Automatic Defect Recognition in X-ray Testing using Computer Vision

Domingo Mery
Department of Computer Science
Universidad Católica de Chile
domingo.mery@uc.cl

Carlos Arteta
Department of Engineering Science
University of Oxford, UK
carlos.arteta@eng.ox.ac.uk

Abstract

To ensure safety in the construction of important metallic components for roadworthiness, it is necessary to check every component thoroughly using non-destructive testing. In last decades, X-ray testing has been adopted as the principal non-destructive testing method to identify defects within a component which are undetectable to the naked eye. Nowadays, modern computer vision techniques, such as deep learning and sparse representations, are opening new avenues in automatic object recognition in optical images. These techniques have been broadly used in object and texture recognition by the computer vision community with promising results in optical images. However, a comprehensive evaluation in X-ray testing is required. In this paper, we release a new dataset containing around 47.500 cropped X-ray images of 32 $\times$ 32 pixels with defects and no-defects in automotive components. Using this dataset, we evaluate and compare 24 computer vision techniques including deep learning, sparse representations, local descriptors and texture features, among others. We show in our experiments that the best performance was achieved by a simple LBP descriptor with a SVM-linear classifier obtaining 97% precision and 94% recall. We believe that the methodology presented could be used in similar projects that have to deal with automated detection of defects.

1. Introduction

Automated defect recognition is a key issue in many quality control processes based on visual inspection [4]. Non-homogeneous regions can be formed within the work piece in the production process, which can manifest, for example, as bubble-shaped voids, fractures, inclusions or slag formations. This kind of defects can be observed as discontinuities in the welding process [31], concrete defects [12], glass defects [6], steel surface defects [23] or flaws in light-alloy castings used in the automotive industry [33], among others. In order to ensure safety, it is necessary to check every part thoroughly.

Light-alloy castings produced for the automotive industry, such as wheel rims, steering knuckles and steering gear boxes, are considered important components for overall roadworthiness. In casting inspection, automated X-ray systems have not only raised quality through repeated objective inspections and improved processes, but have also increased productivity and consistency by reducing labor costs. Some examples of castings with defects are illustrated in Fig. 1. Cropped images of regions with and without defects are shown in Fig. 2, in which it is clear that there are some patterns that can be easily detected (*e.g.* defects that are bright bubbles with dark background and no-defects that are regular structures with edges). However, the recognition of both classes can be very difficult for low contrast defects because they are very similar to homogeneous no-defects. Typically, the signal-to-noise ratio (SNR) in this kind of images is low, *i.e.* the flaws signal is only slightly greater than the background noise.

Different methods for the automated detection of casting discontinuities using computer vision have been described in the literature over the past thirty years [33]. One can see that the different approaches for this task can be divided into three groups: *i)* approaches where an error-free reference image is used; *ii)* approaches using pattern recognition, expert systems, artificial neural networks, general filters or multiple view analyzes to make them independent of the position and structure of the test piece; and *iii)* ap-

Table 1. Defect recognition on castings

Reference	Description
Carrasco [5]	Multiple view correspondence with non-calibrated model
Cogranne [9]	Statistical hypothesis testing using nonparametric tests
Li [22]	Wavelet technique
Li [21]	Peak location algorithm combined with neural networks
Mery [34]	Multiple view model with calibrated model
Mery [30]	Multiple views using an un-calibrated tracking approach
Pieringer [39]	3D model from 2D images
Pizarro [40]	Multiple views based on affine transformation
Ramirez [41]	Generative and discriminative approaches
Tang [48]	Segmentation using fuzzy model
Zhao [53]	Statistical feature based on grayscale arranging pairs
Zhao [54]	Sparse representations

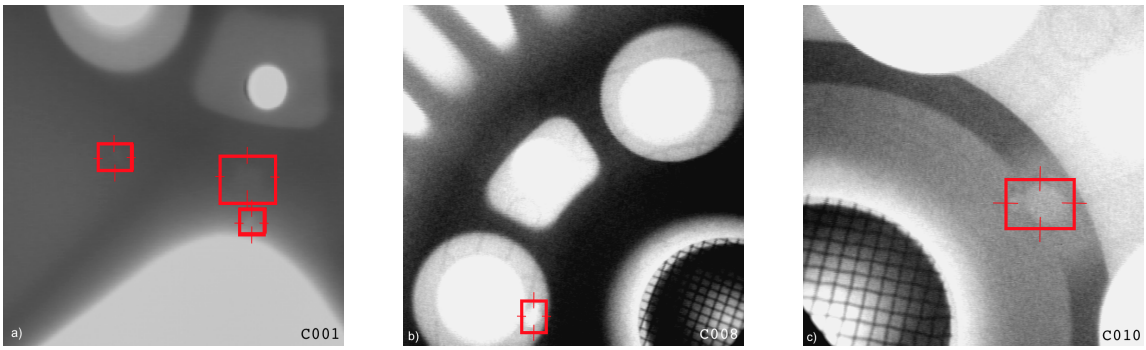

Figure 1. Examples of defects in real X-ray images of wheels from GDXray dataset [35].

proaches using computer tomography to make a reconstruction of the cast piece and thereby detect discontinuities [29]. A selection of recent approaches are summarized in Table 1. In general, they use classical image processing and pattern recognition techniques with a single view or multiples views, where decision is made by analyzing the correspondence among the views.

Nowadays, modern computer vision techniques, such as deep learning, *e.g.* [18], [3] and [45], and sparse representations, *e.g.* [49] and [52], are opening new avenues in automatic object recognition in optical images. These techniques have been broadly used in object and texture recognition by the computer vision community with promising results in optical images, however, a comprehensive evaluation in X-ray testing is required.

This paper attempts to make a contribution to the field of automatic defect recognition with computer vision, specially when inspection is performed using X-ray testing. Our contribution is threefold:

1. Dataset: We release a new data set obtained from GDXray database [35]. It consists of 47.520 cropped X-ray images of 32 $\times$ 32 pixels and their labels (see some examples in Fig. 2). The dataset is public and can be used for research and educational purposes[1]. See details in Section 2. For the dataset, we define a standard evaluation protocol based on disjoint learning and testing data, *i.e.* pieces that are present in learning set are not allowed to be in testing set. Future work can be established using the same dataset and protocol in order to make fair comparisons. See details in Section 4.

2. Evaluation of computer vision techniques: We compare 24 computer vision techniques including deep learning, sparse representations, local descriptors and texture features among others. To the best of our knowledge, this is the first work in X-ray testing with an exhaustive evaluation of a considerable variety of computer vision methods including deep networks. See details of the methods in Section 3 and the comparisons in Section 5.

3. Code: We release the (Matlab) code used in our experiments. We believe that our methodology could be easily adapted to other similar projects that have to deal with automated detection of defects. See details in Section 5.

The rest of the paper is organized as follows. First, the dataset is introduced in Section 2. The 24 computer vision techniques used in our experiments are explained in Section 3, and the evaluation protocol used is described in Section 4. The obtained results are presented and discussed in Section 5. Finally, some concluding remarks and future work are given in Section 6.

2. Database

We now present the defect detection database. In this task, there are two classes: defects and no-defects (see same examples in Fig. 2). Representative (cropped) X-ray images from both classes are extracted from the group 'castings' of the GDXray database [35]. The digital X-ray images are quantized at 8 bits, the focal length is 884mm and the field of view is 40cm. GDXray is a public database for X-ray testing with more than 20,000 images (including castings, welds, baggage and nature products). The X-ray images of GDXray can be used free of charge, for research and educational purposes only[2]. Fourteen series of GDXray were used (C001, C002, C007, C008, C010, C030, C034, C041, C045, C051, C054, C057, C062 and C065). These series contain 610 X-ray images of aluminum wheel rims and steering knuckles. Series C001 and C002 belong to an aluminum wheel commonly used for testing purposes (see for example [34], [5] and [41]). Defects on these castings are of two types. First, a group of blow holes defects (with

[1] The dataset consists of original and augmented patches. In preliminary experiments we tested with the original patches (only) and some augmentation (some rotations), however, the performance was significantly lower than the reported results. We used the reported augmentation (see Section 2) in all experiments in order to perform fair comparisons (the evaluation protocol was exactly the same for every method). Nevertheless, in our dataset the information of which patch is original and which is augmented is available, so other experiments can be done in the future.

[2] Available at http://dmery.ing.puc.cl/index.php/material/gdxray/.

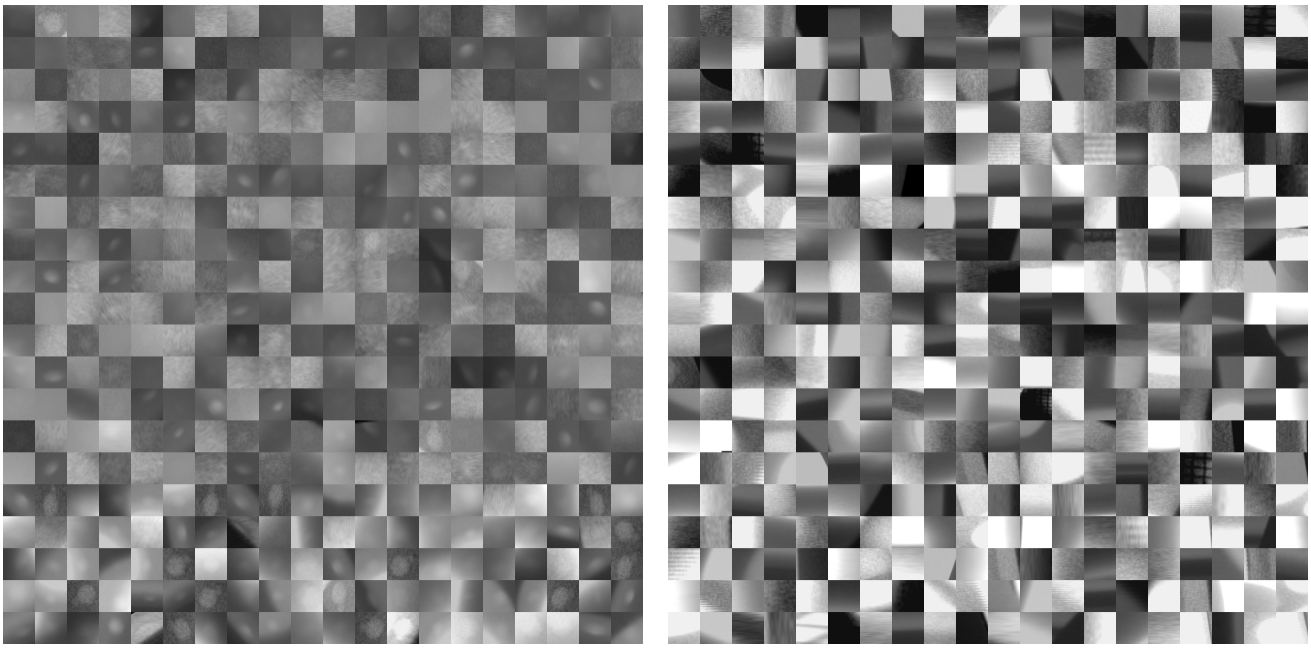

Figure 2. Examples of patches containing defects (left) and no-defects (right) from our database (extracted from $\mathbb{GDX}$ray dataset [35]).

$\emptyset$ = 2.0 – 7.5 mm) which were already present in the castings. They were initially detected by (human) visual inspection (see for example Fig. 1a). The remaining defects in these castings were produced by drilling small holes (with $\emptyset$ = 2.0 – 4.0 mm) in positions of the casting which were known to be difficult to detect (for example at edges of regular structures). Series C001 and C002, with 163 images, are used for testing purposes and the other mentioned series, C007 ... C065, with 447 images, for learning purposes (see some examples in Fig. 1b-c). It is worth mentioning that learning and testing sets are completely disjoint, *i.e.* the casting pieces of learning set do not belong to testing set (and viceversa).

We used the bounding boxes of $\mathbb{GDX}$ray to extract 990 cropped images of the defects that are present in the mentioned 610 X-ray images (676 and 314 cropped images from learning and testing images respectively). All cropped images are resized to 32×32 pixels. We built a large set of defects by rotating the original images at 6 different angles $(0^0, 60^0, 120^0 \ldots 300^0)$. In addition, each rotated image is flipped vertically, horizontally and both vertically and horizontally, yielding 4 different versions of each rotated image. Thus, each original cropped defect has $6 \times 4 = 24$ different versions, *i.e.* our database has $24 \times 990 = 23.760$ cropped images of the class defects.

The set of defects is split into learning and testing subsets as follows: $24 \times 676 = 16.224$ defects were obtained from the learning series (C007 ... C065) and $24 \times 314 = 7.536$ defects were obtained from learning series (C001 and C002).

In order to build the class no-defects, we extracted randomly 23.760 cropped images in the same mentioned series of $\mathbb{GDX}$ray in places where there is no defect. Again, 16.224 no-defects were obtained from learning series (C007 ... C065) and 7.536 no-defects from learning series (C001 and C002).

The whole database, *i.e.* 47.520 cropped X-ray images of 32×32 pixels and their labels, is available on our webpage[3].

3. Computer vision methods

In order to establish a computer vision benchmark using the dataset in Section 2, we tested the following 24 approaches that can be used in defect detection in X-ray testing. Many of these methods correspond to a descriptor or a feature vector extracted from the patch, and the discrimination between the two classes, defects and no-defects, can be performed using a classifier. In our experiments we tested several instances of K-nearest neighbors (KNN), Support Vector Machines (SVM), and artificial neural networks (ANN) classifiers.

• **Intensity grayvalues:** The $w \times w$ grayscale cropped image is converted into a vector $\mathbf{x}$ of $m = w^2$ elements given by stacking its columns. In our experiments, we used $\mathbf{x}/||\mathbf{x}||$ (with 1.024 elements for $w = 32$) as a descriptor of the cropped image. The uni-norm is used to make the descriptor invariant to contrast. In our experiments, we call these features INT.

[3] See http://dmery.ing.puc.cl/index.php/material/

• **Local binary patterns:** Texture information extracted from occurrence histogram of local binary patterns (LBP) computed from the relationship between each pixel intensity value with its neighbors. The features are the frequencies of each one of the histogram bins [38]. Other LBP features like *semantic* LBP (sLBP) [36] can be used in order to bring together similar bins. In our experiments, we used the uniform LBP features (LBP), the rotation invariant LBP features (LBP-ri) and the semantic LBP features (sLBP) of 59, 36 and 123 elements respectively.

• **Crossing Line Profile:** This is an image processing technique that was developed for defects detection [28]. It is based on features extracted from gray level profiles along straight lines crossing the cropped images (*e.g.* standard deviation, difference between maximum and minimum and Fourier components). There are 12 features extracted from this profile. In our experiments, we call these features CLP.

• **Statistical textures:** They are computed utilizing *co-occurrence matrices* that represent second order texture information (the joint probability distribution of intensity pairs of neighboring pixels in the image), where mean and range –for 3 different pixel distances in 8 directions– of the following variables were measured: 1) Angular Second Moment, 2) Contrast, 3) Correlation, 4) Sum of squares, 5) Inverse Difference Moment, 6) Sum Average, 7) Sum Entropy, 8) Sum Variance, 9) Entropy, 10) Difference Variance, 11) Difference Entropy, 12 and 13) Information Measures of Correlation, and 14) Maximal Correlation Coefficient [14]. The size of the descriptor is $2 \times 14 \times 3 = 84$. In our experiments, we call these features Haralick.

• **Fourier and Cosine Transforms:** These 2D–Transforms give us information about the 2D frequencies that are present in the cropped images [11]. For Fourier Transform, we used the magnitud of the first quadrant of the 2D Discrete Fourier Transform of the 32×32 image image, yielding a descriptor of $16 \times 16 = 256$ elements. For Cosine Transform [11], we use the whole transformed image, *i.e.* the descriptor has 1024 elements. In our experiments, we call these features Fourier and DCT respectively.

• **Gabor:** This texture information is based on 2D Gaussian-shaped bandpass filters, with dyadic treatment of the radial spatial frequency range and multiple orientations, which represent an appropriate choice for tasks requiring simultaneous measurement in both space and frequency domains. Additionally, the maximum, the minimum and the difference between both are computed [17]. We tested several scales and orientations, the best results were obtained using 4 scale and 4 orientations. The size of the descriptor is $4 \times 4 + 3 = 19$. In our experiments, we call these features Gabor. In addition, we compute a descriptor as a concatenation of the output of all Gabor filters (in four different scales and four orientations) of a subsampled input image yielding a feature vector of $16 \times 16 \times 4 \times 4 = 4.096$ elements [13]. In our experiments, we call these features Gabor+.

• **Binarized statistical image features:** This is a texture description based on a binary code that is computed via linear projection using independent component analysis of natural images [15]. We tested all filters proposed by the authors, the best performance was achieved by a filter of 7×7 pixels, with a binarization of 11 bits and a normalized histogram. The size of the descriptor is 2.048. In our experiments, we call these features BSIF.

• **Sparse representation:** In the sparse representation approach, a dictionary is built from the learning images, and testing is done by reconstructing the testing image using a sparse linear combination of the atoms of the dictionary. The testing image is assigned to the class with the minimal reconstruction error [49]. We tested two well known sparse representation approaches. In the first one, the dictionary is the gallery of training images itself [52]. In the second one, the dictionary is computed from the training images using KSVD algorithm [1]. We tested several configurations and we chose the best one. For the implementation of the first case, we used a dictionary of 1.024×32.448 elements, and for the sparse representation we use ten non-zeros coefficients. In our experiments, we call this method SRC. In the second case, the dictionary of each class has 5.000 atoms, and the reconstruction is performed using ten non-zero coefficients. In our experiments, we call this method Sparse.

• **SIFT:** The SIFT-descriptors (Scale Invariant Feature Transform) are local features based on the appearance of the object. The descriptors are invariant to scale, rotation, lighting, noise and minor changes in viewpoint [24]. In addition, they are highly distinctive and relatively easy to extract and match against a (large) database of local features. The SIFT-descriptor consists of a 128-element vector, which corresponds to a set of 16 gradient oriented histograms of 8 bins distributed in a 4×4 grid. In our experiments, we call these features SIFT.

• **SURF:** The Speeded Up Robust Feature (SURF) is based on the sum of the Haar Wavelet response in 4×4 square sub-regions of the patch [2]. Similar to SIFT, SURF–descriptor is invariant to scale and rotation. The descriptor has 64 elements. In our experiments, we call these features SURF.

• **HOG:** HOG is a descriptor that contains histograms of oriented gradients in cells distributed in a grid manner across the image [10]. In our case, the cell-size is 8 pixels. Thus, the descriptor has 496 elements. In our experiments, we call these features HOG.

• **BRISK:** The Binary Robust Invariant Scalable Keypoint (BRISK) is a descriptor composed as a binary string by concatenating the results of simple brightness comparison tests using a circular sampling pattern [20]. The descriptor has 64 elements. In our experiments, we call these features BRISK.

• **Deep models** In recent years, deep learning has been successfully used in image and video recognition (see for example [18], [3] and [45]), and it has been established as the state of the art in many areas of computer vision. The key idea of deep learning 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, energy based models, Boltzmann machines, and deep belief networks, among others [3]. Convolutional neural networks (CNN), originally inspired by a biological model [19], is a very powerful class of methods for image recognition [16] which replaces feature extraction and classification with a single neural network. A CNN maps an input image $\mathbf{X}$ onto an output vector $\mathbf{y} = \mathcal{G}_L(\mathbf{X})$, where the function $\mathcal{G}_L$ can be viewed as a feed-forward network with L layers which are typically linear operations followed by non-linearities (*e.g.* convolutions followed by rectified linear units [37]). These layers f_l, for $l = 1 \dots L$ contain parameters $\mathbf{w} = (\mathbf{w}_1, \cdots \mathbf{w}_L)$ that can be discriminatively learned from training data: a set of input images $\mathbf{X}_i$ and their corresponding labels $\mathbf{z}_i$, for $i = 1, \cdots n$, so that $\sum_i \ell(\mathcal{G}_L(\mathbf{X}_i), \mathbf{z}_i)/n \rightarrow \min$, where ℓ is a loss function. This optimization problem can be solved using the back-propagation algorithm [42].

We explore several CNN models for our task of defect detection. First, we took standard CNN architectures that have been previously trained on the ImageNet dataset [43], a large and highly-variable collection of (optical) images, and used them as general-purpose feature extractors. This was done by taking the activations of one of the hidden layers of the CNN model and used it as the feature vector in our problem. In this configuration, we evaluated some of the most popular CNN models related to ImageNet: GoogleNet [47], AlexNet [16] and VGG networks VGG-19 [45], VGG-F and VGG-2048 [8]. The size of the descriptors are 1.000, 4.096, 4.096, 4.096 and 2.048 respectively. The input image of these models are images of $244 \times 244 \times 3$ pixels. For this reason, our 32×32 pixel-images were resized to the required size (all three channels are equal). In our experiments, we call these features GoogleNet, AlexNet, VGG-19, VGG-F and VGG-2048 respectively.

In addition to the off-the-shelf CNN models, we designed our own and much simpler CNN for the defect detection task: a five weight-layer net (and 10 layers in total) called Xnet (see Fig. 3). In our notation, layer l corresponds to m_l 3D filters of $p_l \times p_l \times q_l$ elements. The input is a 32×32 pixel cropped image, and the output is a 2-element vector which is a softmax over the two classes. The layers are called $f_1 \cdots f_{10}$, and they consist of five convolutional kernels, $\mathbf{C}_1 \cdots \mathbf{C}_5$, two pooling layers $\mathbf{P}_1, \mathbf{P}_2$ and one rectified linear unit $\mathbf{R}_1$. Additionally, we included a dropout block $\mathbf{D}_1$ that randomly turns off connections of the neural network during training, which has been shown to reduce significantly the over-fitting [46]. We tested several CNN configurations, and the best one –that we call Xnet– is given in Table 2. Finally, we tested Xnet after an additional pre-training on ImageNet. We refer to this pre-trained Xnet network as ImageXnet. In both cases, we extracted a feature vector in layers #9, #8 and #7 (we used the vector in the classification) and the accuracy was very similar compared with the output of layer #10. For this reason, we report the accuracy using layer #10 only.

Table 2. Details of proposed Xnet (see Fig. 3).

l	Name	Layer type	Layer size $m_l(p_l \times p_l \times q_l)$	Output size $n_l \times n_l \times c_l$
0	Input	–	–	$32 \times 32 \times 1$
1	C_1	conv	$32(7 \times 7 \times 1)$	$26 \times 26 \times 32$
2	P_1	pool-max	$2 \times 2 \times 1$	$13 \times 13 \times 32$
3	C_2	conv	$64(5 \times 5 \times 32)$	$9 \times 9 \times 64$
4	P_2	pool-max	$2 \times 2 \times 1$	$4 \times 4 \times 64$
5	C_3	conv	$128(3 \times 3 \times 64)$	$2 \times 2 \times 128$
6	R_1	relu	–	$2 \times 2 \times 128$
7	D_1	dropout	–	$2 \times 2 \times 128$
8	C_4	conv	$64(2 \times 2 \times 256)$	$1 \times 1 \times 64$
9	C_5	conv	$2(1 \times 1 \times 64)$	$1 \times 1 \times 2$
10	S_1	softmax	–	$1 \times 1 \times 2$

4. Evaluation protocol

As explained in Section 2, the learning set consists of $n_{\mathsf{learn}} = 32.448$ samples (16.224 from each class)[4]. In addition, the testing set consists of $n_{\mathsf{test}} = 15.072$ samples (7.536 from each class). We define a standard hold-out evaluation protocol based on disjoint learning and testing data, *i.e.* pieces that are present in the learning set are not allowed to be in the testing set. We denote the cropped X-ray images and their labels $(\mathbf{X}, \mathbf{z})$ as:

• Learning: $\{\mathbf{X}^{(i)}_{\mathsf{learn}}, \mathbf{z}^{(i)}_{\mathsf{learn}}\}_{i=1}^{n_{\mathsf{learn}}}$ 32.448 samples (16.224 from each class)

• Testing: $\{\mathbf{X}^{(i)}_{\mathsf{test}}, \mathbf{z}^{(i)}_{\mathsf{test}}\}_{i=1}^{n_{\mathsf{test}}}$ 15.072 samples (7.536 from each class)

We distinguish between learning and testing protocols. In the learning protocol, we learn a model that is able to represent a cropped image $\mathbf{X}$ as a discriminative descriptor $\mathbf{d}$ that can be used in a classification approach $h(\mathbf{d})$.

A classifier h can be designed using the descriptors and the labels of the learning set. Thus, the classifier can be learned using $(\mathbf{d}^{(i)}_{\mathsf{learn}}, \mathbf{z}^{(i)}_{\mathsf{learn}})$ where $\mathbf{d}^{(i)}_{\mathsf{learn}} = s(\mathbf{X}^{(i)}_{\mathsf{learn}})$ and $s(\cdot)$ is the function that extracts the features (descriptor) of

[4] 75% of them could be selected randomly for training and 25% for validation purposes. Thus, the training dataset contains $n_{\mathsf{train}} = 24.336$ samples (12.168 from each class), and the validation dataset contains $n_{\mathsf{val}} = 8.112$ samples (4.056 from each class).

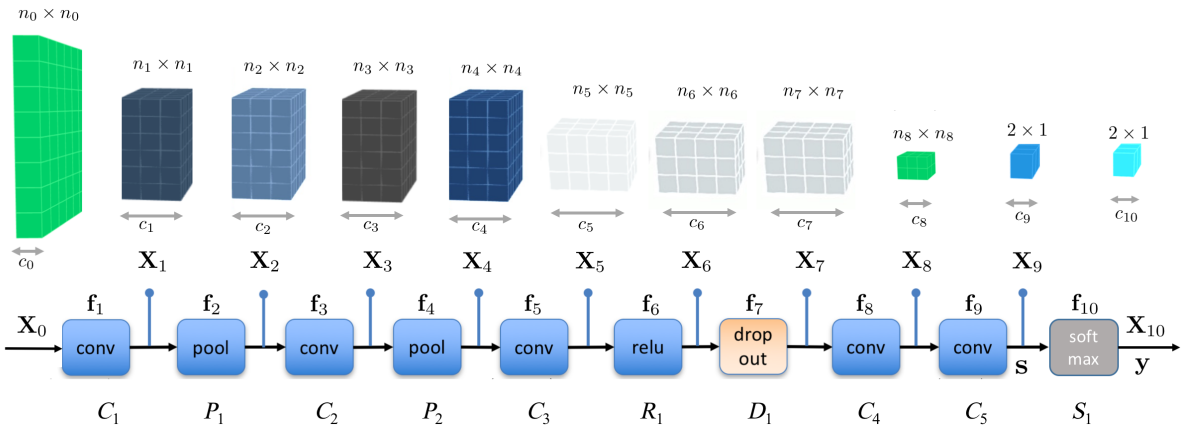

Figure 3. Proposed convolutional neural network Xnet (see definition of variables in Table 2).

the image. After training, $h(\mathbf{d}^{(i)}_{\text{learn}})$ should ideally be $\mathbf{z}^{(i)}_{\text{learn}}$. In order to evaluate the performance of the designed classifier, we use the learned model on the testing set to determine if a testing image is correctly classified. Thus, we test if $h(\mathbf{d}^{(i)}_{\text{test}})$ is equal to $\mathbf{z}^{(i)}_{\text{test}}$, where $\mathbf{d}^{(i)}_{\text{test}} = s(\mathbf{X}^{(i)}_{\text{test}})$. We can measure on the whole testing set for $i = 1 \dots n_{\text{test}}$:

• True Positive (TP): number of defects correctly classified.

• True Positive (TN): number of no-defects correctly classified.

• False Positive (FP): number of no-defects classified as defects. The false positives are known as 'false alarms' and 'Type I error'.

• False Positive (FN): number of defects classified as no-defects. The false negatives are known as 'Type II error'.

From these statistics, we can obtain the well known Precision (Pr), Recall (Re) and Accuracy (η):

$$Pr = \frac{TP}{TP+FP} \quad Re = \frac{TP}{TP+FN} \quad \eta = \frac{TP+TN}{P+N} \qquad (1)$$

Ideally, $Pr = Re = \eta = 1$, *i.e.* all defects are detected with no false alarms.

In our experiments, we report the obtained precision, recall and accuracy of each model.

5. Experimental Results

We tested 24 methods (see details in Section 3) using the evaluation protocol of Section 4 based on a hold-out schema with 32.448 cropped images for learning and 15.072 cropped images for testing purposes. The results are summarized in Table 3 and in Fig. 4. The methods are sorted in descending order according to obtained the accuracy (1). For methods based on sparse representations, the classification was performed by minimizing the reconstruction error. For methods based on deep learning, the classification was performed using SoftMax. The rest of the methods (LBP, Gabor, Fourier, etc.) consist of feature vectors that are extracted from the cropped images. The features must be classified using a classifier. In these cases, we tested the following seven classifiers: KNN with 1, 3 and 5 neighbors, SVM with linear and RBF kernels and pattern recognition neural networks (ANN) with 15 and 30 hidden layers[5]. For these cases, we report in Table 3 the classifier that obtained the best accuracy.

According to the accuracy obtained by the 24 methods, we can distinguish in Fig. 4 three Levels: A) methods 1–3, B) methods 4–8, and C) methods 9–24 . Level A has a good accuracy (between 92.2% and 95.2%). Level B has a medium accuracy (between 83.8% and 86.4%) and Level C has a low accuracy (less than 72.6%).

The best performing pipeline was using Local Binary Pattern descriptors (LBP + SVM-Linear with 95.2% and LBP-ri + ANN-15 with 93.8%), they belong to Level A. These hand-engineered descriptors showed to be sensitive to very local textures, which are key in the X-ray defect detection dataset.

More generally, we observe that the plain use of features extracted from CNN networks pre-trained in ImageNet (VGG, AlexNet and GoogleNet), although a successful approach for recognition in natural images [44], showed a poor performance in our dataset (they belong to Level C). This is not surprising considering that the features required to tackle the X-ray dataset may differ substantially from those extracted from natural images. Therefore, it was expected that fine-tuning the CNN models in the X-ray dataset would be critical. This is reflected in the Xnet experiments, which showed reasonable results, and in which the pre-training on image net helped the accuracy ImageXnet (they belong to Level B). We attempted to fine-tune the larger CNN models on the X-ray dataset, but did

[5]In this case, we used function `patternnet` of Toolbox of Neural Networks of Matlab [27].

Table 3. Performance of the various bench-marked methods in the task for defect detection on our dataset of X-ray images. See Fig. 4 for further visualizations of these results.

Level	#	Method	Classifier	TP	TN	FP	FN	Pr	Re	η
	0	Ideal	–	7536	7536	0	0	1.0000	1.0000	1.0000
A	1	LBP	SVM-Linear	7072	7282	254	464	0.9653	0.9384	0.9524
	2	LBP-ri	ANN-15	6792	7339	197	744	0.9718	0.9013	0.9376
	3	BSIF	SVM-Linear	6870	7026	510	666	0.9309	0.9116	0.9220
B	4	ImageXnet	SoftMax	5686	7335	201	1850	0.9659	0.7545	0.8639
	5	sLBP	SVM-Linear	6077	6861	675	1459	0.9000	0.8064	0.8584
	6	Xnet	SoftMax	5523	7348	188	2013	0.9671	0.7329	0.8540
	7	BRISK	ANN-30	5215	7480	56	2321	0.9894	0.6920	0.8423
	8	HOG	ANN-30	5163	7474	62	2373	0.9881	0.6851	0.8384
C	9	SPARSE	Rec-Err	5070	5874	2554	1574	0.6650	0.7631	0.7261
	10	CLP	KNN-3	4357	6477	1059	3179	0.8045	0.5782	0.7188
	11	SURF	KNN-5	4301	6500	1036	3235	0.8059	0.5707	0.7166
	12	SIFT	KNN-3	4321	6421	1115	3215	0.7949	0.5734	0.7127
	13	INT	KNN-3	4294	6350	1186	3242	0.7836	0.5698	0.7062
	14	DCT	KNN-3	3371	6955	581	4165	0.8530	0.4473	0.6851
	15	VGG-2048	KNN-3	3608	6546	990	3928	0.7847	0.4788	0.6737
	16	SRC	Rec-Err	3259	6883	653	4277	0.8331	0.4325	0.6729
	17	VGG-F	KNN-5	3172	6861	675	4364	0.8245	0.4209	0.6657
	18	Gabor+	KNN-1	3302	6676	860	4234	0.7934	0.4382	0.6620
	19	VGG-19	KNN-3	3263	6644	892	4273	0.7853	0.4330	0.6573
	20	AlexNet	KNN-3	3025	6868	668	4511	0.8191	0.4014	0.6564
	21	Fourier	SVM-RBF	3009	6724	812	4527	0.7875	0.3993	0.6458
	22	GoogleNet	ANN-15	2458	6899	637	5078	0.7942	0.3262	0.6208
	23	Haralick	KNN-1	1608	7034	502	5928	0.7621	0.2134	0.5734
	24	Gabor	KNN-1	2901	5636	1900	4635	0.6042	0.3850	0.5664

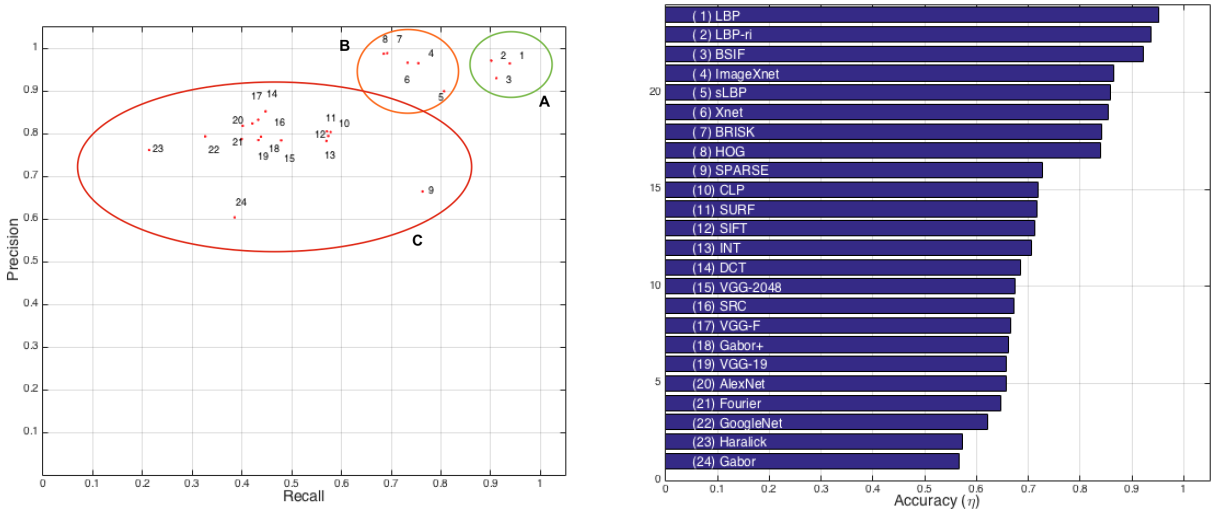

Figure 4. Precision, Recall and Accuracy for tested methods. The number of each point in Precision-Recall graphic corresponds to the number of the method in the bar graphic and the number of the row in Table 3.

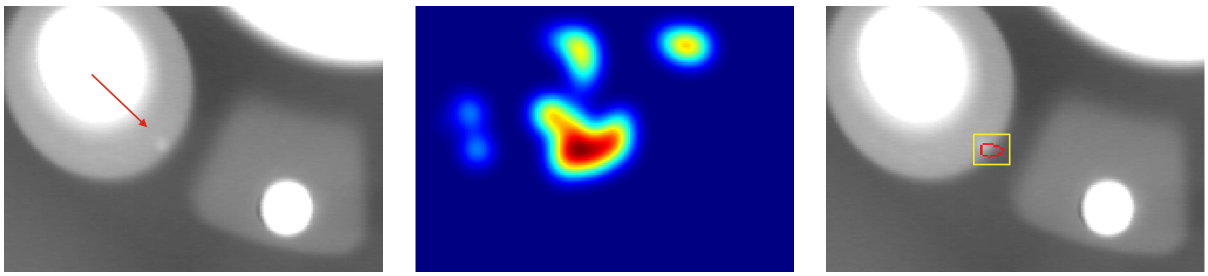

Figure 5. Defect detection in an X-ray image using a sliding-window approach: left) original image with a small defect (see arrow), middle) heat map after sliding-window, c) detection by thresholding the heat map.

not reach any improvement, most likely due to severe over-fitting.

Interestingly, methods based on sparse representations (SPARSE and SRC) did not show in our experiments a satisfactory performance (they belong to Level C). The best performance was achieved in both cases with large dictionaries, *i.e*. with higher redundancy, however, they probably increased the chance of over-fitting.

In order to illustrate the effectiveness of the best learned detector (LBP descriptor + SVM classifier), we implemented a very simple sliding-window strategy for a whole X-ray image that was not used in the learning set. Fig. 5 shows the obtained results. In this case, the size of the sliding-window is 32×32 pixels, *i.e*. the size of the cropped images of the dataset. We observe that the small defect present at the edge of the regular structure (see arrow in left image) could be detected perfectly. In this implementation, the size of the image is 140×200 pixels, and the step of the sliding-window process was 1 pixel. The heat-map of this detection, obtained by superimposing Gaussian-masks when a detection window is classified as defect is illustrated in the middle image. Evidently, a more robust sliding-windows strategy must be tested at different scales in which saliency maps can be used to pre-filter false detection and to reduce the search space of candidate sliding-windows. An evaluation on a broader data base is necessary.

Implementation details: We used Matlab implementations for these approaches as follows: Computer Vision System Toolbox for SURF and BRISK descriptors [26]; Neural Network Toolbox for ANN [27]; Toolbox Balu [32] for LBP, CLP, Haralick, Fourier, Gabor and BSIF; Library SPAMS for sparse representations [25]; Library VLFeat for HOG and SIFT descriptors [50], Library MatConvNet [51] for the deep learning models; and Library LibSVM for SVM classifier [7]. The code is available on our webpage [6].

6. Conclusions

In this paper, we present a comparative evaluation of 24 algorithms that can be used in the automated defect recognition in X-ray testing. We evaluate modern computer vision techniques, such as deep learning and sparse representations. These techniques had not been used in this kind of problems so far. We release a new dataset containing around 47.500 cropped X-ray images of 32×32 pixels with defects and no-defects in automotive components. We define an evaluation protocol that can be used to experiment on the dataset, we evaluate and compare 24 computer vision techniques (including deep learning, sparse representations, texture features among others). To the best knowledge of the authors, this is the first work in X-ray testing with an exhaustive evaluation of so many computer vision methods including deep networks.

We show in our experiments that the best performance was achieved by a simple LBP descriptor with a SVM-linear classifier obtaining 95.2% of accuracy (with 96.5% precision and 93.8% recall). These hand-engineered descriptors showed to be sensitive to very local textures, which are key in the X-ray defect detection dataset. Preliminary results with CNN models showed that features extracted from CNN networks pre-trained in ImageNet have a poor performance. However, a CNN model trained on X-ray images could improve the accuracy significantly (approximatly from 65% to 85%). We attempted to fine-tune the larger CNN models on the X-ray dataset, but did not reach any improvement, most likely due to severe over-fitting.

In the future, we will train our own deep learning network using a larger dataset, and we will test it on whole X-ray images using a sliding-windows strategy. It is worth noting, that this methodology can be applied to other kind of defects (such as discontinuities in the welding process [31], concrete defects [12], glass defects [6] or steel surface defects [23] among others) if we have a large enough database of representative images.

Acknowledgments

This work was supported by Fondecyt Grant No. 1161314 from CONICYT, Chile.

[6] See http://dmery.ing.puc.cl/index.php/material/

References

[1] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[2] H. Bay. SURF: speeded up robust features. In *ECCV'06: Proceedings of the 9th European conference on Computer Vision*, pages 404–417, Berlin, Heidelberg, May 2006. Springer-Verlag.

[3] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[4] J. Beyerer and F. P. León. Automated visual inspection and machine vision. In *Proc. of SPIE Vol*, volume 9530, pages 953001–1, 2015.

[5] M. Carrasco and D. Mery. Automatic multiple view inspection using geometrical tracking and feature analysis in aluminum wheels. *Machine Vision and Applications*, 22(1):157–170, 2011.

[6] M. Carrasco, L. Pizarro, and D. Mery. Visual inspection of glass bottlenecks by multiple-view analysis. *International Journal of Computer Integrated Manufacturing*, 23(10), Oct. 2010.

[7] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[8] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[9] R. Cogranne and F. Retraint. Statistical detection of defects in radiographic images using an adaptive parametric model. *Signal Processing*, 96:173–189, 2014.

[10] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.

[11] R. C. Gonzalez and R. E. Woods. *Digital image processing*. Prentice hall Upper Saddle River, New Jeresey, 2008.

[12] N. Gucunski et al. *Nondestructive testing to identify concrete bridge deck deterioration*. Transportation Research Board, 2013.

[13] M. Haghighat, S. Zonouz, and M. Abdel-Mottaleb. Cloudid: trustworthy cloud-based and cross-enterprise biometric identification. *Expert Systems with Applications*, 42(21):7905–7916, 2015.

[14] R. Haralick. Statistical and structural approaches to texture. *Proc. IEEE*, 67(5):786–804, 1979.

[15] J. Kannala and E. Rahtu. Bsif: Binarized statistical image features. In *21st International Conference on Pattern Recognition (ICPR2012)*, pages 1363–1366. IEEE, 2012.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. *NIPS*, pages 1106–1114, 2012.

[17] A. Kumar and G. Pang. Defect detection in textured materials using gabor filters. *IEEE Trans. on Industry Applications*, 38(2):425–440, 2002.

[18] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[19] Y. LeCun, L. Bottou, and Y. Bengio. Gradient-based learning applied to document recognition. In *Proceedings of the Third International Conference on Research in Air Transportation*, 1998.

[20] S. Leutenegger, M. Chli, and R. Y. Siegwart. BRISK: Binary robust invariant scalable keypoints. In *2011 International conference on computer vision*, pages 2548–2555. IEEE, 2011.

[21] W. Li, K. Li, Y. Huang, and X. Deng. A new trend peak algorithm with X-ray image for wheel hubs detection and recognition. In *Computational Intelligence and Intelligent Systems*, pages 23–31. Springer, 2015.

[22] X. Li, S. K. Tso, , X.-P. Guan, and Q. Huang. Improving automatic detection of defects in castings by applying wavelet technique. *IEEE Transactions on Industrial Electronics*, 53(6):1927–1934, 2006.

[23] H.-W. Liu, Y.-Y. Lan, H.-W. Lee, and D.-K. Liu. Steel surface in-line inspection using machine vision. In *First International Workshop on Pattern Recognition*. International Society for Optics and Photonics, 2016.

[24] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov. 2004.

[25] J. Mairal, F. Bach, J. Ponce, G. Sapiro, R. Jenatton, and G. Obozinski. Spams: Sparse modeling software, 2014. Software available on http://spams-devel.gforge.inria.fr.

[26] Matlab. *Computer Vision System Toolbox for Matlab: Users Guide*. Mathworks, 2016.

[27] Matlab. *Neural Network Toolbox for Matlab: Users Guide*. Mathworks, 2016.

[28] D. Mery. Crossing line profile: a new approach to detecting defects in aluminium castings. *Proceedings of the Scandinavian Conference on Image Analysis (SCIA 2003), Lecture Notes in Computer Science*, 2749:725–732, 2003.

[29] D. Mery. Automated radioscopic testing of aluminum die castings. *Materials Evaluation*, 64(2):135–143, 2006.

[30] D. Mery. Automated detection in complex objects using a tracking algorithm in multiple X-ray views. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pages 41–48. IEEE, 2011.

[31] D. Mery. Automated detection of welding defects without segmentation. *Materials Evaluation*, 69(6):657–663, 2011.

[32] D. Mery. BALU: A Matlab toolbox for computer vision, pattern recognition and image processing, 2011. Software available at http://dmery.ing.puc.cl/ index.php/balu.

[33] D. Mery. *Computer Vision for X-Ray Testing*. Springer, 2015.

[34] D. Mery and D. Filbert. Automated flaw detection in aluminum castings based on the tracking of potential defects in a radioscopic image sequence. *IEEE Trans. Robotics and Automation*, 18(6):890–901, December 2002.

[35] D. Mery, V. Riffo, U. Zscherpel, G. Mondragón, I. Lillo, I. Zuccar, H. Lobel, and M. Carrasco. GDXray: The database of X-ray images for nondestructive testing. *Journal of Nondestructive Evaluation*, 34(4):1–12, 2015.

[36] Y. Mu, S. Yan, Y. Liu, T. Huang, and B. Zhou. Discriminative local binary patterns for human detection in personal album. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, pages 1–8, 2008.

[37] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[38] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[39] C. Pieringer and D. Mery. Flaw detection in aluminium die castings using simultaneous combination of multiple views. *Insight*, 52(10):548–552, 2010.

[40] L. Pizarro, D. Mery, R. Delpiano, and M. Carrasco. Robust automated multiple view inspection. *Pattern Analysis and Applications*, 11(1):21–32, 2008.

[41] F. Ramírez and H. Allende. Detection of flaws in aluminium castings: a comparative study between generative and discriminant approaches. *Insight-Non-Destructive Testing and Condition Monitoring*, 55(7):366–371, 2013.

[42] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1998.

[43] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg, and L. Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[44] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813, 2014.

[45] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR abs/1409.1556*, 2014.

[46] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, 15:1929–1958, June 2014.

[47] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR 2015*, 2015.

[48] Y. Tang, X. Zhang, X. Li, and X. Guan. Application of a new image segmentation method to detection of defects in castings. *The International Journal of Advanced Manufacturing Technology*, 43(5-6):431–439, 2009.

[49] I. Tosic and P. Frossard. Dictionary learning. *Signal Processing Magazine, IEEE*, 28(2):27–38, 2011.

[50] A. Vedaldi and B. Fulkerson. Vlfeat: An open and portable library of computer vision algorithms. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1469–1472. ACM, 2010. Software available on http://www.vlfeat.org/.

[51] A. Vedaldi and K. Lenc. MatConvNet – Convolutional Neural Networks for Matlab, 2014. Software available on http://www.vlfeat.org/.

[52] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):210–227, 2009.

[53] X. Zhao, Z. He, and S. Zhang. Defect detection of castings in radiography images using a robust statistical feature. *JOSA A*, 31(1):196–205, 2014.

[54] X. Zhao, Z. He, S. Zhang, and D. Liang. A sparse-representation-based robust inspection system for hidden defects classification in casting components. *Neurocomputing*, 153(0):1 – 10, 2015.