kernel descriptors,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Colorado Springs, CO, USA, 2011, pp. 1729–1736.

[35] K. Simonyan, O. M. Parkhi, A. Vedaldi, and A. Zisserman, “Fisher vector faces in the wild,” in *Proc. BMVC*, vol. 2. Bristol, U.K., 2013, p. 4.

[36] R. Arandjelovic and A. Zisserman, “All about VLAD,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Portland, OR, USA, 2013, pp. 1578–1585.

[37] J. Wright *et al.*, “Sparse representation for computer vision and pattern recognition,” *Proc. IEEE*, vol. 98, no. 6, pp. 1031–1044, Jun. 2010.

[38] J. Yang, K. Yu, Y. Gong, and T. Huang, “Linear spatial pyramid matching using sparse coding for image classification,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Miami, FL, USA, 2009, pp. 1794–1801.

[39] A. Thomas *et al.*, “Towards multi-view object class detection,” in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, New York, NY, USA, Jun. 2006, pp. 1589–1596.

[40] H. Su, M. Sun, L. Fei-Fei, and S. Savarese, “Learning a dense multi-view representation for detection, viewpoint classification and synthesis of object categories,” in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Kyoto, Japan, 2009, pp. 213–220.

[41] V. Ferrari, T. Tuytelaars, and L. Van Gool, “Simultaneous object recognition and segmentation from single or multiple model views,” *Int. J. Comput. Vis.*, vol. 67, no. 2, pp. 159–188, 2006.

[42] S. Savarese and L. Fei-Fei, “3D generic object categorization, localization and pose estimation,” in *Proc. IEEE 11th Int. Conf. Comput. Vis. (ICCV)*, Rio de Janeiro, Brazil, 2007, pp. 1–8.

[43] J. Liebelt and C. Schmid, “Multi-view object class detection with a 3D geometric model,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, San Francisco, CA, USA, 2010, pp. 1688–1695.

[44] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, “Robust tracking-by-detection using a detector confidence particle filter,” in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Kyoto, Japan, 2009, pp. 1515–1522.

[45] Y. Cheng, R. Cai, X. Zhao, and K. Huang, “Convolutional fisher kernels for RGB-D object recognition,” in *Proc. IEEE Int. Conf. 3D Vis. (3DV)*, Lyon, France, 2015, pp. 135–143.

[46] W. J. Beksi and N. Papanikolopoulos, “Object classification using dictionary learning and RGB-D covariance descriptors,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Seattle, WA, USA, 2015, pp. 1880–1885.

[47] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Y. Ng, “Convolutional-recursive deep learning for 3D object classification,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 665–673.

[48] Ç. Aytekin, Y. Rezaeitabar, S. Dogru, and İ. Ulusoy, “Railway fastener inspection by real-time machine vision,” *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 7, pp. 1101–1107, Jul. 2015.

[49] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[50] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[51] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” unpublished paper, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[52] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[53] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Proc. NIPS*, 2012, pp. 1106–1114.

[54] A. Vedaldi and K. Lenc. (2014). *MatConvNet–Convolutional Neural Networks for MATLAB*. [Online]. Available: http://www.vlfeat.org/

[55] D. Mery *et al.*, “GDXray: The database of X-ray images for nondestructive testing,” *J. Nondestruct. Eval.*, vol. 34, no. 4, pp. 1–12, 2015.

[56] C.-F. Tsai, “Bag-of-words representation in image annotation: A review,” *ISRN Artif. Intell.*, vol. 2012, 2012, Art. no. 376804.

[57] M. Heikkilä, M. Pietikäinen, and C. Schmid, “Description of interest regions with local binary patterns,” *Pattern Recognit.*, vol. 42, no. 3, pp. 425–436, 2009.

[58] J. A. Hartigan and M. A. Wong, “Algorithm as 136: A K-means clustering algorithm,” *J. R. Stat. Soc. C (Appl. Stat.)*, vol. 28, no. 1, pp. 100–108, 1979. [Online]. Available: http://www.jstor.org/stable/2346830

[59] A. Liaw and M. Wiener, “Classification and regression by randomforest,” *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[60] A. Dixit. (2014). *Adaptive Kmeans Clustering for Color and Gray Image*. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/45057-adaptive-kmeans-clustering-for-color-and-gray-image Matlab Central, File Exchange, Jan. 2014.

[61] T. Rückstieß, C. Osendorfer, and P. van der Smagt, “Sequential feature selection for classification,” in *AI 2011: Advances in Artificial Intelligence* (LNCS 7106), D. Wang and M. Reynolds, Eds. Heidelberg, Germany: Springer, 2011, pp. 132–141. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-25832-9_14

[62] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, “Online learning for matrix factorization and sparse coding,” *J. Mach. Learn. Res.*, vol. 11, pp. 19–60, Mar. 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1756006.1756008

[63] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, “Robust face recognition via sparse representation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, Feb. 2009.

[64] B. Leibe, A. Leonardis, and B. Schiele, “Robust object detection with interleaved categorization and segmentation,” *Int. J. Comput. Vis.*, vol. 77, nos. 1–3, pp. 259–289, 2008.

[65] C. Szegedy *et al.*, “Going deeper with convolutions,” in *Proc. CVPR*, Boston, MA, USA, 2015, pp. 1–9.

[66] R. Arandjelović and A. Zisserman, “Three things everyone should know to improve object retrieval,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Providence, RI, USA, Jun. 2012, pp. 2911–2918.

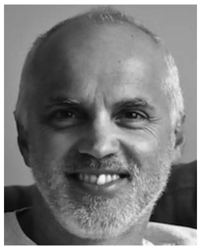
[67] U. M. Fayyad and K. B. Irani, “The attribute selection problem in decision tree generation,” in *Proc. AAAI*, San Jose, CA, USA, 1992, pp. 104–110.

[68] A. Vedaldi and B. Fulkerson. (2008). *VLFeat: An Open and Portable Library of Computer Vision Algorithms*. [Online]. Available: http://www.vlfeat.org/

[69] D. Mery. (2011). *BALU: A Matlab Toolbox for Computer Vision, Pattern Recognition and Image Processing*. [Online]. Available: http://dmery.ing.puc.cl/index.php/balu

[70] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, "Neural codes for image retrieval," in *Computer Vision—ECCV 2014* (LNCS 8689), D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham, Switzerland: Springer, 2014, pp. 584–599.

[71] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, p. 27, 2011. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm



Domingo Mery (M'01) received the M.Sc. degree in electrical engineering from the Technical University of Karlsruhe, Karlsruhe, Germany, in 1992, and the Ph.D. (with distinction) degree from the Technical University of Berlin, Berlin, Germany, in 2000.

He was a Research Scientist with the Institute for Measurement and Automation Technology, Technical University of Berlin with the collaboration of YXLON X-Ray International. He has received scholarships from the Konrad Adenauer Foundation and the German Academic Exchange Service (DAAD). In 2001, he served as an Associate Researcher with the Department of Computer Engineering, Universidad de Santiago Chile, Santiago, Chile. During 2014 he was a Visiting Professor with the University of Notre Dame, Notre Dame, IN, USA. He is currently Full Professor with the Department of Computer Science, Pontificia Universidad Católica de Chile, Santiago, where he served as the Chair from 2005 to 2009. He has authored 65 technical SCI publications and over 75 conference papers. His current research interests include image processing for fault detection in aluminum castings, X-ray imaging, real-time programming, and computer vision.

Dr. Mery was a recipient of the Ron Halmshaw Award in 2005 and 2012, the John Green Award in 2013 from the British Institute of Non-Destructive Testing, which was established to recognize the best papers published in the Insight Journal on Industrial Radiography, and the Best Paper Award at the International Workshop on Biometrics in conjunction with the European Conference on Computer Vision (ECCV 2014). He is the Local Co-Chair of ICCV2015, Santiago. He served as the General Program Chair of Pacific-Rim Symposium on Image and Video Technology (PSIVT2007), the Program Chair of PSIVT2009, and the General Co-Chair of PSIVT2011 and the 2007 Ibero-American Congress on Pattern Recognition. He is currently Associate Editor of IEEE TRANSACTIONS OF INFORMATION, FORENSICS AND SECURITY.

Erick Svec received the B.S. and M.S. degrees in computer science from the Catholic University of Chile, Santiago, Chile, in 2016.

His current research interests include pattern recognition and computer vision for X-ray testing.

Marco Arias received the B.S. and M.S. degrees in computer science from the Catholic University of Chile, Santiago, Chile, in 2014 and 2016, respectively.

His current research interests include pattern recognition, machine learning, and computer vision for X-ray testing.

Vladimir Riffo received the B.Sc.Eng. degree in electronic engineering from the Universidad de Antofagasta, Antofagasta, Chile, in 1998, and the M.Eng. degree from the Pontificia Universidad Católica de Chile (PUC), Santiago, Chile, in 2011, where he is currently pursuing the Doctoral degree.

He is an Associate Professor with the Department of Computer Engineering and Computer Science, Universidad de Atacama, Copiapó, Chile. He is a Fellow of GRIMA, the Machine Intelligence Group, PUC. His current research interests include pattern recognition, object detection, computer vision using multiviews and X-ray testing, and the ways to combine those approaches.

Mr. Riffo was a recipient of a scholarship from the Comisión Nacional de Investigación Científica y Tecnológica and the Ron Halmshaw Award in 2012 and the John Green Award in 2013 from the British Institute for Non-Destructive Testing, which was established for the best papers published in Insight Journal on Industrial Radiography.

Jose M. Saavedra received the M.Sc. degree in computer science from the Universidad Nacional de Trujillo, Trujillo, Peru, and the Ph.D. degree in computer science from the University of Chile, Santiago, Chile, in 2013.

He is currently a Computer Vision Researcher with Orand, Santiago, and co-founder at Impresee, Santiago. He is also a part-time Professor with the Department of Computer Science, University of Chile. His current research interest includes pattern recognition, computers vision, and machine learning.

Sandipan Banerjee received the B.S. degree from the National Institute of Technology, Durgapur, India, in 2012. He is currently pursuing the Ph.D. degree with the University of Notre Dame, Notre Dame, IN, USA.

His current research interests include deep learning-based face recognition, artificial face synthesis, and synthetic facial aging.