An Application of Machine Learning and Statistics to Defect Detection

R. Cucchiara[1], P. Mello[2], M. Piccardi[3], F. Riguzzi[3]

[1] Dipartimento di Scienze dell'Ingegneria, Università di Modena, Italy,

Via Campi 213/b, 41100 Modena, Italy - e-mail: rita.cucchiara@unimo.it

[2] D.E.I.S., Università di Bologna, Italy,

V.le Risorgimento 2, 40136 Bologna, Italy - e-mail: pmello@deis.unibo.it

[3] Dipartimento di Ingegneria, Università di Ferrara, Italy

Via G. Saragat 1, 44100 Ferrara, Italy - e-mail: {mpiccardi, friguzzi}@ing.unife.it

Abstract. We present an application of machine learning and statistics to the problem of distinguishing between defective and non-defective industrial workpieces, where the defect takes the form of a long and thin crack on the surface of the piece. From the images of pieces a number of features are extracted by using the Hough transform and the Correlated Hough transform. Two datasets are considered, one containing only features related to the Hough transform and the other containing also features related to the Correlated Hough transform. On these datasets we have compared six different learning algorithms: an attribute-value learner, C4.5, a backpropagation neural network, NeuralWorks Predict, a k-nearest neighbour algorithm, and three statistical techniques, linear, logistic and quadratic discriminant. The experiments show that C4.5 performs best for both feature sets and gives an average accuracy of 93.3 % for the first dataset and 95.9 % for the second dataset.

1 Introduction

We present an application of machine learning and statistics to a problem of Automated Visual Inspection (AVI) that consists of automatically inspecting the integrity of metallic industrial workpieces. The aim is to classify each piece as defective or non-defective depending on whether it contains or not surface defects, visible only under UV light. The surface defect is a crack that is visible under UV light as a bright, thin and roughly rectilinear shape.

In order to recognize cracks, a set of visual primitives has been selected for characterizing the images of pieces. In this way, each image is described by a set of numerical attributes and machine learning can be applied in order to find a classifier for new images.

In particular, we use the Hough transform (HT) that has been proposed in the literature of image analysis for detecting straight lines [9]. The HT transforms the image space into another two-dimensional space (called Hough space) where each local maximum point corresponds to a straight edge in the image space. Moreover, another transformation is used, the Correlated Hough transform (CHT), which has the specific aim of detecting shapes that are bright, rectilinear and thin [3]. The CHT transforms an image from the Hough space to the Correlated Hough space where each local maximum point represents a pair of close, straight edges in the image.

In order to test the effectiveness of these different primitives in classification, we have considered two different datasets, one containing features from the Hough and the Correlated Hough space, and another containing features from the Hough space only.

On the two datasets, we have compared the effectiveness of six different learning algorithms: an attribute-value learner, C4.5, a backpropagation neural network, NeuralWorks Predict, a k-nearest neighbour algorithm and three statistical techniques, linear, logistic and quadratic discriminant.

The paper is divided as follows: the next section introduces the specific application. Section 3 discusses the adopted visual primitives. Section 4 discusses the results of experiments, providing a comparative analysis among the different algorithms. Section 5 presents a discussion of the time complexity of the approach and of its applicability in domains with more examples and tighter time constraints. Section 6 presents related works and a comparison with results obtained in the StatLog project on image datasets. Finally, the last section provides our conclusions.

2 Defect Detection

The application goal is visual integrity inspection of metallic industrial workpieces and in particular the location of surface and subsurface defects in ferromagnetic materials.

This goal cannot be reached by normal, visible-light inspection but is usually accomplished by adopting a “Magnetic-Particle Inspection” technique (MPI) [10]. First, the piece is magnetised and dipped in a water suspension of fluorescent ferromagnetic particles; then, it is exposed under ultraviolet light and examined by a human inspector. When surface or subsurface defects are present, they produce a leakage field that attracts and concentrates the ferromagnetic particles. The human eye can then easily perceive defects, since ultraviolet light greatly enhances fluorescence. Off-the-shelf CCD cameras and frame grabbers are used in order to acquire the images.

Examples of images with cracks are shown in figures 1, 2 and 3. Figure 1 shows a whole image, while figures 2 and 3 show two cracks in detail, more and less evident respectively.

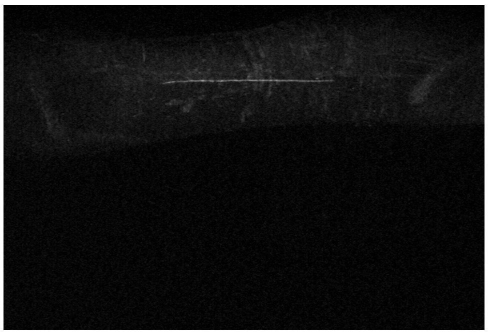

Figure 1. Image with a crack.

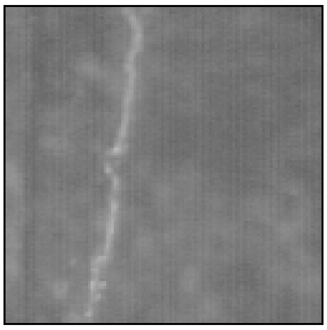

Figure 2. Detail of an evident crack.

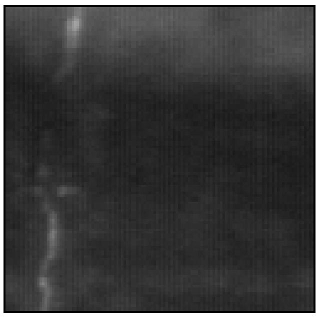

Figure 3. Detail of a less evident crack.

3 Classification by Visual Primitives

The defect shape was known a-priori by means of a qualitative model provided by human inspectors. They defined it as a “thin, roughly rectilinear and very bright shape”.

On the basis of this rather generic model, we elicited a set of measurable visual properties that are associated with certain aspects of the qualitative model:

- *bright shape* $\rightarrow$ high local gradient of luminosity around the edges;
- *rectilinear* $\rightarrow$ with two main edges approximately straight;
- *thin* $\rightarrow$ with an upper-bounded distance between the two main edges.

Once the visual properties are elicited, a set of quantitative image operators able to reflect them must be defined. Usually, the approach consists of defining a redundant set of image operators, or features, each being somehow related to one or more visual properties, that will be later selected in a machine learning phase. The choice of the initial feature set is critical since the information lost at this step cannot be recovered later.

To this aim, we defined and compared two different feature sets, motivated by opposite rationales: in the first set, we included a specialized primitive called Correlated Hough transform (CHT [3]), which has been proposed for detection of objects corresponding exactly to our model; in the second set, we used only image operators of general use. The two feature sets reflect a different control of the visual aspects of the problem, the first one calling for the insight on image operators typical of a computer vision specialist, while the second requires just the use of well-known image operators.

Both feature sets include the Hough transform (HT), whose essential characteristics will now be described. The HT has been proposed in the computer vision literature to detect straight lines [9]. It consists of a space transformation from the image space to a 2-coordinate parameter space: "collinear" points forming a straight line segment in the image space are collected into a single point of the parameter space, where the point's first co-ordinate, ϑ, is the slope of the straight line and the second co-ordinate, ρ, is its distance from the origin. Each point in the Hough space has a value which is exactly the number of collinear points in the straight line segment; thus, the longer the line segment, the higher is the point's value in the Hough space. Furthermore, in this work we adopted a refined version of the Hough transform, called gradient-weighted Hough transform (GWHT, [9]) in which each collinear point is weighted by its luminosity gradient. Therefore, peaks in the Hough space (i.e. points with high values) correspond to the existence of straight, bright lines in the image space, or, in other words, the problem of detecting lines in the image space is simplified into the task of detecting peaks in the Hough space.

In the inspected images, a crack has two edges with similar gradient magnitude, the same direction but opposite orientation; since the crack is thin, the distance between the two edges is upper-bounded. Therefore, two peaks must be detected in the Hough space, with similar values and their ρ, ϑ parameters mutually constrained. In alternative to the separate detection of these two peaks, it is possible to exploit the Correlated Hough transform. The CHT performs a post-processing of the GWHT Hough space by correlating the area where the first peak is detected with the one where the second peak should be located: if it is actually present, the resulting correlation value is very high and can be easily detected. The CHT has been proven robust to non-ideality and noise, since the detection after correlation is more reliable than the detection of the two separate peaks in the Hough space. However, the CHT itself is not sufficient for detecting cracks which differ strongly from their ideal aspect, and therefore many other features were also considered.

The dataset based on the CHT (called CH dataset) contains the following features:

1. *CH (Correlated_Hough_Peak)*: this is the maximum value in the Correlated Hough space; its ρ, ϑ co-ordinates (where $\vartheta \in [0, \pi]$), which correspond to the parameters of a straight line in the image located on the crack, in the case that a crack is present.

2. *H1 (First_Hough_Peak)*: this is the value in the point of the Hough space with the same co-ordinates ρ, ϑ as the correlated Hough peak, where the first peak is formed if a crack is present.
3. *H2 (Second_Hough_Peak)*: the peak in the Hough space between π and 2π at the ideal point where the second straight edge should be found.
4. *H22 (Second_Hough_Average)*: this feature is CH divided by H1; it measures by how much the correlation operation increases the evidence of the crack with respect to the uncorrelated Hough space.
5. *Thickness*: the mutual distance between H1 and H2. It represents the object thickness.
6. *Number_of_Points*: the number of voting points accumulated in H1, which estimates the edge length.
7. *Average_Vote*: the average "vote" of the voting points, i.e. the average luminosity gradient of each point voting for H1 (computed by dividing H1 by the Number_of_Points); it measures the average luminosity gradient along the crack profile.
8. *Average_Image_Gradient*: the average luminosity gradient of the image; it is a different property with respect to the others, since it is global, meaning that it is an overall feature of the whole image. It might be used by the classifier as a “normalization” attribute, since images with low values of the average gradient have proportionally lower CH and Hough space values.

Operationally, we acquire images with relevant views of the mechanical piece and for each image we compute the CHT. Then, we detect the CHT maximum (the CH feature) and record a case with CH and the other associated feature values. We then detect all the points of the correlated Hough space whose value is greater than or equal to an assigned percentage of the maximum (75% was used in the experiments), and record a case for each of them; this is done in order to catch multiple cracks that can be present in a single image. After acquiring the cases, we pre-classify each of them into the two categories of *Defect* or *NoDefect* by checking manually if the straight line segment corresponding with the case was located on a real crack in the image or not. For the CH dataset, the runtime required to extract all the cases from an image of 768 x 576 pixels was about 3.4 s on a 500 MHz Pentium III PC.

In the approach followed the CHT plays a major role, since the CH maximum is the feature that determines the position where the crack may be located. However, the CHT is a highly specialized operator, and it is interesting to approach the problem with a feature set with additional standard features, and compare the performance of the resulting classifiers.

Therefore, in the second dataset set (called H1 H2 dataset) we excluded the CH value and included the following features:

1. *H1*: the value of the Hough maximum in the range $\vartheta \in [0,\pi]$, where the first peak is formed in the case that a crack is present; its ρ, ϑ co-ordinates correspond to the parameters of a straight line located on one edge of the crack.
2. *H2*: the value of the Hough maximum in the range $\vartheta \in [\pi, 2\pi]$, where the second peak is formed if a crack is present; its ρ', ϑ' co-ordinates correspond to the parameters of a straight line located on the other edge of the crack. However, if multiple cracks are present, H1 and H2 may not be associated with the same crack.
3. *Number_of_Votes*: the sum of the number of image points that were transformed into H1 and H2.
4. *Distance*: the mutual distance between H1 and H2 in the Hough space. It represents the object thickness if H1 and H2 correspond to the same crack.

5. *Delta_rho*: the $|\rho' - \rho|$ value, and
6. *Delta_theta*: the $|\vartheta' - \vartheta - \pi|$ value. *Delta_rho* and *Delta_theta* express the distance between the two peaks along the ρ and ϑ directions, respectively. In the case of a same real crack, *Delta_theta* should be close to 0 and *Delta_rho* upper bounded. *Delta_rho* and *Delta_theta* are related to *Distance* by the following formula :

$$Distance=\sqrt{Delta\_rho^2 + Delta\_theta^2} \ .$$

7. *Delta_product*: the product *delta_rho * delta_theta*. It correlates the *Delta_rho* and *Delta_theta* values, expecting small values for the product in the case of a same real crack.
8. *Average_Image_Gradient*: The average luminosity gradient of the image.

Since there is not an explicit correlation operation between H1 and H2, we also added some basic arithmetic functions of the H1 and H2 values:

9. *Product*: the product H1 * H2: should be high in the case of a real crack (about the square of each of the two values).
10. *Ratio*: the ratio H1 / H2: should be close to 1 in the case of a real crack.
11. *Sum*: the sum H1 + H2: should be high in the case of a real crack (about double each of the two values).
12. *Difference*: the difference H1 - H2: should be close to 0 in the case of a real crack.

These arithmetic functions are simply combinations of other features and as such may be considered redundant; yet they have been explicitly included in this feature set since they are related with the model and may improve the classifiers' performance should the classifier not explore linear or quadratic combinations or ratios of the feature values.

Operationally, we acquire images with relevant views of the mechanical piece and for each image we compute the Hough space with the GWHT. Then, we detect the H1 and H2 maxima and record them in a case with the other associated feature values. We then repeat the process for all the points of the Hough space in the range [0, π] and [π, 2π] whose value is greater or equal to an assigned percentage of H1 and H2, respectively, and record a case for each couple; this is done in order to catch multiple cracks that can be present in a single image. After acquiring the cases, we pre-classify each of them into the two categories of *Defect* or *NoDefect* by checking manually if the straight-line segments corresponding with H1 and H2 were located on a same real crack. For the H1 H2 dataset, the runtime required to extract all the cases from an image of 768 x 576 pixels was about 3.1 s on a 500 MHz Pentium III PC.

4 Experiments

We have experimented and compared three different machine learning techniques: attribute-value learning, backpropagation neural networks and instance-based learning. Moreover, due to the numeric nature of all the attributes, we have also used statistical techniques in order to compare their performance with that of machine learning tools.

For attribute-value learning we have used C4.5 [14] which is able to learn both decision trees and rules. For backpropagation neural networks, we have used a commercial system, Predict by NeuralWorks[1]. For instance-based learning, we have used the k-nearest neighbour algorithm that is included in the Weka library of machine learning software[2]. As regards statistical techniques, we have used the algorithms Discrim, Logdisc and Quadisc, developed

[1] A demo version of Predict can be found at http://www.neuralware.com/ .

[2] The Weka machine learning software can be downloaded from http://www.cs.waikato.ac.nz/ml/weka/index.html

under the Statlog project [11], that implement respectively the linear discriminant, the logistic discriminant and the quadratic discriminant.

In the following, we first give a brief description of each algorithm and then we present the results of experiments.

4.1 Discrim

Discrim finds a linear discriminant, i.e., a hyperplane in the p-dimensional space of the attributes. Given the values of the attributes of a new pattern, its class is found by looking at the position of the corresponding point with respect to the hyperplane.

The hyperplane equation is found on the assumption of normal probability distribution: the attribute vectors for the examples of class A_i are independent and follow a certain probability distribution with probability density function (pdf) f_i. A new point with attribute vector x is then assigned to that class whose probability density function $f_i(\mathbf{x})$ is greatest. This is a maximum likelihood method. The distributions are assumed normal (or Gaussian) with different means but the same covariance matrix. The probability density function of the normal distribution is

$$f_i(\mathbf{x}) = \frac{1}{\sqrt{|2\pi\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_\mathrm{i})' \Sigma^{-1} (\mathbf{x}-\mu_\mathrm{i}) \right) \tag{1}$$

where μ is a p-dimensional vector denoting the theoretical mean for class *i* and Σ, the theoretical covariance matrix, is a $p \times p$ matrix that is necessarily positive definite. In this case the boundary separating the two classes, defined by the equality of the pdfs, can be shown to be a hyperplane that passes through the mid-point of the two means. Its equation is

$$\mathbf{x}'\Sigma^{-1}(\mu_1 - \mu_2) - \frac{1}{2}(\mu_1 + \mu_2)'\Sigma^{-1}(\mu_1 - \mu_2) = 0 \qquad (2)$$

where μ_i is the population mean for class A_i. When using this formula for classification the exact distribution is usually not known and the parameters must be estimated from the available sample. With two classes, if the sample means are substituted for μ_i and the pooled sample covariance matrix for Σ, then Fisher's linear discriminant [8] is obtained. The covariance matrix for a dataset with n_i examples from class A_i is

$$S_i = \frac{1}{n_i - 1} X^T X - \bar{\mathbf{x}}^T \bar{\mathbf{x}} \qquad (3)$$

where X is the $n_i \times p$ matrix of attribute values and $\bar{\mathbf{x}}$ is the p-dimensional row vector of attribute means. The *pooled covariance matrix S* is

$$S = \frac{\sum (n_i - 1) S_i}{n - q} \qquad (4)$$

where the summation is over all the classes and *(n-q)* is chosen to make the pooled covariance matrix unbiased.

4.2 Quadisc

Quadisc performs a quadratic discrimination. Quadratic discrimination is mostly similar to linear discrimination, with the difference that the surface separating the two regions is quadratic. This means that the discriminating function will contain not only the attributes but also their squares and the products of two attributes. With respect to the case of probability maximization seen for Discrim, if we remove the assumption that the normal distributions have the same covariance matrix S, we obtain a quadratic surface, for example an ellipsoid or a hyperboloid.

The simplest quadratic discrimination function for a class is defined as the logarithm of the corresponding probability density function and is given by equation 5 in the case of differing prior probabilities. The suffix i is used to indicate class A_i.

$$\log \pi_i f_i(\mathbf{x}) = \log \pi_i - \frac{1}{2}\log(\det(\Sigma_i)) - \frac{1}{2}(\mathbf{x}-\mu_i)' \Sigma_i^{-1}(\mathbf{x}-\mu_i) \qquad (5)$$

In this equation π_i stands for the prior probability of class A_i. As before, the means and covariance matrix are substituted by their sample counterparts obtained from the training set. In the same way, π_i is substituted by the sample proportion of class A_i examples. For classification, the discriminant is calculated for each class and the one giving the highest value is chosen.

The most frequent problem with quadratic discriminants arises when some attribute has zero variance in one class, for then the covariance matrix cannot be inverted. One way to avoid this problem is to add a small positive constant term to the diagonal terms in the covariance matrix (this corresponds to adding random noise to the attributes).

4.3 Logdisc

Logdisc performs a logistic discrimination. As linear discriminants, a logistic discriminant consists of a hyperplane separating the classes in the best possible way, but the criterion used to find the hyperplane is different. The method adopted in this procedure is to maximize a conditional probability. In theory, when the attributes have a normal distribution with equal covariances and are independent from each other, linear and logistic discriminants are equivalent. Different results are obtained when these hypotheses are not satisfied.

The method described here is partially parametric, as the actual pdfs for the classes are not modelled, but rather, the ratio between them. In particular the logarithms of the ratios of the probability density functions for the classes are modelled as linear functions of the attributes. Thus, for two classes,

$$\log \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} = \alpha + \beta' \mathbf{x} \qquad (6)$$

where α and the p-dimensional vector β are the parameters of the adopted model and must be estimated. The case of normal distribution is a special case in which these parameters are functions of the prior probabilities, the class means and the common covariance matrix.

The parameters are estimated by maximum conditional likelihood. The model implies that, given attribute values $\mathbf{x}$, the conditional class probabilities for classes A_1 and A_2 take the forms:

$$P(A_1 \mid \mathbf{x}) = \frac{\exp(\alpha + \beta' \mathbf{x})}{1 + \exp(\alpha + \beta' \mathbf{x})} \qquad (7)$$

$$P(A_2 \mid \mathbf{x}) = \frac{1}{1 + \exp(\alpha + \beta' \mathbf{x})} \qquad (8)$$

Given independent samples for the two classes, the parameters are estimated by maximizing the probability:

$$L(\alpha, \beta) = \prod_{\{A_1, sample\}} P(A_1 \mid \mathbf{x}) \prod_{\{A_2, sample\}} P(A_2 \mid \mathbf{x}) \qquad (9)$$

Iterative methods have been proposed in order to estimate the parameters, for example in [2] and [4]. Since in practice there is often little difference between the logistic and linear discriminants, the latter are taken as a starting point for the former.

4.4 k-nearest neighbour

In instance based learning, training examples are used directly for classifying unseen examples, no explicit description of the target concept is built. In order to classify an unseen example, the distance of the unseen example from each training example is computed (according to a chosen distance function) in order to determine the class of the instance. In the k-nearest neighbour algorithm, the unseen example is assigned the most common class value in the set of k-nearest training examples.

In our experiments we have chosen k=1 in order to compare our results with those of the Statlog project (therefore in the following we will refer to nearest-neighbour as NN). Moreover, a standard Euclidean distance was used where the square of each attribute difference is divided by the standard deviation of the attribute in the training set, in order to take into account the different scales of the various attributes:

$$d(x', x'') = \sqrt{\sum_{i=1}^{n} \left(\frac{x'_i - x''_i}{\sigma i} \right)^2} \qquad (10)$$

The implementation of NN which was used for the experiments is the IB1 algorithm from the Weka library of machine learning software.

4.5 NeuralWorks Predict

Predict by NeuralWorks is a system for training multi-layer neural nets. Predict uses an adaptive gradient learning rule which is a form of back-propagation. Predict does not start from a fixed network architecture but uses a constructive method for determining a suitable number of hidden nodes. This constructive method is referred to as "Cascade Learning" [5] and is loosely characterized by the fact that hidden nodes are added either one or a few at a time. New hidden nodes have connections from both the input buffer and the previously established hidden nodes. Construction is stopped when performance on an independent test set shows no further improvement.

4.6 C4.5

C4.5 [14] is a system for learning rules and decision trees. Its peculiarity arei the heuristics it adopts in order to select the test to perform at each step. These heuristics are based on the notion of entropy from information theory which represents the amount of “dis-uniformity” of examples in the training set with respect to the class attributes: at each step the test chosen is that which makes the resulting subsets as uniform as possible with respect to the class attribute, thus ultimately selecting subsets containing examples from only one, or a small number, of classes.

4.7 Results

All systems have been tested on the CH and H1 H2 datasets employing 10-fold cross validation. Both datasets contain 317 cases of which 67 belong to the Defect class and 250 to the NonDefect class. The spread of attribute values is larger for the Defect class.

NN and Predict were run on a 500MHz Pentium III PC while the other algorithms were run on a Sun Sparcstation 10. The runtime for learning each classification algorithm was between a few hundredths to a few tenths of a second, while the classification time for each piece was in the order of milliseconds. Therefore these times are negligible with respect to the feature computation time in the overall inspection process. The industrial process we considered allows for a maximum inspection time of 20 s, thus fulfilling the inspection time constraint.

Table 1. Average accuracies

	Discrim	Logdisc	Quadisc	NN	Predict	c4.5 tree	c4.5 rules
CH	0.853	0.857	0.853	0.885	0.873	0.959	0.959
H1 H2	0.855	0.928	0.316	0.845	0.864	0.933	0.933

Table 2. Total false negative (FN) and false positive (FP) errors

	Discrim		Logdisc		Quadisc		NN		Predict		c4.5 tree		c4.5 rules	
	FN	FP	FN	FP	FN	FP	FN	FP	FN	FP	FN	FP	FN	FP
CH	37	9	36	9	31	15	37	9	15	25	6	7	6	7
H1 H2	26	19	14	9	40	177	28	24	3	40	13	8	12	9

Table 3. Values for the t statistics for the CH dataset

	Logdisc	Quadisc	NN	Predict	c4.5 tree	c4.5 rules
Discrim	1.00	0.00	1.00	0.46	**2.50**	**2.50**
Logdisc		0.17	0.90	0.38	**2.52**	**2.52**
Quadisc				0.62	**3.61**	**3.61**
NN				0.49	**2.53**	**2.53**
Predict					**2.67**	**2.67**
c4.5 tree						0.00

Table 4. Values for the t statistics for the H1 H2 dataset

	Logdisc	Quadisc	NN	Predict	c4.5 tree	c4.5 rules
Discrim	**1.75**	**4.20**	0.44	0.34	**1.59**	**1.69**
Logdisc		**5.49**	**2.51**	1.10	0.15	0.16
Quadisc			**3.96**	**3.65**	**7.14**	**6.92**
NN				0.34	**1.59**	**1.69**
Predict					0.84	0.86
c4.5 tree						0.00

Table 1 shows the average accuracies of the classification algorithms for both datasets. Table 2 shows the total number of false negative and false positive errors summed over the ten folds. False negatives are defective pieces that are classified as non-defective and false positives are non-defective pieces that are classified as defective. It is important to distinguish between these two types of errors because the damage that derives from a false negative is much higher than that deriving from a false positive. Therefore, we should prefer an algorithm that minimizes the number of false negatives.

In order to evaluate if the accuracy differences between algorithms are significant, we have computed a 10-fold cross-validated paired *t* test for every pair of algorithms (see [1] for an overview of statistical tests for the comparison of machine learning algorithms).

This test is computed as follows. Given two algorithms A and B, let $p_A^{(i)}$ (and respectively $p_B^{(i)}$) be the observed proportion of test examples misclassified by algorithm A (respectively B) in trial i. If we assume that the 10 differences $p^{(i)}=p_A^{(i)}-p_B^{(i)}$ are drawn independently from a normal distribution, then we can apply Student *t* test by computing the statistic

$$t = \frac{\bar{p} \cdot \sqrt{n}}{\sqrt{\frac{\sum_{i+1}^{n} (p^{(i)} - \bar{p})^2}{n-1}}} \tag{10}$$

where n is the number of folds (10) and $\bar{p}$ is

$$\bar{p} = \frac{1}{n} \sum_{i=1}^{n} p^{(i)} \tag{11}$$

In the null hypothesis, i.e. that A and B have the same accuracy, this statistic has a *t* distribution with n-1 (9) degrees of freedom. If we consider a probability of 90%, then the null hypothesis can be rejected if

$$|t| > t_{9,0.90} = 1.383 \tag{12}$$

Table 3 shows the values of the statistic for the CH dataset, while table 4 shows the values of the statistic for the H1 H2 dataset. The value of the statistic for algorithms A and B can be found at the crossing of line A with column B. The numbers in bold are those that provide a probability of 90% or more of rejecting the null hypothesis.

From these tables it can be seen that, for the CH dataset, the accuracy ranking of algorithms is: C4.5 (rules and trees), NN, Predict, Logdisc, Quadisc and Discrim. However, the accuracy difference is significant only between C4.5 and all the other algorithms and between NN and Quadisc.

On the H1 H2 dataset, the ranking of algorithms is: C4.5 (rules and trees), Logdisc, Predict, Discrim, NN and Quadisc. The accuracy difference is significant only between the best performing algorithms, C4.5 and Logdisc, and all the others apart from Predict.

When comparing the algorithm in terms of accuracy, we can conclude that, for both datasets, the best overall accuracy has been obtained by C4.5 both for the case of trees and rules. The comparison of machine learning and statistical techniques in terms of accuracy shows that C4.5 performs better than statistical techniques for the CH dataset, while on the H1 H2 dataset Logdisc and C4.5 are equivalent. As can be seen, the CH feature is very important because it leads to more accurate classifiers for all systems with the exclusion of Logdisc and Discrim. As regards the number of false negatives, C4.5 yields the lowest number of them for the CH dataset, while for the H1 H2 dataset the lowest number is given by Predict.

As regards the computation time, it was not necessary to perform a detailed comparison of the various algorithms because the learning and classification times of all algorithms were negligible if compared to the time required to compute the features.

These results show that machine learning tools can be preferred over k-nearest neighbour and statistical classifiers for the application considered.

5 Time complexity analysis

The approach presented can also be applied to more difficult industrial inspection problems requiring a larger number of examples in the training set, or to problems with tighter constraints on inspection time. In order to solve these problems, we must first distinguish between two main time costs: an off-line cost, which is the time spent once for all to generate the classifier, and an on-line cost, which is the time needed for classifying each produced piece.

The off-line cost includes two main terms: the time required to pre-classify the training set and that required to train the classifier from the training set. In the absence of an automated inspection process, teams of human experts manually verify a sample of the whole production. In order to automate the inspection process, for each piece it is necessary to compute the features and to store their values together with the expert's classification value. The time required to compute the features is negligible with respect to human classification time.

As regards the time to train the classifier, it is negligible with respect to the time to pre-classify the training set. This can be inferred from the fact that it was negligible in our case, where we had 317 examples, and from the fact that the time complexity of all classifiers is linear with respect to the number of examples in the training set except for NN for which it is constant.

On-line costs also include two main terms: the time required to compute the visual features' values for each piece, and the time required to run the classifier. In our application, the first term was largely the most relevant since computing the features takes between 3.1 and 3.5 s.

and running the classifier a few ms. Both terms are independent from the number of examples in the training set except for the classification time for NN which is linear in the number of examples. Thus the classification time may become relevant only if NN is used.

Therefore, the limits to the number of examples that can be used for training are given by the human classification time and by the on-line classification time if NN is used.

Similarly, if a tighter constraint on the inspection time is given, the limits are given by the time required to compute the features and by the on-line classification time if NN is used.

Another important issue is the possibility of changes in the nature of defects. Over time, in fact, it may happen that the type of defects changes because of variations in the production system. To this purpose, the performance of the classifier must be monitored by manually classifying a sample of the pieces produced. When the classification errors overcome an assigned threshold, the classifier must be updated. Of all the learning algorithms, only NN is an incremental algorithm where the new classifier can be obtained from the old one, while for the others the learning process must be repeated. However, as shown above, the learning time for a few hundred examples is in the order of a few hundredths of a second, and therefore re-learning does not pose any real kind of limitation to the detection system.

6 Related works and comparison with the Statlog Project

Machine learning has been widely exploited for object classification in computer vision. Learning is often essential for defining an effective classifier in the case of unstructured objects or shapes, which are difficult to model in terms of geometric, topologic or other metric features. Examples of the use of learning in computer vision are, for instance, recognition of hand gestures, landscape inspection, medical images analysis, and appearance-based recognition [1,6,13,12]. However, the most comprehensive work concerning the use of learning for object classification in computer vision is the StatLog project [11]. StatLog includes several classification algorithms, covering machine learning, neural and statistical techniques. The algorithms are compared against several different classification tasks, or datasets, nine of which consists of classifying images (Dig44, KL, Vehicle, Letter, Chrom, SatIm, Segm, Cut20, Cut50). As reported in the StatLog results in [11], the ranking of classifiers in terms of error rates varies with the image classification task. As stated in the analysis of results still in [11], some of these tasks mainly address classification of pixel areas, while others address classification of derived features computed from the pixel values. These tasks are very different in nature, and this may be a major reason for the different ranking of classifiers' error rates.

The NN classifier achieves on average the best error rate. However, one pitfall of the NN method is the fact that it assigns all the variables the same relevance, and this may be the reason for the few exceptions (Vehicle and Segm); in these cases, many other algorithms outperform NN, including C4.5.

Quadisc achieves on average the best error rates for those image datasets considered as object recognition tasks (Dig44, KL, Vehicle, Letter, Chrom), while it performs badly on average on the image datasets considered as segmentation tasks (SatIm, Segm, Cut20, Cut50).

The machine learning algorithm C4.5 tends to demonstrate good performance on the segmentation tasks (Segm, Cut20, Cut50) and in particular, it largely outperforms Quadisc on the Cuts dataset. On the other datasets, C4.5 ranks on average positions.

The application we considered in this paper can hardly be considered as solely object recognition or solely segmentation: it can be considered object recognition since it uses object-level features like the Hough transform, while at the same time it can be regarded as segmentation since objects are not segmented in advance. In this application, C4.5 achieves the best error rates for both feature sets, while NN has an uneven behaviour, ranking in the second position for the CH dataset and second last for the H1 H2 dataset. This relative ranking is the same as that of the StatLog's Vehicle and Segm datasets, but not of the StatLog's image datasets on average. Instead, Quadisc assesses significantly worse performance, similar to that of the other classifiers for the CH feature set and drastically lower for the H1 H2 one. This result is in good accordance with those from the StatLog segmentation datasets.

7 Conclusion

We have presented an application of machine learning and statistics to the problem of recognizing surface cracks on metallic pieces. In order to learn from the images of the pieces, we have identified a set of visual features for characterizing each image that has been expressly designed for the recognition of straight lines and rectilinear shapes.

In order to test the effectiveness of these various features on classification, we have considered two different sets, one containing features from the Hough and the Correlated Hough space, and another one containing features from the Hough space only.

Various machine learning and statistical techniques have been applied to the problem. As regards machine learning, we have employed an attribute value learner, C4.5, a neural network trainer, NeuralWorks Predict and a k-nearest neighbour algorithm. As regards statistical techniques, we have employed linear, logistic and quadratic discriminants.

The results of the experiments show that, of the two feature sets, the one containing the CHT leads to more accurate classifiers for all learning methods apart from Logdisc and Discrim, thus confirming the usefulness of highly specialized operators for Computer Vision.

Among all systems, C4.5 had the best accuracy, significantly higher than that of the other systems on the CH dataset, while on the H1 H2 dataset it was significantly higher only with respect to Discrim, Quadisc and NN. As regards the computation time, the learning times of all algorithms were comparable and they were negligible if compared to the time required to compute the features.

The accuracy results differ from those obtained in the Statlog project, where NN was the best classifier for images. This can be due to the fact that NN treats (normalized) variables with equal weight, while in the datasets considered the relevance of the various variables differs. Another major result of our experiments is the evidence that an attribute-value machine learning tool like C4.5, designed for treating both numeric and non-numeric attributes, can perform better than statistical classifiers even on purely numeric datasets such as those we have addressed in this paper.

The approach presented can also be applied to other object recognition domains. In the case that the object to be recognised has a linear shape, then the same visual primitives and learning algorithms discussed in the paper can be applied. In the case that the shape differs, then different visual primitives must be used, and the learning algorithms re-tested in order to re-identify the one with the best accuracy.

8 References

1. Cho, K., Dunn, S. M. (1994): Learning shape classes, IEEE Trans. on Pattern Anal. Machine Intell., Vol. 16, No. 9, pp. 882-887.
2. Cox, D. R. (1966): Some procedures associated with the logistic qualitative response curve, in Research Papers on Statistics: Festschrift for J. Neyman, Wiley, pp. 55-77.
3. Cucchiara, R., Piccardi, M. (1999): Eliciting visual primitives for detection of elongated shapes, Image and Vision Computing, Vol. 17, No. 5-6, pp. 347-355.
4. Day, N., Kerridge, D. (1967): A general maximum likelyhood discriminant, Biometrics, Vol. 23, pp. 313-324.

5. Dietterich, T. (1998): Approximate statistical tests for comparing supervised classification learning algorithms, Neural Computation, Vol. 10, No. 7, pp. 1895-1924.
6. Drapter, B. A., Brodley, C. E., Utgoff, P. E. (1994): Goal directed classification using Linear Machine Decision tree, IEEE Trans. on Pattern Anal. Machine Intell., Vol. 16, No. 9, pp. 888-893.
7. Fahlmann, S. E., Lebiere, C. (1988): The Cascade-Correlation Learning Architecture, Advances in Neural Information Processing Systems 2, Morgan Kaufmann, 1988.
8. Fisher, R. (1936): The use of multiple measurements in taxonomic problems, Annals of Eugenics, Vol. 7, pp. 179-188.
9. Illingworth, J., Kittler, J. (1988): A survey of the Hough transform, Comp. Vision Graphics, Image Process., Vol. 44, No. 1, pp. 87-116.
10. Lovejoy, D. J. (1993): Magnetic Particle Inspection, Kluwer Academic Publishers.
11. Michie, D., Spiegelhalter, D. J., Taylor, C.C., eds. (1994): Machine Learning, Neural and Statistical Classification, Ellis Horwood.
12. Murase, H., Nayar, S. K. (1996): Learning by a generation approach to appearance based object recognition, in Proc. of 13th Int. Conf. on Pattern Recognition, Vol. 1, pp. 24-30.
13. Pellegretti, P., Roli, F., Serpico, S., Vernazza, G. (1994): Supervised learning of descriptions for image recognition purposes, IEEE Trans. on Pattern Anal. Machine Intell., Vol. 16, No. 1, pp. 92-98.
14. Quinlan, J. R. (1993): C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, California.