SPARSE KNN – A METHOD FOR OBJECT RECOGNITION OVER X-RAY IMAGES USING KNN BASED IN SPARSE RECONSTRUCTION

ERICK SVEC P.

Thesis submitted to the Office of Research and Graduate Studies in partial fulfillment of the requirements for the degree of Master of Science in Engineering

Advisor:
DOMINGO MERY QUIROZ

Santiago de Chile, August 2016

© MMXV, ERICK VACLAV SVEC PARRA
SPARSE KNN – A METHOD FOR OBJECT RECOGNITION OVER X-RAY IMAGES USING KNN BASED IN SPARSE RECONSTRUCTION

ERICK SVEC P.

Members of the Committee:
DOMINGO MERY QUIROZ
KARIM PICHARA BAKSAI
SERGIO HERNÁNDEZ ÁLVAREZ
CRISTIÁN ANDRÉS TEJOS NÚÑEZ

Thesis submitted to the Office of Research and Graduate Studies in partial fulfillment of the requirements for the degree of Master of Science in Engineering

Santiago de Chile, August 2016
To my Mother, Father and Sisters,
for encouraging me and their
unconditional support
ACKNOWLEDGEMENTS

First of all, I want to show gratitude to my parents: Wenceslao and Alicia Svec in addition to my sisters Katiushka and Valeshka Svec for walking this path by my side always with loving support. With appreciation, I share my gratitude to Domingo Mery who taught me not only of computer vision, but has been a true companion and guide during my learning period and thesis development. I also wish to give thanks to Tomás Larrain and Marco Arias for numberless conversations and ideas while we develop our thesis. Finally, thanks to Rocio Escobedo for her support during this last period.

"Now ye may suppose that this is foolishness in me; but behold I say unto you, that by small and simple things are great things brought to pass; and small means in many instances doth confound the wise."

—ALMA, About 74 B.C.
# TABLE OF CONTENTS

ACKNOWLEDGEMENTS ................................................................. iv

LIST OF FIGURES ................................................................. vii

LIST OF TABLES ................................................................. ix

ABSTRACT ................................................................. x

RESUMEN ................................................................. xi

1. INTRODUCCION ................................................................. 1
   1.1. The problem ......................................................... 1
   1.2. State of the Art .................................................... 2
   1.3. Motivation .......................................................... 4
   1.4. Known Methods .................................................. 5
       1.4.1. k-Nearest Neighbors (knn) .......................... 5
       1.4.2. Sequential Forward Selection (SFS) .............. 6
       1.4.3. SIFT ........................................................ 7
       1.4.4. K-means .................................................... 8
       1.4.5. Sparse Reconstruction ................................. 9

2. PROPOSED METHOD ........................................................... 11
   2.1. Training ........................................................... 12
       2.1.1. Feature Extraction ................................... 12
       2.1.2. Delta Feature .......................................... 13
       2.1.3. SFS ....................................................... 14
       2.1.4. Kmeans .................................................. 16
       2.1.5. Dictionary Construction ............................ 17
   2.2. Testing .......................................................... 17
       2.2.1. Feature Extraction ................................... 18
2.2.2. Classifiers ........................................... 18
2.2.3. SoftVoting ............................................. 21

3. Experiments and Results ........................................ 23
  3.1. DataBase .................................................. 23
      3.1.1. Training Images ..................................... 23
      3.1.2. Testing Images ........................................ 24
  3.2. Experiments ................................................. 26
      3.2.1. Parameters adjustment ................................. 26
  3.3. Curves .......................................................... 31
  3.4. Results ......................................................... 35
      3.4.1. Validation Performance ................................. 35
      3.4.2. Comparison of different methods ...................... 36

4. CONCLUSION .................................................... 39

References .......................................................... 41
LIST OF FIGURES

1.1 Knn Classification .................................................. 6
1.2 K-means Clustering .................................................. 8

2.1 Method Overview ...................................................... 11
2.2 Training Diagram ..................................................... 12
2.3 Morphological Processing Example ............................... 13
2.4 Representation of what the \( \delta \) feature represents ........ 14
2.5 SIFT visual representation in a Shuriken Image ............... 15
2.6 Normalized sift gradient representation .......................... 16
2.7 Abundant similar keypoints in the same object ............... 16
2.8 Testing Diagram ....................................................... 17
2.9 Visualization of feature selection ................................. 19
2.10 Example SCI threshold ............................................. 20

3.1 Example of training set ............................................. 24
3.2 A gun rotating as an example of rotation of the objects .... 24
3.3 Example of testing images .......................................... 25
3.4 Example of testing Set ................................................ 25
3.5 Jmax vs number of features for SFS .............................. 27
3.6 Clean process of noisy keypoints .................................. 28
3.7 Accuracy Vs DistanceTresholdPistol ......................... 29
3.8 Accuracy Vs DistanceTresholdRazor ...................... 30
3.9 Accuracy Vs DistanceTresholdShuriken .................... 30
3.10 Accuracy Vs minMaxDistance .................................... 31
3.11 Precision-Recall Curves varying SCI parameter ........... 32
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Precision-Recall Curves variating <em>distanceTresholdPistol</em> parameter</td>
<td>33</td>
</tr>
<tr>
<td>3.1 Precision-Recall Curves variating <em>distanceTresholdShuriken</em> parameter</td>
<td>33</td>
</tr>
<tr>
<td>3.1 Precision-Recall Curves variating <em>distanceTresholdRazor</em> parameter</td>
<td>34</td>
</tr>
<tr>
<td>3.1 Precision-Recall Curves variating <em>minMaxDistance</em> parameter</td>
<td>35</td>
</tr>
<tr>
<td>3.1 Confusion Matrix</td>
<td>38</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>3.1</td>
<td>Distribution of images among all sets</td>
</tr>
<tr>
<td>3.2</td>
<td>Precision &amp; Recall for validation set</td>
</tr>
<tr>
<td>3.3</td>
<td>Accuracy for each detector in validation set (in percentages) in experiment (B)</td>
</tr>
<tr>
<td>3.4</td>
<td>Precision &amp; Recall for each detector</td>
</tr>
<tr>
<td>3.5</td>
<td>Accuracy for each detector</td>
</tr>
</tbody>
</table>
ABSTRACT

Object recognition in X rays images is not a trivial task. There is dealing with fundamental problems such as the absence of color or the particular occlusion inherent of this problem, the second being different of what we are accustomed, inasmuch as in radiography the object is always visible, as much as any other object present in the image at all times, therefore, more than obstruct the image visibility, X-Rays combine them. It is for this purpose we introduce a method based on SIFT keypoints extracted from the image. These keypoints are first to be used to train a model for each present class, even Pistol, Shuriken, Razor and Others or No-class. Then, using this model, we calculate the sparse reconstruction of these keypoints, these are then classified by a metric distance, training a different classifier for each class. A key stage of our method is feature selection, we select around 50 features of the 128 SIFT keypoints features, being this selection different for each classifier. The purpose for this selection is to diminish the added noise of the radiologic occlusion before mentioned. It is empirically proved the parameters tunings importance, which take a great importance by raising the algorithms performance, among these tuning parameters are found, acceptance distances threshold, Sparsity Concentration Index threshold, being this an index used to know if the sparse representation of one or many classes; and a representation of the importance of the relative keypoint center of the image position delta. The effectiveness of our method was tested in the recognition of three different threat objects: pistols, shuriken (ninja stars) and razor blades. With above, in our experiments, after parameter tuning, the highest recognition rate achieved is over 94% accuracy.

Keywords: sparse, rayos X, computer vision, object recognition, KNN, SCI, SIFT
RESUMEN

El reconocimiento de objetos en imágenes de rayos X no es una tarea trivial, lidiando con problemas fundamentales tales como la ausencia de color o la oclusión particular del problema, diferenciando esta segunda de la oclusión con la que generalmente nos encontramos, puesto que en una radiografía el objeto siempre es visible, así como cada otro objeto en la imagen, por lo que más que obstruir la visión, las imágenes de rayos X las combinan. Es por esto que se presenta un método basado en los puntos SIFT extraídos de la imagen. Estos puntos son primeramente usados para entrenar un modelo para cada clase presente, Pistola, Shuriken, Hoja de afeitar y Otros o No clase. Luego, utilizando este modelo, se busca la reconstrucción sparse de estos puntos, los cuales son clasificados utilizando una métrica de distancia, construyendo un clasificador por cada clase. Una fase clave en este método es la selección de características, seleccionamos alrededor de 50 características de las 128 del descriptor SIFT, esta selección diferente para cada clasificador. La finalidad de esta selección es disminuir el ruido agregado por la oclusión radiológica antes explicada. Se demuestra empíricamente la importancia del ajuste de parámetros, los cuales toman una gran importancia para aumentar el rendimiento del algoritmo, entre los parámetros a ajustar se encuentran, umbrales de aceptación de distancia, umbral sobre el Sparsity Concentration Index, siendo este un índice usado para saber si la reconstrucción sparse es representativa de una o varias clases; y una representación de la importancia de posición relativa al centro del objeto a cada punto SIFT encontrado. La efectividad de nuestro método fue probada reconociendo tres objetos peligrosos diferentes: pistolas, shuriken (estrellas ninja) y cuchillas de afeitar. Con esto, en nuestros experimentos, se obtienen resultados sobre un 94% de precisión.

Palabras Claves: sparse, rayos X, visión por computación, reconocimiento de objetos, KNN, SCI, SIFT
1. INTRODUCCION

Security by visual inspection has always been a used method. Since unmemorable times, it has been common to detect dangerous objects in a bag or suitcase at the airport gate. In this case, an agent must check the X-Ray image of this bag and visually detect objects that were previously tagged as dangerous. This methodology, trusting human factor, has its pros and cons. The human brain can perform image process really fast in detection and recognition tasks among others. However, there are many common external factors which distract us. It is an unexpected noise, small talk from a colleague or merely being exhausted from a long shift which lowers considerably the performance. This is a main reason why researchers have developed different techniques in the computer vision area to perform object detection and recognition automatically. Each method pursues to find the best representation for the object. For example, by shape [4, 9], by color [24], by points of interest [3], by model part [12], etc.

1.1. The problem

X-Ray security at airports and others alike are based on human visual inspection. Therefore, performance is variable; mainly due to human related factors [51]. A persons concentration is variable during the day as well. People may be more receptive during the morning hours than after lunch when are less alert and sleepy. Human performance also may vary on more extended periods. For example, during the week if an individual sleeps less than eight hours; then, they will experience sleep deprivation and their performance will decay [2]. It’s for this reason that a support computer system has been introduced in order to compensate for the decay of human performance due to external factors. These systems are not to replace the security agent but rather to help him with object recognition via visual inspection.

Radiological images bring their own difficulties, such as the variance in the number of objects present in one image or the absence of these, or the different rotations of the image and/or the occlusion particular for X-Ray images. The variance in the number is because
most baggage won’t contain any dangerous objects, but bags with objects, most likely will contain more than one. Another difficulty is all the possible rotations and scales of the object in the image. This is not only for different angles on the view, but also because objects rotate, their projected shape changes in the image, creating self-occlusion.

We needed to use a particular occlusion approach for X-Ray images, because different from normal images where the object is present completely, partially occluded or absent; in X-Ray images, the object is always present. Not just the object is present always, but all objects are, this because the image is an addition function of all non-transparent objects, depending of their density and composition. This occlusion brings a hinder the isolation of an objects of interest from the background or from other objects.

1.2. State of the Art

X-Ray image detection has been used in numerous applications, among them: for medical purposes such as [53], food industry [19], find welding defects [8, 60] and some other less common as human identification [18]. For Object recognition X-Ray images are helpful when normal inspection can penetrate one or more layer without manipulating the environment. Scenarios where we is not possible to open a suitcase or cutting open a tube to check its welding quality. However, using X-Ray images also has its disadvantages introduced for problems like occlusion, grayscale images, generally low resolution, etc.

Baggage inspection using X-Ray screening is a priority task that reduces the risk of crime, terrorist attacks and propagation of pests and diseases [58]. Security and safety screening with X-Ray scanners has become an important process in public spaces and at border checkpoints [41]. 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 [6]. Manual detection of threat items by human inspectors is extremely demanding [50]. 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 substances) takes a great
deal of concentration. In addition, human inspectors receive only minimal technological support. Furthermore, during rush hours, they have only a few seconds to decide whether or not a bag contains a threat item [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% [38]. In baggage inspection, automated X-Ray testing remains an open question due to: i) generality lost, which means that approaches developed for one task may not transfer well to another; ii) deficient accuracy detection, which means that there is a fundamental tradeoff between false alarms and missed detections; iii) limited robustness given that requirements for the use of a method are often met for simple structures only; and iv) low adaptiveness in that it may be very difficult to accommodate an automated system to design modifications or 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 [29]. For this research proposal, 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 based on visual codebooks [46]) and dual-energy X-Ray images (e.g., Gabor texture features [56], bag of words based on SURF features [55] and pseudo-color, texture, edge and shape features [59]). More complex approaches that deal with multiple X-Ray images have been developed as well. In the case of mono-energy imaging, see for example the recognition of regular objects using data association in [30] and active vision [45] where a second-best view is estimated. In the case of dual-energy imaging, see the use of visual vocabularies and SVM classifiers in [15]. Progress also has been made in the area of computed tomography (CT). For example, in order to improve the quality of CT images, metal artifact reduction and de-noising [39] techniques were suggested. Many methods based on 3D features for 3D object recognition have been developed (see, for example, RIFT and SIFT descriptors [13], 3D Visual Cortex Modeling 3D Zernike descriptors and histogram of shape index [28], Adaptive Implicit Shape Model
(AISM), which was presented originally in [47] by Riffo et al and which shares similarities with our method). There are contributions using known recognition techniques (see, for example, bag of words [14] and random forest [40]). As we can see, the progress in automated baggage inspection is modest and still very limited compared to what is needed (ideally automated with no human interaction) as a support for this we can see that until this day, X-Ray screening systems are mostly based in human inspectors. Automated recognition in baggage inspection is far from being perfected given that the appearance of the object of interest can become extremely difficult due to problems of self-occlusion, noise, acquisition, clutter, high intra-class variability, etc.

1.3. Motivation

We believe that algorithms based on sparse representations can be used for this general task because in many computer vision applications (and under the assumption that natural images can be represented using sparse decomposition) state-of-the-art results have been significantly improved [54]. Thus, it is possible to cast the problem of recognition into a supervised recognition form with X-Ray images and class levels (e.g., objects to be identified) using learned features in a unsupervised way. In the sparse representation approach, a dictionary is built from the training X-Ray images, and matching is done by reconstructing the query image using a sparse linear combination of the dictionary. Usually, the query image is assigned to the class with the minimal reconstruction error.

Reflecting on the problems confronting recognition of objects, we believe that there are some key ideas that should be present in new proposed solutions. First, it is clear that certain parts of the objects are not providing any information about the class to be recognized (for example occluded parts). For this reason, such parts should be detected and should not be considered by the recognition algorithm, avoiding noisy features and improving processing time. Second, in recognizing any class, there are parts of the object that are more relevant than other parts (for example the sharp parts when recognizing sharp objects like knives). For this reason, relevant parts should be class-dependent, and could be found using unsupervised learning. Third, in the real-world environment, and given that
X-Ray images are not perfectly aligned and the distance between detector and objects can vary from capture to capture, analysis of fixed parts can lead to misclassification. For this reason, feature extraction should not be in fixed positions, and can be in several random positions. Moreover, it would be possible to use a selection criterion that enables selection of the best regions. Fourth, an object that is present in a query image can be subdivided into ‘sub-objects’, for different parts (e.g., in case of a handgun there are trigger, muzzle, grip, etc.). For this reason, when searching for images of the same class it would be helpful to search for image parts in all images of the training images instead of similar training images.

Inspired by these key ideas, we propose a method for objects recognition using single view in X-Ray bags images.

1.4. Known Methods

In this section we introduce the base methods used in the papers mentioned before, and a few others that we well use later on.

1.4.1. k-Nearest Neighbors (knn)

In pattern recognition, the k-Nearest Neighbors algorithm (knn) is a non-parametric method used for classification and regression. In our case, used for classification, where the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor [21].
FIGURE 1.1. Example of knn classification of 200 samples classified with a 92% accuracy using a knn trained model with $k = 10$ and 800 samples equally divided into two classes and two features. The first image shows the training samples, in second image we can see the ideal classification of testing samples and the third image shows the classification of knn.

1.4.2. Sequential Forward Selection (SFS)

Feature selection is the process of selecting a subset of features, ideally the most discriminant of the general set. This reduces dimensionality, should improve accuracy in the classification and reduce performance time. A common method of feature selection is sequential feature selection. This method has two components: First: An objective function, called the criterion, which the method seeks to minimize over all feasible feature subsets. Common criteria are mean squared error (for regression models) and misclassification rate (for classification models) or in our case, knn. The evaluation term is calls $J_{\text{max}}$ and correspond to the objective function result in a given time while running the algorithm, for example with knn, it would be the accuracy obtained with the current features selected. Second: A sequential search algorithm, which adds or removes features from a candidate subset while evaluating the criterion. Since an exhaustive comparison of the criterion value at all $2^n$ subsets of an n-feature data set is typically infeasible (depending on the size of n and the cost of objective calls), sequential searches move in only one direction, always growing or always shrinking the candidate set. In Sequential forward selection (SFS) [49], features are sequentially added to an empty candidate set until the addition of further features does not decrease the criterion or all features are selected [48].
1.4.3. SIFT

Scale Invariant Feature Transform (SIFT) is an image descriptor for image-based matching and recognition developed by David Lowe [23]. This descriptor, as well as related image descriptors, are used for a large number of purposes in computer vision related to point matching between different views of a 3-D scene and view-based object recognition. It consists in finding representative points (such as edges, borders, blobs, etc.) in images called keypoints. The basic steps of this algorithm are: Localize keypoints by Scale-space extrema detection, although this produces too many keypoint candidates, some of which are unstable. The next step in the algorithm is to perform a detailed fit to the nearby data for accurate location, scale, and ratio of principal curvatures. This information allows points to be rejected that have low contrast (and are therefore sensitive to noise) or are poorly localized along an edge. Then, for each keypoint:

(i) Magnitude and gradient orientation are calculated around keypoint

(ii) A histogram is created where it’s maximum determinates the keypoint orientation

(iii) If the histogram contains a peak over 80 % of their max, a new keypoint is generated for each peak with corresponding orientation

(iv) A Neighborhood is segmented in regions of $4 \times 4$ pixels

(v) For each region a histogram of gradient orientation is calculated using a Gaussian weighted function of 4 pixels

(vi) Each pixel contributes to all his neighbors using a weight $1 - d$, where $d$ is the distances from the keypoint to the center of the image

(vii) Then, a feature histogram is computed, using 8 principal directions, we have $4 \times 4 \times 8 = 128$ vectors

(viii) Repeat previous step for each keypoint
1.4.4. K-means

K-Means [17] is a clustering algorithm. Its purpose is to partition a set of vectors into K groups that cluster around common mean vector. This can also be thought as approximating each of the input vector with one of the means, so the clustering process finds, in principle, the best dictionary or codebook to vector quantize the data. This is done by the Lloyd algorithm, a method that alternates between optimizing the cluster centers and the data-to-center assignments. Once this process terminates, the matrix centers contains the cluster centers and the vector assignments of the input data to the clusters. The cluster centers are also called means because it can be shown that, when the clustering is optimal, the centers are the means of the corresponding data points. The cluster centers and assignments can be visualized in Figure 1.2:

![Figure 1.2. K-means clustering of 5000 randomly sampled data points. The black dots are the cluster centers. Image extracted from: (http://www.vlfeat.org/demo/kmeans_2d_rand.jpg)](http://www.vlfeat.org/demo/kmeans_2d_rand.jpg)
1.4.5. Sparse Reconstruction

Sparse methodology can be divided into sparse dictionary, sparse representation $x$ of a given signal $y$, and sparse reconstruction $\hat{y}$ of the signal using the dictionary and the sparse representation. Constructing a sparse dictionary $D$ can be achieved by different techniques of minimization, for example, as explained in [26], given a training set $y^1, ..., y^n$. It aims to solve equation 1.1

$$\min_{D \in C} \lim_{n \to +\infty} \frac{1}{n} \sum_{i=1}^{n} \min_{x^i} \left( \frac{1}{2} \| y^i - Dx^i \|_2^2 + \psi(x^i) \right)$$  \hspace{1cm} (1.1)$$

where $\psi$ is a sparsity-inducing regularizer constant and $C$ is a constraint set for the dictionary. As shown in [26], various combinations can be used for $\psi$ and $C$ for solving different matrix factorization problems. Even more, positivity constraints can be added to $x$ as well. The function admits several modes for choosing the best settings by optimizing the parameters presented in [26], or using the parameter-free strategy proposed in [25]. But in our case we use the method introduced for Wright [57], where the dictionary is constructed as a concatenation of feature vectors and not calculated as a minimization problem.

For sparse representation several methods are proposed, among them Orthogonal Matching Pursuit algorithm (or forward selection) [27], K-SVD [1] and Lasso, a fast implementation of LARS [11]. Lasso is the one chosen for us, it also solves a minimization problem where, given a matrix of signals $Y = [y^1, ..., y^n]$ in $\mathbb{R}^{m \times n}$ and a dictionary $D$ in $\mathbb{R}^{m \times p}$, the algorithm returns a matrix of coefficients $X = [x^1 ... x^n]$ in $\mathbb{R}^{p \times n}$, where that for every column $y$ of $Y$, the corresponding column $x$ of $X$ is the solution of

$$\min_{x \in \mathbb{R}^p} \| y - Dx \|_2^2 \quad s.t. \quad \| x \|_1 \leq \lambda$$  \hspace{1cm} (1.2)$$

Sparse reconstruction $\hat{y}$ is much simpler to compute and can be achieved by the following equation

$$\hat{y} = \| y - Dx \|_2^2$$  \hspace{1cm} (1.3)$$
the previous can be summarize as a method to retrieve the sparse reconstruction \( \hat{y} \) of \( y \) as a linear combination \( x \) of the individual or feature vectors storage in \( D \). With this, the metric SCI introduced in [57] and defined as

\[
SCI(y) = \frac{nc \cdot \max_i \|\Pi_i(x)\|_1/\|x\|_1 - 1}{nc - 1} \in [0, 1] \quad (1.4)
\]

where \( nc \) is the number of classes and \( \Pi_i(x) \) is a function that leaves \( x \) coefficients that correspond to all classes but the \( i \) class to 0. As mentioned in [43] this can be used as a image quality measure, as for SCI takes values between 0 and 1. SCI values close to 1 correspond to the case where the test image can be approximately represented by using only images from a single class. The test vector has enough discriminating features of its class, so has high quality. If \( SCI(y) = 0 \), then the coefficients are spread evenly across all classes. So the test vector is not similar to any of the classes and is of poor quality. A threshold can be chosen to reject the images with poor quality. For instance, a test image can be rejected if \( SCI(\hat{y}) \leq SCI_{\text{threshold}} \) and otherwise accepted as valid, where \( SCI_{\text{threshold}} \) is some chosen threshold between 0 and 1.

Even if we do not use the residual \( r_i(\hat{y}) \) metric, it is widely used as a classification metric in other works, for example in [33], and therefore it is of worth to introduced it. The residual represents the reconstruction error of the original signal with the sparse calculated, and follows the equation 1.5.

\[
r_i(y) = \|y - D\delta_i(x)\|_2^2 \quad (1.5)
\]
2. PROPOSED METHOD

With everything mentioned before, we introduce a new method based in nearest neighbors classification over Sparse reconstruction. As many methods we can divide this in three stages: first, an offline training stage where we obtain one model for each class; then another offline stage called validation comes, in which stage parameters for each model are tuned; and finally the third, an online stage, testing, where for each query image the predicted class is obtained by the votes of each model. Training and Testing stages are represented in figure 2.1. These two stages are explained most thoroughly on this section and validation protocol is mentioned in section 3.2.1.
2.1. Training

As mentioned before, in training stage, for each class in training set except for Other class, a model is built. This training consist of 4 steps: feature extraction based on SIFT keypoints, SFS feature selection (including the other classes and their labels), calculating a $\delta$ feature, Kmeans clustering of the class training set. This is explained in detail in the following sections.

2.1.1. Feature Extraction

In training set, images were taken without background objects (isolated object over white background) and, therefore, contain an isolated object, making segmentation an easier task. Then, for each image, the object is segmented using an adaptive kmeans clustering for grayscales images implemented by Ankit Dixit [10]. The X-Ray of different objects could have a high variation in the number of grey-values, so a traditional kmeans with a fixed $k$ will give less accurate results if the $k$ is larger than the amount of representative greys values for a given object. This is an adaptive method, there is no need to choose the number of clusters. As the object are darker because they absorb higher amounts of
X-Rays, only the cluster corresponding to the clearer pixels is selected as background. Background pixels are assigned as 1 and pixels from all other clusters are considered as the object, assigning them as 0. Then, an morphological transformation is applied dilating the image and closing all holes in the selected object obtaining in this manner a mask $I_{bw}$ as shown in figure 2.3. Then, SIFT keypoints are extracted over the entire image and all keypoints outside this mask are rejected and erased, avoiding noisy keypoints.

![Figure 2.3](image)

**Figure 2.3.** The first image is the original X-Ray object, the second image shows the clusters using adaptive kmeans, the third image shows the mask obtained after the dilation and the fourth image shows the selected keypoints.

As the result of this we will have a matrix $F_i$ $128 \times r$ where $r$ is the number of accepted keypoints in all training images and $i$ is the class being trained, a $d_{train}$ vector also is obtained, this vector contains the class label for each SIFT keypoint in $F_i$, where this class is the class of the image where the keypoints where extracted.

### 2.1.2. Delta Feature

Part models not always capture the shape of the object, but rather the shape of the model parts. We might encounter an object that shares a few keypoints with one of our classes, but they don’t share the same positions within the image, then we can doubt it is the same object. This is why we introduce one more feature that intends to capture position relative to the center of the image by calculating the distance $d$ from the center of the image to the center of the keypoint (see figure 2.4), and then, normalize the $d$ using the width $w$ and high $h$ of the image.

$$
\delta = \frac{d}{w + h}
$$
and it is concatenated to the keypoint features, then, including SFIT descriptor, our feature will be a vector of $129 \times r$ and $F_i$ will turn into $F'_i$ of $129 \times r$.

![Figure 2.4](image)

**Figure 2.4.** Each green arrow represent the distance used to calculate the $\delta$, this is the distance between keypoint and center of the image.

As the min value of $d$ is 0 when the keypoint is in the center of the image we have that $\delta_{min} = \frac{0}{w+h} = 0$, and max value of $d$ is given when the keypoint is in the corner of the image, then $\delta_{max} = \frac{\sqrt{w^2+h^2}}{2(w+h)}$ and as $w, h \in \mathbb{N}$, then $\delta_{max} \leq 1$. With this we have that $\delta \in [0, 1]$.

### 2.1.3. SFS

An offline feature selection is done selecting the most discriminant features for each class. This is made using Search Forward Selection feature Selection (SFS) [49]. The prior computed features $F'_i$ are concatenated and with the label vector $d_i$ where $d_i$ has the same length of $d_{train}$ and its values are 1 when $d_{train}$ values are $i$ and 0 in other cases, and are used as input, ending up with one different feature selection for each class against all others. The method used to evaluate features in SFS is knn with $k = 5$, this is to be closer to the final classifier. Then, using $J_{max}$ criteria, features are selected when there are no more significative improvement as shown in figure 3.5.
The idea behind this feature selection, is mainly, to deal with our particular occlusion. Due to our training images contain only the object and no other background artifacts, the idea is to keep only the features corresponding to gradients that will remain equal or very similar when this background is present. In figure 2.5 we can see the representation of a keypoint, it is clear at our eye that when we add background to these images, there will be gradients that will tremendously variate, adding noise to our classifier, SFS deals with this problem by selecting the best features for this problem. It is clear that not all keypoints will share the same gradients and that is why SFS representation might not make sense at when we first think about it, but if we take a look at figure 2.6 with all gradients as 1 we will see a sort of background subtraction by feature selection. It is fair to remember that SFS is applied to the entire class and not for each part of the object, it is for Shuriken, not one for the shark and other for corners and so forth. We choose SFS due to the intensity of the data, our feature selection method must be considerable fast and shouldn’t consume much resources. we also tried with feature transformation methods, like PCA [16, 42] but accuracy was under 50%. As the result of this, our matrix $F_{i}$ we will be reduce to $S_{i}$ $s \times r$ where $s$ is the number of features selected with SFS.
2.1.4. Kmeans

A kmeans algorithm is then used to deal with keypoints pseudo-duplications, as seen in image 2.7, and considering the invariance of SIFT, many similar keypoints are selected. The idea is to bring all keypoints that are similar into the same cluster, reducing the number of selected keypoints and avoiding duplicity in training data, obtaining the feature matrices $M_{i,1}$ and $M_{i,0}$ corresponding to the centroids of cluster processed for the $i$ class stored at $M_{i,1}$ and the centroids of the cluster that don’t belong to $i$ class stored at $M_{i,0}$.
2.1.5. Dictionary Construction

Then, when all classes had gone through the prior process, the resulting matrices of features $M_{i,1}$ and $M_{i,0}$ for each class are concatenated to form a dictionary $D_i$. For each class $i$, a binary vector $d_{train,i}$ is obtained, which contains the labels for each row of $D_i$, whether belongs or not to $i$ class. Also a vector $d_{c,train,i}$ is obtained, which contains the class labels for each row of $D_i$. This last vector is used for SCI purposes (see 1.4) and is a parameter used in Testing stage (section 2.2.2.3).

2.2. Testing

\[ predictedClass = SoftVoting(\{v_1, p_{c_1}\}, \ldots, \{v_j, p_{c_j}\}, \ldots, \{v_m, p_{c_m}\}) \]

**Figure 2.8.** Diagram shows the method overview for testing stage

Analogously to training stage, there will be one binary classifier for every class, query image $I_q$ will not enter directly to the classifier, but rather all query vector $q_j$, with $j$ is the number of keypoints extracted from query image (detailed in the following sections). Each one of these classifiers will have a binary output, 1 if the query vector is from the given class and 0 if not, and a scalar measure used to predict the class using soft-voting. With this, for a query image, the procedure is as presented.
2.2.1. Feature Extraction

Feature extraction in this section does not include the segmentation presented in training stage, due to background present in images and other artifacts altogether with occlusion, segmentation is far more difficult. Feature extraction will be equivalent to SIFT extraction obtaining \( r \) keypoints of query image \( I_q \). Then, for each feature extracted the \( \delta \) feature is calculated, this will be our query vector \( q_j, j = 1..r \). Each \( q_j \) will be classified and a vote will be computed.

2.2.2. Classifiers

Just like in training, the sections 2.2.2.1, 2.2.2.2, 2.2.2.3 and 2.2.2.4 explain the pipeline procedure to obtain the votes used in soft-voting stage (2.2.3). These sections are inside the “for each class” scope.

2.2.2.1. SFS

The SFS computed in 2.1.3 is now used to choose the selected features of our \( q_j \) vector.

\[
y_j = SFS(q_j)
\]

Feature selection importance on this method can be appreciated in image 2.9, that background influences highly in the cells where training images had nothing, changing considerable SIFT descriptor, misleading the classifier. This is the reason why we use the same features selected in the training section, reducing the noise introduced by the background occlusion.
2.2.2.2. Sparse Reconstruction

Once $y_j$ vector is obtained, we calculate its sparse reconstruction $\hat{y}_j$ using Section 1.4.5. First, with $D_i$ we find the sparse representation vector $x$ of $y_j$. Then, the sparse reconstruction $\hat{y}_j$ is calculated with $D_i$ and $x$ as explained in 1.4.5. This sparse reconstruction searches to find the optimum linear combination of the vectors in $D_i$ using $x$ as weights, in other words, to find the best reconstruction of $y_j$ using a linear combination of all samples in training set, including information from one, two or all classes to do this.

2.2.2.3. SCI threshold

In the prior section we explained how the sparse reconstruction may contain information of all classes, this obviously in not very good for our case. Then, we try to identify those $y_j$ reconstructed from many classes, and a threshold for accepting them or not. We achieve this by using the Sparsity Concentration Index or SCI [57] which is calculated with the equation 1.4. Using a threshold over SCI, the method can decide whether to continue processing the vector or discard it, in which case, this vector will no affect the predicted outcome.
2.2.2.4. KNN Classifier and Distance threshold

Usually sparse reconstruction method use as classifier a SVM [7] or reconstruction error. Having tried these, we aim for a simpler classifier, a k - nearest neighbors is used to classify the reconstructed vector $\hat{y}_j$ retrieving not the binary prediction class or no class, but a soft-voting normalized. Using the closest distance to a neighbor ($k$ is not necessarily 1), we use a $distanceThreshold$ 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.

**Algorithm 1** KNN Classifier and Distances Thresholds

```plaintext
for all class $i \neq other$ do
    $(knnPredictedClass, knnDistance) = KNN_i(\hat{y}_j)$
    if $knnDistance \geq distanceThreshold_{knnPredictedClass}$ then
        $\hat{y}_j \leftarrow$ discarded
    else
        $distances(\hat{y}_j, knnPredictedClass) \leftarrow knnDistance$
```

Figure 2.10. Visual example of a few keypoints discarded by SCI threshold
It is good to remember that each KNN classifier will give us if $\hat{y}_j$ belongs to the classifier class or to the class other, as every classifier was trained for a different class. With this, each $\hat{y}_j$ will give us one prediction class and one distance for each class $i$.

### 2.2.3. SoftVoting

Once all distances of every $\hat{y}_j$ had been collected, the votes corresponding to each class are added and special cases are handled; if there is no votes, it means that there where no keypoints selected or all where discarted, then, the image belongs instantly to the other class.

We use a soft voting approach in an attempt to trust more in those keypoints that are closer to the original image keypoints and not only in the presence of them, improving the classifiers performance. In order to do this, each knnDistance is considered to be the vote. In order to use a maximization approach, all votes are calculated by substracting \textit{knnDistance} to the corresponding distanceThreshold. In the case of the \textit{other} class, there is no classifier so there is no threshold either. Finally, the predicted class will be selected as the one with the higher votes.

\begin{algorithm}
\caption{Soft-voting Calculation}
\begin{algorithmic}
\FORALL{$j$}
\FORALL{class $i \neq \text{other}$}
\STATE $v_j = \sum_{j} \text{distanceThreshold}_i - \text{distances}(\hat{y}_j, i)$
\STATE $pc_j = 1$
\ENDFOR
\ENDFOR
\end{algorithmic}
\end{algorithm}

Then, the parameter \textit{minMaxDistance} is calculated and used as a threshold to detect untrusted set of votes, this is done by comparing this parameter with the standard deviation of the votes sample, then the previous algorithm:
Algorithm 3 Soft-voting Classification

for all $j$ do
    for all class $i \neq \text{other}$ do
        $\text{votes}(i) = \sum_{p_{cj}=i} v_j$
    if $\text{std}(\text{votes}) < \text{minMaxDistance}$ then
        $\text{predictedClass}(I_q) \leftarrow 4$
    else
        $\text{predictedClass}(I_q) \leftarrow \text{max}(\text{votes})$
}
3. EXPERIMENTS AND RESULTS

This chapter contains the practical procedures used regardless our method, for this, first the database will be introduced, then the experiments will be explained as well as the parameters adjustment, and finally results will be presented, confusion matrices and performances curves.

3.1. DataBase

We use a subset of GDXray database [37] created for this kind of experiments (folders B0049 though B0082). This has two main sets, training images and testing images, although testing images will be split into validation and testing. These sets are X ray images taken from four different classes, guns, shurikens, razor and others, the other or non – object class contains images as wide as possible, for example, castings, weldings, fishbones, springs, bags portions that don’t include object of the 3 prior classes. Due the legal difficulty to obtain a gun, we used the same gun in different rotations and environments. The specific technicalities are explained in each subsection.

3.1.1. Training Images

There are 900 X-Ray images for training; 300 for the object classes, this is, 100 for guns, 100 for shurikens, and 100 for razors; and 600 for the other class.

The 300 object class contained only one instance of object of the class, centered and complete; this object has no occlusion, nor present other artifacts in the image. Images where taken in different angles with rotations of 20° in X axis and rotations in Y axis generating all possible views to be detected by SIFT keypoints.
3.1.2. Testing Images

There are 1050 X ray images in testing set; 150 correspond to the object classes, this is, 150 for guns, 150 for shurikens, and 150 for razors; and 600 correspond to the class other.

Images are cropped around the object (as for sliding windows approach). The 450 images with guns, razors and shurikens, contain only one object, centered and complete; these images correspond to a object inside a bag, therefore contains other artifacts present, background, creating occlusion of different degrees as shown in figure 3.1. The 600 images without object are X ray images from a variety of environment, among them, the same bag used in the object class, wood, fruit, weldings, other bags without object class present, etc. Testing images are randomly divided into two sets, testing set and validation set. Testing set is to be used for finding parameters, testing a variety of methods, try and error, etc. Validation set was used just one time once the method and its parameters were defined in order to prove the method performance.
FIGURE 3.3. Example of testing images, left image is the complete bag X-Ray, the right image is the cropped and oriented object.

FIGURE 3.4. Example images of testing set (From left to right, guns, shurikens, razors and class other)
Table 3.1. Distribution of images among all sets

<table>
<thead>
<tr>
<th>Class</th>
<th>Training</th>
<th>Validation</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gun</td>
<td>200</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>Shuriken</td>
<td>100</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>Razor</td>
<td>100</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>Other</td>
<td>500</td>
<td>200</td>
<td>400</td>
</tr>
</tbody>
</table>

3.2. Experiments

The experiments used to measure performance for this method are explained on this section. First, for testing stage, complete training set is used as explained in section 2.2. Then, for testing and validation stage testing images are selected by a random vector which is saved for future methods performance comparison, this selection is made by $\frac{1}{3}$ for validation and $\frac{2}{3}$ for testing.

Training and validation stage is where the heavy lifting is made, trying different methods, variations of these methods, and combinations of them forged our currently method. Validation is the repetitive stage where parameters are tuned. Then the method is trained with these selected parameters. In subsection 3.2.1 parameters and it’s optimization is explained.

Then when method is defined and all parameters are selected, we ignore training and validation images and use testing images (unseen for the algorithm so far). In this stage, both, method and parameters remain untouched in such a way that every time we run it the output values are exactly the same.

3.2.1. Parameters adjustment

There are several parameters to adjust in this method, in this section we mention the most relevant parameters to our consideration and how they where selected for this particular set of images. We will divide them into two main method groups, by observation and by exploration.
3.2.1.1. Observation parameter selection

SFS contain two parameters, one is an arbitrary chosen $k$ for the inner knn, the other is $ns$ the number of selected features in SFS. This parameter is selected by running SFS until $J_{max}$ value becomes stable ($J_{max}$ stop increasing or increase is not significant), this is done by visual inspection in the graphic $J_{max}$ vs $ns$ as shown in picture 3.5. $k$ is selected as the same $k$ in 3.2.1.2, then, selected value of $s$ is:

$$s = 50$$

We use the same $s$ for each classifier in order to maintain vectors of same size and a fair comparison among the results.

![Figure 3.5](image)

FIGURE 3.5. In the top, Jmax vs number of features using SFS for each class. In the bottom, a zoom in to the appreciate the stabilization point of $J_{max}$ (near 50 in every case)
SIFT contains several parameters, all parameters are set as default except for the threshold for peak selection $PeakThresh$, which basically is a sensitivity threshold selecting more or less keypoints. We use $PeakThresh = 1$ to avoid selecting keypoints that are actually noise in the image as shown in figure 3.6.

![Figure 3.6](image)

**Figure 3.6.** The first image shows the original image, the second correspond to the segmented image and third image is the mask mentioned in 2.1.1 and the fourth image shows all SIFT keypoints filtered using the mask of third image.

### 3.2.1.2. Exploration parameter selection

There are several parameters founded using search by exploration among two values. The $k$ of K-means classifier was selected iterating with $k \in [1, 10]$. In sparse reconstruction we use default parameters, except the $SCIThreshold$ parameter was iterated using $SCIThreshold \in [0, 1]$ using 1 decimal precision. All distance threshold were iterated in the interval $[10000, 100000]$ with jumps of 10000 in a for loop, parameters for each class where selected at the same time, is meaningful to mention that the value for the $other$ class is 0, meaning that none $\hat{y}_j$ classified as $other$ is affected. The $minMaxDistance$ parameter is also iterated using $minMaxDistance \in [0, 100000]$ using intervals of 100. Finally, selected parameters of this section are:

\[
k = 5
\]
\[
sciThreshold = 0.9^1
\]
\[
distanceThresholdPistol = 50000
\]
\[
distanceThresholdShuriken = 80000
\]

^1 all SCI threshold turn out to be the same result
distanceThresholdRazor = 90000

minMaxDistance = 21700

In figures 3.7, 3.8, 3.9 and 3.10 we can see the accuracy for our method while we variate one of the prior parameters while maintaining the others static.

**FIGURE 3.7.** We can see the accuracy behavior when variating the DistanceThresholdPistol, this accuracy correspond to the final performance of our method, including the four classes. We can see clearly a global optimal value at 5000.
FIGURE 3.8. We can see the accuracy behavior when varying the DistanceThresholdRazor, this accuracy correspond to the final performance of our method, including the four classes. We can see clearly a global optimal value at 9000 and then it starts to decay. Having a higher value in this threshold compared with Pistol threshold shows us that the Razor class has less intraclass variance and thus the threshold can be more refined (lets remember that soft-voting works in a inverse normalized approach).

FIGURE 3.9. We can see the accuracy behavior when varying the DistanceThresholdShuriken, this accuracy correspond to the final performance of our method, including the four classes. We can see clearly a global optimal value at 8000 and then it starts to decay. As with DistanceThresholdRazor, the shuriken class has less variance within, and its threshold can be more refined.
Figure 3.10. We can see the accuracy behavior when varying the minMaxDistance, this accuracy correspond to the final performance of our method, including the four classes. We can see many local optimal values, however, there is a global optimal value at 21700 and then it starts to decay.

3.3. Curves

As we have three classifiers (guns, shurikens and razors), for each parameter, we present the Precision-Recall curves for each one, and using the mean of these curves, we calculate an Precision-Recall curve for the general multi-class classification.

SCI: For each class we present the precision recall curves while varying the SCI threshold. We can appreciate that curves behavior variates for the different classes, this shows that SCI acts within classes, and not isolated one from another.
Distances thresholds: For each class, we present the Precision-Recall curves variating three different parameters distanceThresholdPistol, distanceThresholdShuriken and distanceThresholdRazor. We can see that there is no optimal general solution, this because the intraclass variance, a pistol have more variability among his keypoints than shuriken class. As we variate a particular distance threshold, the other classes performance variates, we deduce that this is because these are not completely disjoint classes, otherwise, a simple knn with k=1 would be sufficient to solve this problem.
Figure 3.12. Precision-Recall Curves varying \textit{distanceThresholdPistol} parameter

Figure 3.13. Precision-Recall Curves varying \textit{distanceThresholdShuriken} parameter
**Figure 3.14.** Precision-Recall Curves variating *distanceThresholdRazor* parameter

**minMaxDistance:** For each class we present the Precision-Recall curves while variating the *minMaxDistance* parameter, we can see that different than SCI threshold, the *minMaxDistance* behavior is similar in all classes, this is because it is a parameter that affect within the class.
3.4. Results

Two different experiments were made

A) 3 binary classifiers: Pistol vs. No Pistol, Shuriken vs. No shuriken, Razor vs. No razor

B) 4-class classifier: Pistol, Shuriken, Razor and Others. This classifier is constructed with the three classifiers above, when none of them classifies as their class, the images is classified as other, when one of them classifies as their class that is the predicted class, and when more then one classifies as their class, we use the soft-voting values to decide.

The numeric results for these two experiments in validation and testing stage are shown ahead.

3.4.1. Validation Performance

Here we present the accuracy, precision and recall obtained in Validation set after parameter tuning (our optim found).
### Table 3.2. Precision & Recall for validation set in experiment (A)

<table>
<thead>
<tr>
<th></th>
<th>Pistol</th>
<th>Shuriken</th>
<th>Razor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pr Re</td>
<td>Pr Re</td>
<td>Pr Re</td>
<td></td>
</tr>
<tr>
<td>Ours</td>
<td>1.00 0.97</td>
<td>1.00 0.99</td>
<td>0.98 0.93</td>
</tr>
</tbody>
</table>

### Table 3.3. Accuracy for each detector in validation set (in percentages) in experiment (B)

<table>
<thead>
<tr>
<th></th>
<th>Pistol</th>
<th>Shuriken</th>
<th>Razor</th>
<th>Other</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours</td>
<td>97.14</td>
<td>99.14</td>
<td>91.71</td>
<td>90</td>
<td>94.5</td>
</tr>
</tbody>
</table>

### 3.4.2. Comparison of different methods

In order to have a referral for our method performance, in table 3.4 and 3.5 we compare our method.

In the first section, we evaluate the two most popular Convolutional Neural Network (CNN) models related with ImageNet: the AlexNet model proposed by Krizhevsky [20] and the GoogleNet model proposed by Szegedy [52]. Also we include Adaptive Implicit Shape Model (AISM), which was presented originally in [47] for object recognition in baggage screening.

In the second section, we compare it with basic general methods. For each method, we use a similar approach as for our method, we extract SIFT from training images and use them to train the models, then in testing we classify each SIFT extracted from the query image using the corresponding method (Knn, SVM, AdaBoost or SRC) and use its predicted class as vote, then the predicted class of the image is calculated by counting votes in each classifier.
### Table 3.4. Precision & Recall for each detector in experiment (A)

<table>
<thead>
<tr>
<th>Detector</th>
<th>Pistol</th>
<th>Shuriken</th>
<th>Razor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$Pr$</td>
<td>$Re$</td>
<td>$Pr$</td>
</tr>
<tr>
<td>Sparse-Knn (Ours)</td>
<td>0.99</td>
<td>0.97</td>
<td>1.00</td>
</tr>
<tr>
<td>GoogleNet [20]</td>
<td>0.83</td>
<td>1.00</td>
<td>0.99</td>
</tr>
<tr>
<td>AISM [47]</td>
<td>0.97</td>
<td>0.97</td>
<td>0.95</td>
</tr>
<tr>
<td>AlexNet [52]</td>
<td>0.85</td>
<td>0.99</td>
<td>1.00</td>
</tr>
<tr>
<td>Knn</td>
<td>0.92</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>SVM</td>
<td>0.90</td>
<td>0.99</td>
<td>1.00</td>
</tr>
<tr>
<td>AdaBoost [44]</td>
<td>1.00</td>
<td>0.87</td>
<td>1.00</td>
</tr>
<tr>
<td>SRC</td>
<td>0.75</td>
<td>1.00</td>
<td>0.80</td>
</tr>
</tbody>
</table>

On these comparisons we can see the power of our method, with a higher accuracy than all classic methods presented and competing with others powerful methods state of the art like AISM and deep learning methods, noting that the best performance was achieved by GoogleNet with a 96.3% accuracy. In the table 3.4 we can see that the weakness of our

### Table 3.5. Accuracy for each detector (in percentages) in experiment (B)

<table>
<thead>
<tr>
<th>Detector</th>
<th>Pistol</th>
<th>Shuriken</th>
<th>Razor</th>
<th>Other</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sparse-Knn (Ours)</td>
<td>97.0</td>
<td>99.4</td>
<td>91.9</td>
<td>90.6</td>
<td>94.7</td>
</tr>
<tr>
<td>GoogleNet [20]</td>
<td>100.0</td>
<td>100.0</td>
<td>95.0</td>
<td>90.0</td>
<td>96.3</td>
</tr>
<tr>
<td>AISM [47]</td>
<td>96.0</td>
<td>94.0</td>
<td>99.0</td>
<td>92.5</td>
<td>95.4</td>
</tr>
<tr>
<td>AlexNet [52]</td>
<td>99.0</td>
<td>100.0</td>
<td>72.0</td>
<td>93.5</td>
<td>91.2</td>
</tr>
<tr>
<td>Knn</td>
<td>93</td>
<td>93</td>
<td>87</td>
<td>83</td>
<td>89</td>
</tr>
<tr>
<td>SVM</td>
<td>91</td>
<td>86</td>
<td>86</td>
<td>79</td>
<td>85.5</td>
</tr>
<tr>
<td>AdaBoost [44]</td>
<td>87</td>
<td>86</td>
<td>84</td>
<td>60</td>
<td>79.25</td>
</tr>
<tr>
<td>SRC</td>
<td>79</td>
<td>83</td>
<td>52</td>
<td>80</td>
<td>73.5</td>
</tr>
</tbody>
</table>

On these comparisons we can see the power of our method, with a higher accuracy than all classic methods presented and competing with others powerful methods state of the art like AISM and deep learning methods, noting that the best performance was achieved by GoogleNet with a 96.3% accuracy. In the table 3.4 we can see that the weakness of our
methods is the class razor blades, with a 0.94 recall vs. a 0.97 and a 0.99 in pistol and shurikens respectively. It is our believe that this is produced by the thinness of the object.

Figure 3.16. Confusion Matrix for Sparse-Knn in testing set
4. CONCLUSION

The validation set and testing set gave similar results, proving that data is stable. Both datasets where obtained from the same parent set, these being chosen in a random manner, with this, if both results where different, it would mean that data was not equivalent in both. This ensures that results are not highly related with the selection of these sets.

Feature selection method SFS has proven to be efficient with SIFT features, reducing the dimensionality from 128 to 50, we do not say that $s = 50$ is a mandatory number, but rather a parameter to be tuned to fit the need of every problem. This selection let us deal with the background noise of the images thanks that the training set has clear images of the objects.

Conceptually, soft voting is a better approach than binary voting system. In our method this has been proven right. We have tried several types of votes, such as max distance of knn, average distance of knn, SCI, error reconstruction over sparse reconstruction, etc.

Even thought our method shows great performance, deep learning based method have similar results and even better in some cases, GoogleNet reaches a 96.3% accuracy, while AlexNet gets a 90.7% (a 4.7% lower than ours). 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 work.

Sparse techniques are very useful in different environments. For example, using the error reconstruction [57] can be very useful for iris recognition, a more complex model based on SRC [33] can be very powerful when it comes to face recognition. In our case, these methods won’t give that good results, but if used as a hidden variable, can provide a helpful index, SCI let us separate those query vector that are reconstructed using many classes and therefore don’t belong to any of those, detecting false positive detection and
correcting them on due time. Hence forth, while distance thresholds are useful within the binary class, SCI thresholds are needed when we deal with several classes, especially with those close to each other (p.e. knife sharp and shuriken sharp).

Dealing with rigid objects, knn has proven to be more efficient than SVM in time and accuracy, and using parametric thresholds it boosts up the accuracy results in our method. This reminds us of the principle of Occam’s Razor; the simpler solution is most commonly the best solution.

Unfortunately our method is not easily extendable to other classes. If we refer to coding, our method is pretty simple. Just add the names, paths and number of classes $i$. But then, SFS would have to be calculated again; the distance thresholds and other parameters. This is true because the new class might be very similar to a prior class and thresholds should be more refined, or the class will be very distant to know classes and threshold could remain without variation. Also, we would have to make sure that the class other doesn’t contain images of the new class. With this, our method is not easily extendable.

Our method considers one view. Then, as future work an extension of it could consider multiple views like [36]. This is likely possible as we use SIFT as a descriptor. Lowe said that key points can be successfully detected in views with 20 degrees rotation. Keypoints are effective for matching them to an object. Multiple views can provide sturdiness to our model, and its filters can be applied in order to recognize those keypoints of interest as done in [31, 35].
References


Domingo Mery, Vladimir Riffó, Uwe Zscherpel, German Mondragón, Iván Lillo, Irene Zuccar, Hans Lobel, and Miguel Carrasco. GDXray: The database of X-ray


Vladimir Riffio and Domingo Mery. Automated detection of threat objects using adapted implicit shape model. 2015.


