Machine Vision and Applications (2010) 21:627–641
DOI 10.1007/s00138-009-0205-z

SPECIAL ISSUE PAPER

Impact of object extraction methods on classification performance in surface inspection systems

Stefan Raiser · Edwin Lughofer · Christian Eitzinger · James Edward Smith

Received: 26 August 2008 / Accepted: 22 June 2009 / Published online: 27 August 2009

Abstract In surface inspection applications, the main goal is to detect all areas which might contain defects or unacceptable imperfections, and to classify either every single 'suspicious' region or the investigated part as a whole. After an image is acquired by the machine vision hardware, all pixels that deviate from a pre-defined 'ideal' master image are set to a non-zero value, depending on the magnitude of deviation. This procedure leads to so-called "contrast images", in which accumulations of bright pixels may appear, representing potentially defective areas. In this paper, various methods are presented for grouping these bright pixels together into meaningful objects, ranging from classical image processing techniques to machine-learning-based clustering approaches. One important issue here is to find reasonable groupings even for non-connected and widespread objects. In general, these objects correspond either to real faults or to pseudo-errors that do not affect the surface quality at all. The impact of different extraction methods on the accuracy of image classifiers will be studied. The classifiers are trained with feature vectors calculated for the extracted objects found in images labeled by the user and showing surfaces of production items. In our investigation artificially created contrast images will be considered as well as real ones recorded on-line at a CD imprint production and at an egg inspection system.

Keywords Surface inspection · Contrast images · Object extraction · Clustering · Image classifiers

1 Motivation

Machine vision systems currently form an integral part of many machines and production lines. They perform different kinds of monitoring or inspection tasks, without the need to direct contact the parts under investigation. The availability of powerful, yet cheap, cameras and computer systems has made 100% inline quality inspection economically feasible instead sampling by examining randomly selected parts.

A typical application of machine vision in industry is surface inspection, where the surface of a discrete part or endless material is checked for imperfections such as scratches or ridges. In this respect, any deviations from the desired characteristics have to be identified. Surface inspection is of great importance in automobile, electronics, semiconductor, metal-working and printing industry in order to assure a high product quality and so therefore decrease customer complaints with their associated costs for re-production.

Methodologies and methods for surface inspection problems were developed during the recent years—for an overview see for instance [26]. In [12,16,21] particular solutions for the inspection of textile products are shown that are based on an analysis of textures by classical image processing methods. In [13] an approach for surface inspection for wooden material (classification into red oak boards, classes of surface defects, and one class of clear wood) is presented, where image pre-processing techniques are applied for separating

Paper for Special Issue on 'Integrated Imaging and Vision Techniques for Industrial Inspection'.

S. Raiser · E. Lughofer (✉)
Johannes Kepler University Linz, Linz, Austria
e-mail: edwin.lughofer@jku.at

C. Eitzinger
PROFACTOR GmbH, Steyr-Gleink, Austria

J. E. Smith
University of the West of England, Bristol, UK

dark and bright regions and the surface defects are characterized by the mean of the gray levels and causal auto-regressive (CAR) model parameters. In [14] detection and classification of structural defects on silicon carbide (SiC) wafers is presented, focussing detecting a specific fault class (so-called micro-pipes). Most of these methods have one thing in common: they were developed for very specific application scenarios—mostly for one specific industrial inspection system (or even specific fault classes)—and are not directly applicable for a wider range of problems. This also means that the whole development effort for one inspection system has to be repeated again and again and can be a quite complex task. Some other approaches such as [10,15] exploit machine learning methods (such as self-organizing maps or coupled hidden Markov models) for surface inspection and are shown to be applicable for application scenarios such as inspection of wooden surfaces or steel surfaces containing three-dimensional flaws [18].

The framework shown in Fig. 1 is applicable for a wider range of surface inspection problems as it includes a combination of multiple generic components such as machine learning methods. The only assumption therein is that a so-called contrast image is available. In general, this image is calculated from the original image of the inspected part by applying specific, low-level image processing routines. All pixels which differ from a perfect master part are set to a non-zero value, depending on the magnitude of deviation. It might be assumed that whenever a certain contrast image contains any (significant) deviation pixel(s), it always shows a defective production item, and should therefore be classified as 'bad'. However, in most of the applications the correct classification depends on the distribution of the deviation pixels and the characteristics of their local accumulation(s). This is because so-called 'pseudo-errors' may occur which do not correspond to real faults (see also Sect. 2). As a first step all 'suspicious' objects that represent potential defects, are extracted from the contrast image. Then for each of these extracted objects, features such as area, homogenity and roundness are calculated ("object features"). These single features are combined in order to characterize the whole bright pixel area in the image ("aggregated features"), for example their number, average brightness and density. During (off line) training these features are calculated for a set of stored images that are already classified by a human expert, and a classifier is created by applying machine learning methods to this input. In the extraction (on-line) phase, the trained classifier automatically assigns each new image to the best-matching class based on the calculated features, and then generates a 'good'/'bad' decision for the currently inspected part.

The main contribution of this paper is to describe various methods for extracting 'suspicious' objects in the contrast images (Sects. 3, 4) and to analyze the impact of these methods on the predictive performance of the image classifiers (Sect. 5). The latter gives someone a measure of how well the object extraction methods perform in identifying the fault-candidate objects. It will be demonstrated that this cross-link between object extraction methods and performance of the image classifiers (as applicable in our framework and shown in Fig. 1) is an essential aspect, when intending to achieve surface inspection systems with a high accuracy. This means choosing an appropriate object extraction method for a concrete surface inspection problem often guides the image classifiers to a higher accuracy than when choosing a default method. The cross-link will be analyzed based on three data sets, one artificial data set including specific rules mimicking typical errors at surface inspection systems, and two data sets from real-world inspection problems, eggs and CD imprints. Three different types of classification algorithms are applied in order to eliminate any linkage between the biases of a classifier and of a certain object extraction methods, and therefore to see whether one approach can be favored over others.

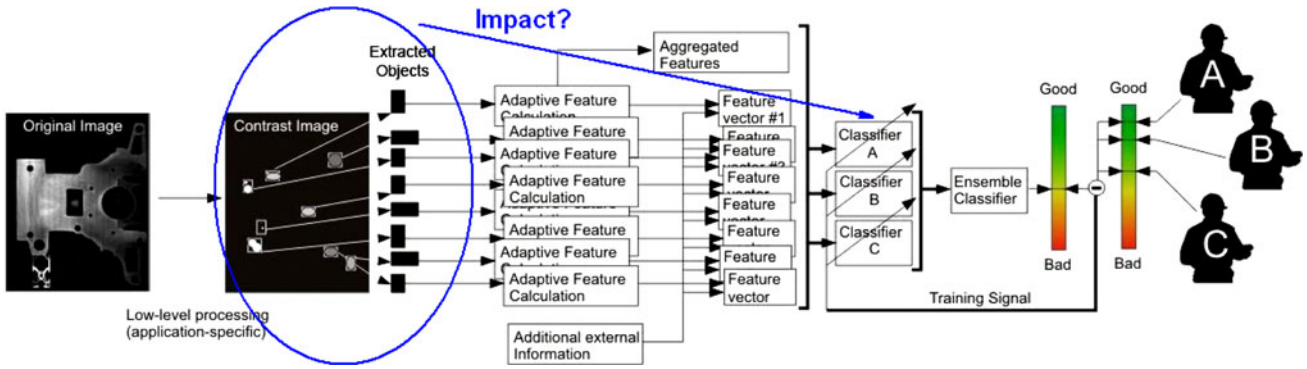

Fig. 1 Surface inspection framework, the object extraction component in contrast images surrounded with an *ellipsoid*; also indicated the (possible) impact on the performance of the machine vision classifiers which are built up based on the feature vectors extracted from the recognized objects—this impact will be studied in the paper

2 Problem statement

All non-zero pixels in the contrast image correspond to regions which differ from the ideal appearance, and the amount of deviation is reflected by the pixel's values. In most cases, the contrast image can be visualized as an 8-bit or 16-bit gray-scale image. Since all deviating areas are detected, the bright pixels can represent either real defects or pseudo-errors, i.e. artifacts caused by varying illumination conditions or improper image alignment.

Figure 2a,b shows contrast images from two different surface inspection applications: an egg inspection and a CD imprint inspection system. In these examples, most pixels are black, whereas the deviating regions consist of rather bright gray values. In both cases arc-type structures occur that represent pseudo-errors, while the other non-zero pixels result from real faults.

In order to process the information contained in the contrast image further, the bright pixels have to be grouped together into meaningful objects, ideally separating pseudo-errors from real ones. These distinct regions are then returned individually to the next processing steps, i.e. the feature extraction.

Typically, the objects are characterized by forming a connected set of pixels. Two pixels are considered to be connected if they are next to each other on the rectangular pixel grid which constitutes a digital image. If the diagonal adjacent pixels are excluded it is called 4-connectivity, if all 8 neighbors are considered it is called 8-connectivity. Every set of pixels that are connected according to either of these two definitions is called a connected component. But in a variety of surface inspection environments, non-connected objects can occur.

Figure 3a shows the contrast image of an egg covered with disconnected spots of yolk (a fault case). An example of highly scattered objects occurring during CD imprint inspection is shown in Fig. 3b. In general, object extraction methods have to be able to find correctly all objects, and these consist of bright pixels that visually belong together, but need not necessarily be connected. For a human it is quite easy to find the natural groupings in the contrast image, because he or she is able to interpret the image in a 'wider context' and can perceive patterns, even if they are very fragmented. Automatic procedures are challenged by the fact that they have to deal with connected and non-connected objects of arbitrary size, shape, and number, without any further information about the image content. Thus, in order to match the human interpretation of an image (as reflected in their labeling), the object extraction method should approximate the human grouping as closely as possible. Otherwise, the features, calculated from the automatically found objects will be misleading for a classifier trained to make human-like decisions. Feature values may get disturbed and probably reflect other classes than they really belong to.

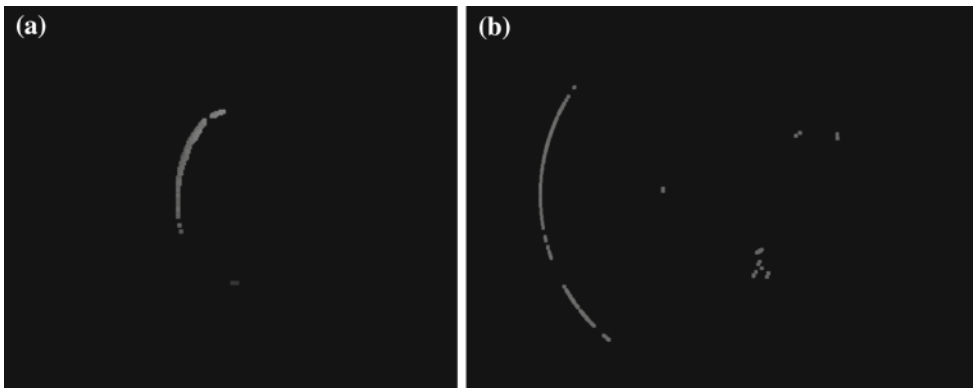

Fig. 2 Contrast images containing real defects and pseudo-errors from **a** an egg inspection, **b** a CD imprint inspection system

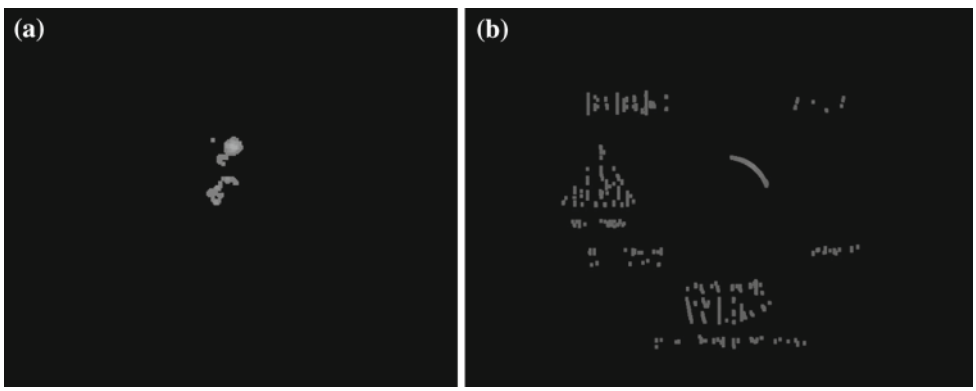

Fig. 3 Contrast images containing disconnected objects from **a** an egg inspection, **b** a CD imprint inspection system

In the following sections we will outline a very common object extraction approach, namely the connected component algorithm, together with some enhancements, and then give an overview on machine learning approaches based on clustering techniques.

3 Common techniques for object extraction

In our framework from Sect. 1, the contrast image is calculated by taking the absolute difference between the current image and a fault-free master image and already contains all interesting areas as bright pixels. Therefore, the following object extraction step has to find meaningful groupings for the bright pixels that reflect the underlying object structure as well as possible. One does not consider any textural structure of the defects and just use a binary version of the image for object extraction. In the simplest case, global thresholding is applied to the contrast image that sets all pixels with a gray value above a certain level to one and the remainder to zero. Therefore, the thresholding produces a binary image with a black (0) background and a white (1) foreground that contains the objects of interest. A detailed description of various thresholding methods can also be found in [6].

The most common approach for obtaining the individual objects from a binary image is the connected component algorithm (CC) [23]. This assumes that the objects to be found consist of a connected set of pixels. Therefore, all connected components detected by the algorithm correspond to the distinct objects. Various implementations of the connected component algorithm are discussed in the literature. The outcome can either be a list of sets of connected pixels, or a colored (labeled) image with different colors (labels) assigned to the distinct objects.

Figure 4 shows the results of applying the connected component algorithm to the binarized contrast images from the previous section. In all cases, the number of objects found exceeds the number that a human would perceive/identify, because the assumption of connectedness is not fulfilled. Especially, for images with the highly scattered 'pixel-clouds' (see Fig. 4d), the algorithm returns far too many objects that leads to erroneous feature values and consequently may deteriorate the performance of the classifiers.

Therefore, in the case of non-connected objects, an algorithm is needed that somehow groups pixels together on a scale, bigger than the pixel neighborhood, in order to reduce the number of objects found. One traditional way is to improve the connected component algorithm by performing a pre-processing step before applying it.

A simple way to enhance the binarized contrast image with respect to the outcome of the connected component algorithm, is to first blur the image—for example with a gaussian low-pass filter in order to close gaps between bright pixels. Then the blurred result is binarized and the connected component algorithm is applied to it. As the connectivity between the pixels has increased due to

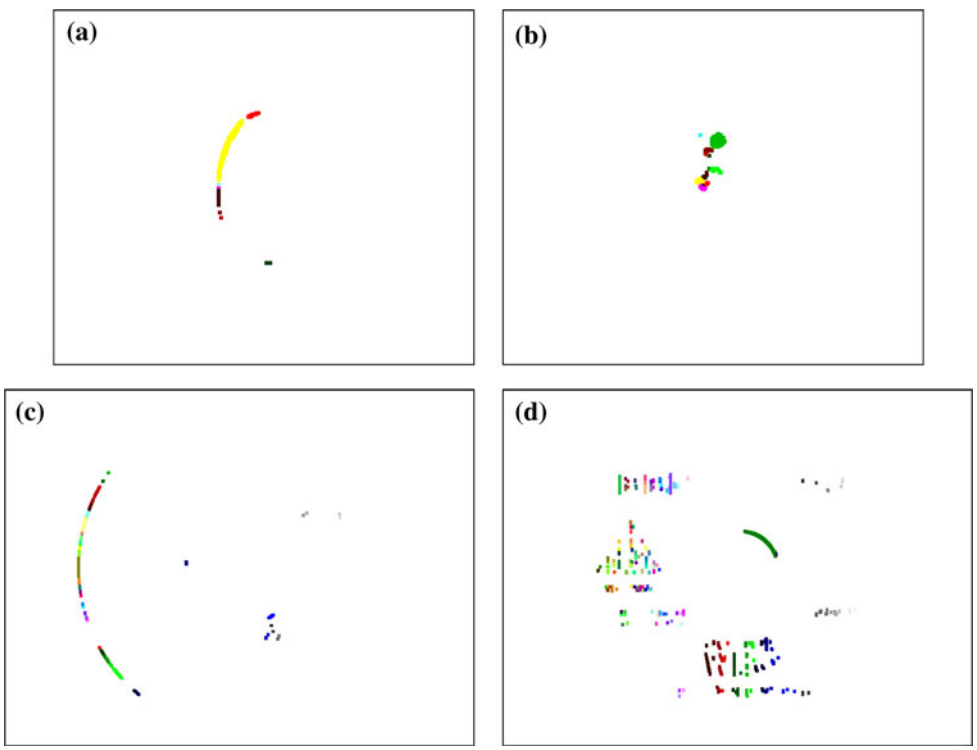

Fig. 4 Results of connected component labeling with **a** 8, **b** 11, **c** 51, **d** 203 objects found

the blurring, the number of objects found is generally much smaller than without filtering. Finally, the pixels generated while blurring the contrast image, are removed again from the labeled image, while keeping the labeling constant. Although in many cases the result is quite good, it strongly depends on the choice of the blurring filter and its parameter settings.

Figure 5a shows a CD imprint contrast image that was binarized, then blurred by a gaussian low-pass filter (with $\sigma = 4.4$ and a kernel size of 30×30 pixels) and finally binarized again by global thresholding. After applying the processing steps outlined above, the final result (Fig. 5b) exhibits a much smaller number of objects.

Another way to increase the connectivity between bright pixels which are close together, but not connected, is by applying morphological operations as a pre-processing step on the binarized contrast image. In order to remove holes in the white pixel areas and small gaps between these areas, a morphological closing can be performed. This consists of a dilation followed by an erosion with a structuring element, i.e. a disk with a certain size (see [6]). After that, the connected component algorithm is applied, usually extracting a smaller number of objects. Finally, the additional pixels generated by the closing are removed again, keeping the labeling constant. The result strongly depends on the choice of the structuring element, its shape and size, respectively. Even small changes can produce a different labeling and a varying number of objects found.

Figure 6a illustrates the effect of closing the CD imprint contrast already shown before with a disk-shaped structuring element. Again, the final result (Fig. 6b) is improved by the pre-processing.

As a conclusion, it can be said that the connected component algorithm alone is (obviously) not a good choice for images containing non-connected objects. Performing various kinds of pre-processing, leads to more reasonable results, but the quality depends strongly on the parameter setting of the pre-processing algorithm(s) and requires additional computational effort. In this paper, specific clustering-based approaches are examined which circumvent these shortcoming under specific conditions as demonstrated in the subsequent section.

4 Clustering-based approaches for object extraction

Clustering methods have been used in a variety of disciplines ranging from statistics and numerical analysis to data mining and machine learning. In general, clustering can find 'natural' groupings (clusters) in data. Each of these clusters consist of data points that are similar to each other and dissimilar to those from other groups with respect to a

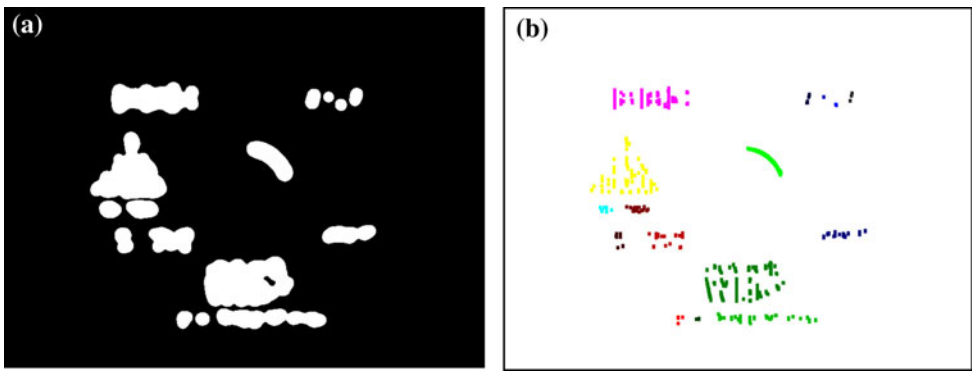

Fig. 5 **a** Blurring with gaussian filter, **b** result of connected component labeling (16 objects found)

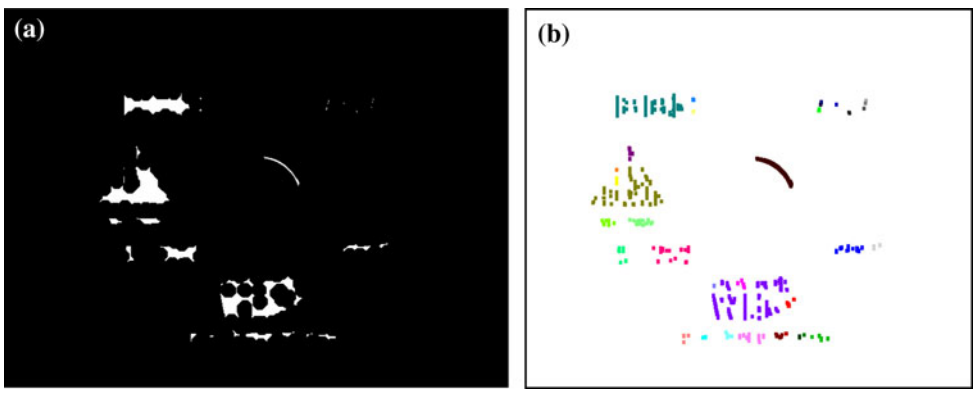

Fig. 6 **a** Morphological closing, **b** result of connected component labeling (35 objects found)

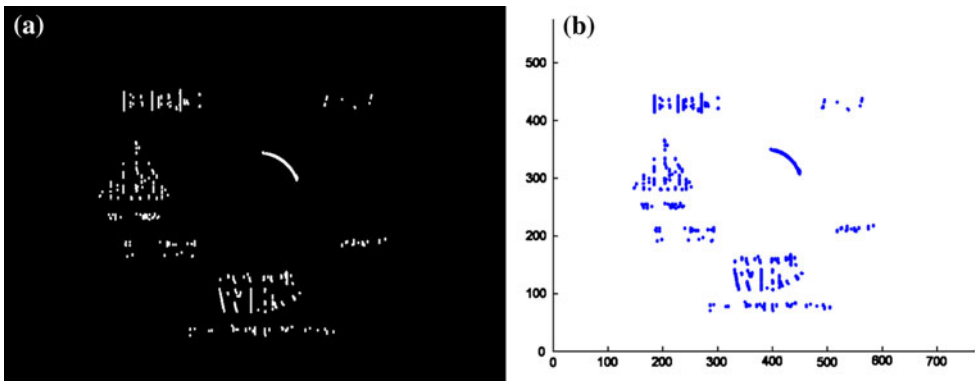

Fig. 7 **a** Binarized CD imprint contrast image, **b** *white* pixels as data points for clustering

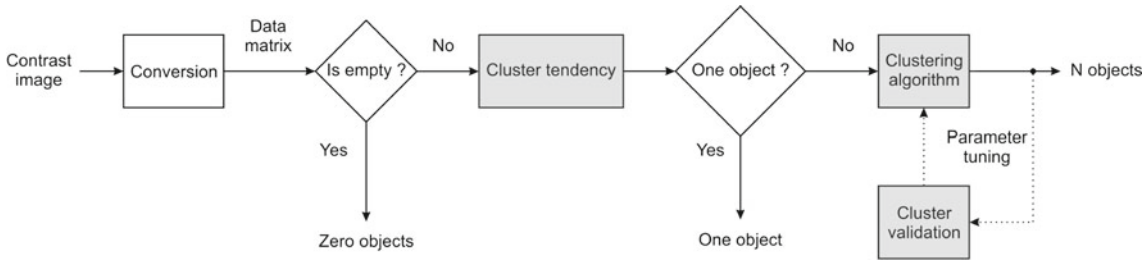

Fig. 8 Integration of clustering methods in the object extraction process

previously chosen similarity/dissimilarity measure. A comprehensive description of clustering can be found in e.g., [11] or [5].

This characteristic makes clustering a candidate for solving object extraction tasks, as usually different types of fault candidates appear in different groups of non-zero pixels in the contrast image. The basic idea is to consider the white pixels in the binarized contrast image as data points that serve as input for the clustering algorithm. Figure 7a shows the binary version of a CD imprint contrast image and Fig. 7b is a visualization of the corresponding data matrix, illustrating the conversion of pixels into two-dimensional data points.

In this case, the original task, namely the extraction of objects, becomes equivalent to the problem of finding clusters in the set of data points. The question of which pixels belong together (i.e. to the same object) is now addressed by the clustering algorithm and how it partitions the data points.

Figure 8 shows how clustering methods can be integrated in the object extraction process.

After the binary contrast image has been 'converted' into a data matrix, a cluster tendency analysis checks for the presence of a clustering structure. If the image contains more than one object, a clustering algorithm is applied to the data points in order to extract the individual objects. Optionally, cluster validation techniques can be used to tune the parameters of the clustering algorithm.

4.1 Cluster tendency

Before applying a clustering algorithm various tests can be performed that indicate whether the available data possess a clustering structure at all. These methods examine the data to see if there is any merit to a cluster analysis or not. In the image processing context, cluster tendency addresses the question of whether there is more than one object in the image. In the case of a single object, clustering is not really useful, since all foreground pixels can be grouped together and the object extraction is finished.

A very intuitive cluster tendency test based on nearest neighbor distances is the calculation of the Hopkins index h (Hopkins test) as described in [9] and [19]. After choosing m random points s_m from the data matrix and also m random points r_m from its convex hull, the distances d_{s_i} (d_{r_i}) from each s_i (r_i) to its nearest neighbor is calculated and inserted in the following formula:

$$h = \frac{\sum_{i=1}^{m} d_{r_i}^2}{\sum_{i=1}^{m} d_{r_i}^2 + \sum_{i=1}^{m} d_{s_i}^2} \in [0, 1] \tag{1}$$

The value of h lies between zero and one, whereas the three typical cases shown in Fig. 9 can be distinguished:

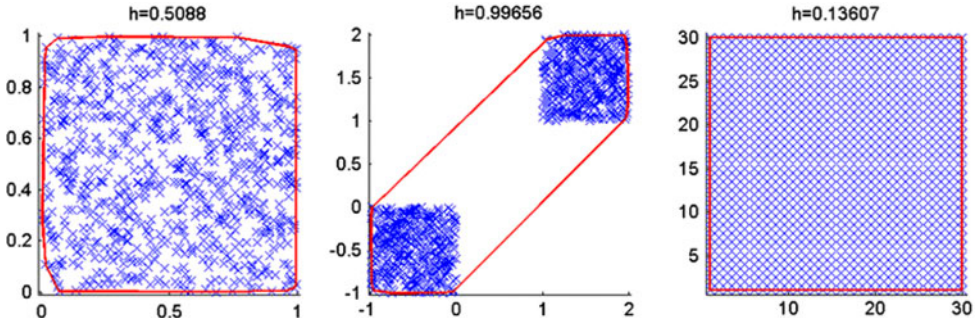

Fig. 9 Three data sets and their Hopkins indices

1. If $h \approx 0.5$ (as seen on the left), then the data points are likely to have a random pattern (only a single cluster).
2. If $h \approx 0$ (as seen on the right), then the data points are located on a regular grid (as many clusters as data points).
3. If $h \approx 1$ (as seen in the middle), then a clustering structure exists (number of clusters somewhere between 2 and the number of data points).

In the context of object extraction, the Hopkins index h can be used as a stand-alone test for cluster tendency. If h is smaller than a certain threshold (threshold values between 0.5 and 0.8 showed good experimental results here), all white pixels belong to the same (single) object and no clustering has to be done.

The performance of the Hopkins index in detecting images with single clusters was tested based on data sets where the real number of objects in the images is known as they were generated artificially (but simulating a real-world inspection process). For each image containing more than 20 white pixels, the index was calculated (note that the value of the Hopkins index has low significance if it is calculated from too few data points). If its value exceeded a threshold of 0.6, the image was considered to contain more than one object. Otherwise, all white pixels in the image were grouped together as a single object. The results were remarkably good as (1) a high percentage (above 85%) of single clusters could be identified, whereas the other 15% were recognized as single clusters by the clustering algorithm, afterwards (hence, the 15% mis-detection were compensated by clustering) and (2) a very low percentage (smaller than 1%) of images were mis-detected as single clusters (as they contained more than one cluster) and hence not sent into the clustering algorithm. This causes a disturbance value of 1% in the features that is fortunately quite a low value.

4.2 Clustering algorithms

After checking the existence of clustering structures, or in other words after verifying that more than one object exists in the binary image, a clustering algorithm can be applied to find the natural grouping of the foreground pixels.

For the object extraction task a clustering algorithm must be able to detect clusters of arbitrary shape, since the foreground pixel areas in the binary image can exhibit any form. After the grouping, each foreground pixel should be assigned to a distinct cluster (object), which means (in terms of clustering), that a crisp partition is needed as output. Also, if the algorithm returns a hierarchy of partitions, a cutoff level has to be determined in order to get a single partition.

The following clustering algorithms are all capable of finding arbitrarily shaped objects in a binary image and their performance has been investigated for the example applications described in this paper:

1. Single linkage hierarchical clustering (HC-SL) [25] is a representative of the agglomerative hierarchical algorithms. It starts with every data point being a cluster and then iteratively merges the closest pair of clusters according to some distance measure, until all data points are in one cluster. In the single linkage version of the algorithm, the distance between two clusters is measured by the distance of the closest pair of data points from each cluster. This kind of distance measurement enables the algorithm to identify clusters of arbitrary shapes and different sizes. Elongated objects can also be detected due to the so-called 'chaining-effect' that groups lines of distinct, yet closely connected data points together. For these reasons the HC-SL algorithm is quite feasible for object extraction. But as this task demands a single partition instead of a hierarchy, the hierarchical algorithm has to be stopped at a specific level of the hierarchy, when the desired partition is obtained and further grouping would only combine clusters that do not belong together. The stopping criterion is implemented as a threshold, called the *cutoff* value that determines the maximum distance allowed for merging two clusters.
2. DBSCAN [4] is a density-based algorithm that regards clusters as dense regions in the data space, separated by regions of lower density. At the beginning, the algorithm initializes all data points as unclassified. Then it starts with an arbitrary data point and finds all points that can be reached from it, while staying inside a region of high data density. If the point turns out to be somewhere inside

a dense area, then a cluster is formed. If the point is at the border of a dense area, the algorithm proceeds with the next unclassified point. If the point is neither inside nor at the border of a dense area, it is classified as noise. This continues until all of the points have been processed. The DBSCAN algorithm is optimized to work efficiently on large spatial databases, which makes it especially useful for object extraction, as the number of bright pixels in the contrast image can be quite large.

3. Reduced Delaunay graph (RDG) [17] represents a graph-based approach and is based on partitioning the Delaunay graph of a given data set. All edges of the graph are weighted according to a normalized distance measure and clustering is performed by removing all edges for which the weight is larger than a fixed threshold. Finally, each connected component of the remaining graph corresponds to a single cluster. According to the authors, the algorithm is able to find clusters of complex shape and it groups data points in a similar way as human observers.
4. Normalized cut (NCUT) [24] is a spectral clustering method that represents a very powerful class of clustering algorithms developed in the early 2000s. It starts with a fully connected graph, where each node represents a data point and the edge between two points is weighted by their similarity, i.e. their distance from each other. The algorithm recursively finds the bi-partitioning of the graph that minimizes a certain criterion, namely the normalized cut value. The recursion process stops, if the quality criterion for the cut becomes higher than a specified threshold or there are only single points left.

4.3 Parameter setting and cluster validation

Clustering, like other object extraction approaches, has at least one input parameter that significantly influences the number of objects found by the algorithm. In general, every image contains a different number of objects, hence it is not sufficient to determine values for these critical parameters by trial-and-error using only a small set of test images. Fortunately, in many cases heuristic methods or estimation formulas are available to set the parameters to reasonable values for each new image to be clustered.

In the following paragraphs, the essential parameters of the clustering algorithms from the previous section are listed and some 'practical' methods for the determination of these parameters are proposed:

1. Single Linkage Hierarchical Clustering (HC-SL). The *cutoff* value, used to select a single partition from the hierarchy, determines the number of objects found. In the context of image processing, it is feasible to make the *cutoff* value depending on the size of the image, because the size gives a strong hint on the scale of the objects, which the image contains. For instance, in many surface inspection applications the object to be investigated often completely fills the whole image. Here, a typical rule of thumb is to set the *cutoff* to

$$cutoff = \frac{\#imageRows + \#imageColumns}{30}. \tag{2}$$

As its value can be determined a priori according to the formula, the algorithm can be stopped, when the distance between the closest clusters exceeds the *cutoff*. In contrast, other methods for finding an appropriate *cutoff* value (i.e. as described in [20]) need the whole hierarchy to be computed. This leads to a longer computation time.

2. DBSCAN. In this case, the setting of two parameters, namely $MinPts$ and ϵ that relate to the expected density of the 'thinnest' cluster, is required. $MinPts$ can be fixed at a value of 4 as proposed by [4] for two-dimensional data. The value of ϵ can either be determined using the *sorted k-distance graph* (see also [4]) or the following estimation formula (taken from [3]):

$$\varepsilon = \sqrt{\frac{\left(\max\limits_{1\le i\le m}(x_m) - \min\limits_{1\le i\le m}(x_m)\right)\cdot\left(\max\limits_{1\le i\le m}(y_m) - \min\limits_{1\le i\le m}(y_m)\right)\cdot MinPts\cdot\Gamma(2)}{m\cdot\pi}} \tag{3}$$

with $MinPts = 4$, Γ the gamma function $\Gamma(z) = \int_0^\infty t^{-1+z}\mathrm{e}^{-t}\mathrm{d}t$ and the m bright pixel coordinates stored in a $m \times 2$ matrix

$$D = \begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_m & y_m \end{bmatrix}.$$

3. Reduced Delaunay Graph (RDG). The threshold for the edge weights, applied when removing edges from the Reduced Delaunay Graph, influences the amount of connected components and hence the number of objects. If all the edge weights in the graph are sorted in a descending order, and a plot made of weight values (y-axis) against the sorted rank (x-axis), an L-shaped curve evolves. A good threshold value can be obtained by taking the edge weight at the 'knee' of this curve. It can be determined automatically by normalizing the two axis and finding the point with the closest (Euclidian) distance to the origin. Figure 10 illustrates the procedure on an artificial sample image.

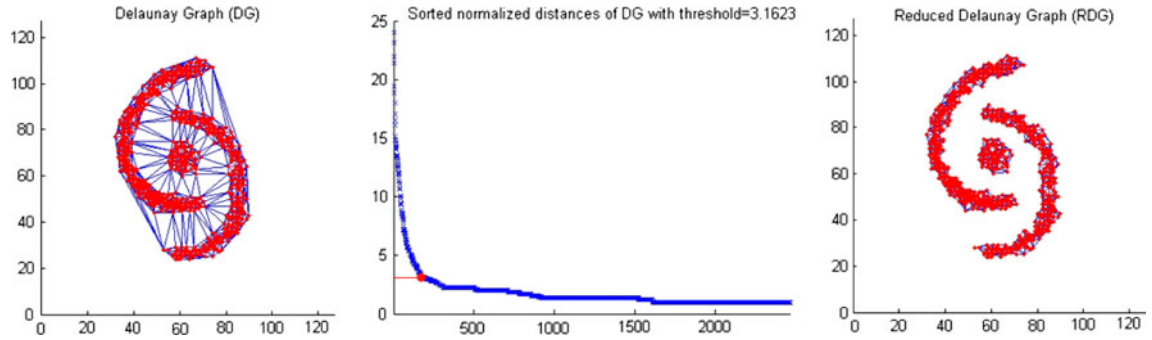

Fig. 10 *Left* Delaunay graph, *middle* threshold determination, *right* reduced Delaunay graph

4. Normalized Cut (NCUT). This algorithm starts by constructing a weighted graph with the weights generally calculated using a Gaussian similarity measure,. Thus the most important parameter is the value of σ in the Gaussian function, since it strongly influences the neighborhood relationships and hence the number of clusters found. In case of object extraction, meaningful values for σ are depending on the image/object size. Hence, it is feasible to fix its value at some percentage (typically between 5 and 10%) of the image dimensions. The second parameter, the threshold for $Ncut$ and used as stopping criterion for the recursion, also has to be set by the user. In practice, the exact value is not very critical and typically somewhat above zero.

All clustering results presented in this paper have been generated using these automatic parameter determination methods.

Another approach is to use so-called cluster validation techniques (see [7]) to determine an appropriate parameter setting, or to perform an automatic fine-tuning of estimated parameter values during on-line operation. Here, the most feasible methods for automatic parameter tuning are the ones based on relative criteria. These offer the possibility of choosing the best out of a set of clustering schemes according to some criterion, reflected by a so-called cluster validation (CV) index. After clustering with different parameter settings, the resulting partitions are rated on-line on the basis of the CV index and the 'best' partition is selected and returned as final result. As this iterative approach demands high computational effort, it is not further discussed here.

4.4 Visualization results on real-world image sets

For the real-image dataset, no expert knowledge has been available about the objects actually contained in the images, hence the assessment of the object extraction quality had to be carried out simply by visual judgement. Figures 11 and 12 illustrate the actual grouping of the white pixels according to the different approaches for images from a CD imprint inspection. In these figures, the colors used to mark the objects found, have no meaning, and can differ between the clustering algorithms even when the objects have been extracted identically. As the objects in this application are highly discontinuous and scattered, all clustering methods clearly outperform the connected component algorithm and return quite meaningful groupings. Here, the parameters of the clustering algorithms were changed and the results inspected manually. The most plausible results are shown in Figs. 11 and 12.

5 Impact of object extraction on classification accuracy

In this section, we study the impact of the various object extraction techniques outlined above on the accuracy of image classifiers trained to distinguish between good and bad production items. As already shown in Sect. 1, the image classifiers are built into a whole image classification framework (Fig. 1), consisting of several components, including an object extraction and feature calculation part. Features are calculated from extracted objects, whereas one labels the pixels according to the objects they belong to. Two types of features can be distinguished: object features and aggregated features. The former characterize single objects by their size, shape, density, statistical values from gray-level histograms, etc. It is quite intuitive that an object extraction approach which does not correctly group the objects of interest will affect the value of the features extracted from these objects. For instance this will happen if two or more objects are connected to one which should not be connected, or too many objects are extracted—as is often the case when using connected components (see figures in previous section). One calls this effect a kind of disturbance of feature vectors in the feature space. This disturbance is assumed to guide the classifier to a wrong solution since it does not represent the correct implicit induction of the classes previously assigned

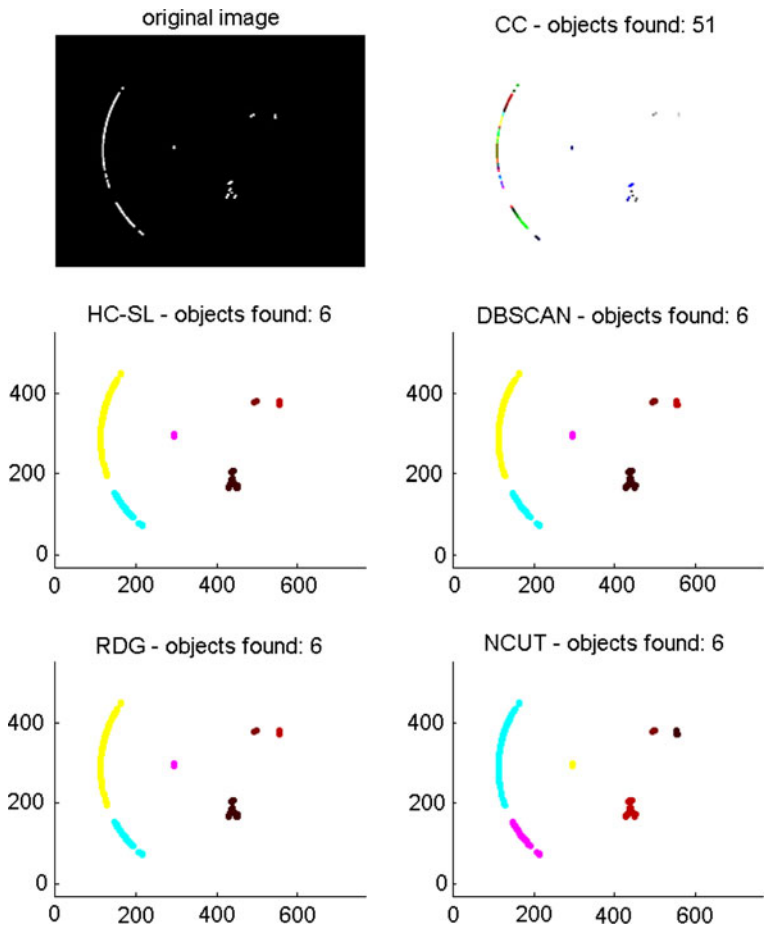

Fig. 11 First CD imprint inspection example: Here, an image with an arc-type structure and four small clusters is shown. All clustering algorithms agree in finding six distinct objects. Although the arc is split in two pieces, the results are very good compared to the outcome of the connected component algorithm, which finds too many objects

by the experts to the real objects (experts usually know what the objects look like). The disturbance can reach a such level that the object feature vector belonging to an (badly recognized) object labeled as class X is similar to another feature vector belonging to a (well recognized) object labeled as a different class Y. This means that the two vectors are very close to each other in the feature space, whereas they represent different classes (hence, the distinction between the classes gets smaller). If this is the case for several feature vectors (objects), it will finally get more difficult for the classifier training algorithm to obtain a reliable decision boundary between the classes X and Y.

The aggregated features describe the characteristics of images as a whole and hence are used for training an image classifier for classifying images into 'good' and 'bad' ones. They implicitly contain information from the single objects and hence may also be 'disturbed' by bad object extraction approaches. The complete list of aggregated features is listed in Table 1.

The following formulas define how these features where calculated; the symbol $featX$ belongs to the Xth feature in Table 1, C denotes the number of objects (clusters), P_m the number of pixels belonging to the mth object, $p_{i,m}$ the ith pixel in the mth object, $g(p_{i,m})$ its gray level (intensity) lying in $[0, 255]$ $p_{i,m}(x)$, respectively $p_{i,m}(y)$ the x- respectively y-coordinate of the ith pixel in the mth object:

$$feat1 = C$$

$$feat2 = \frac{2}{C(C-1)} \sum_{i=1}^{C} \sum_{j=i+1}^{C} \min_{m,n} |p_{i,m} - p_{j,n}|,$$
$$m = 1, \ldots, P_i, \quad n = 1, \ldots, P_j$$

$$feat3 = \min_{i,j=1,\ldots,C} |p_{i,m} - p_{j,n}|$$
$$m = 1, \ldots, P_i, \quad n = 1, \ldots, P_j$$

$$feat4 = P_m, \quad m = \text{argmax}(P_1, \ldots, P_C)$$

$$feat5 = \bar{O}_m(x) = \frac{1}{P_m} \sum_{i=1}^{P_m} p_{i,m}(x),$$
$$m = \text{argmax}(P_1, \ldots, P_C)$$

$$feat6 = \bar{O}_m(y) = \frac{1}{P_m} \sum_{i=1}^{P_m} p_{i,m}(y),$$
$$m = \text{argmax}(P_1, \ldots, P_C)$$

$$feat7 = \max_{j=1,\ldots,C} \left(\left(\frac{1}{P_j} \sum_{i=1}^{P_j} g(p_{i,j}) \right) * P_j \right)$$

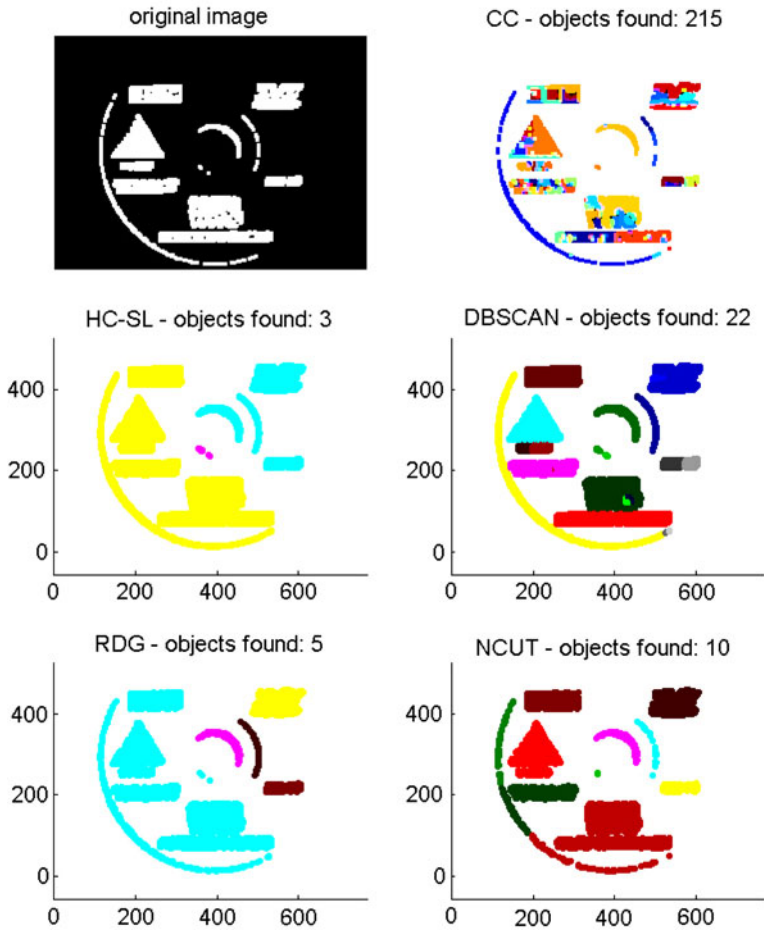

Fig. 12 Second CD imprint inspection example: In this case, clustering algorithms perform dissimilarly: again, the connected component algorithm returns worse results, but also HC-SL is quite weak as connecting too much objects together and delivering to big objects. DBSCAN clearly over-estimates the number of objects (some compact objects are divided up into more), whereas RDG also under-estimates it (as HC-SL). Intuitively normalized cut performs best, except for the longer arc-type object at the left hand side which is divided up into three objects

$$feat8 = \frac{1}{P_m}\sum_{i=1}^{P_m} p_{i,m}(x),$$
$$m = \text{argmax}_{j=1,\ldots,C}\left(\left(\frac{1}{P_j}\sum_{i=1}^{P_j} g(p_{i,j})\right) * P_j\right)$$
$$feat9 = \frac{1}{P_m}\sum_{i=1}^{P_m} p_{i,m}(y),$$
$$m = \text{argmax}_{j=1,\ldots,C}\left(\left(\frac{1}{P_j}\sum_{i=1}^{P_j} g(p_{i,j})\right) * P_j\right)$$
$$feat10 = \max_{j=1,\ldots,C}\left(\max_{i=1,\ldots,P_j} g(p_{i,j})\right)$$
$$feat11 = \frac{1}{\sum_{j=1}^{C} P_j}\sum_{j=1}^{C}\sum_{i=1}^{P_j} g(p_{i,j})$$
$$feat12 = \sum_{j=1}^{C} P_j$$
$$feat13 = \sum_{j=1}^{C}\sum_{i=1}^{P_j} g(p_{i,j}) \quad (4)$$
$$feat14 = \max_{j=1,\ldots,C} |\{O_i | i \neq j \wedge \exists p \in O_i : (\bar{O}_j(x) - p(x))^2 + (\bar{O}_j(y) - p(y))^2 < r^2\}| \quad r \text{ as in (2)}$$
$$feat15 = \frac{1}{C}\sum_{i=1}^{C} |\{O_i | i \neq j \wedge \exists p \in O_i : (\bar{O}_j(x) - p(x))^2 + (\bar{O}_j(y) - p(y))^2 < r^2\}| \quad r \text{ as in (2)}$$
$$feat16 = \bar{P} = \frac{1}{C}\sum_{i=1}^{C} P_i$$
$$feat17 = \frac{1}{C}\sum_{i=1}^{C}(P_i - \bar{P})^2$$

Furthermore, if single objects are not labeled, an aggregation or a clustered information of object features is carried

Table 1 List of aggregated features used

No.	Description
1	Number of objects
2	Average minimal distance between two objects
3	Minimal distance between any two objects
4	Size of largest objects
5	Center position of largest objects, x-coord
6	Center position of largest objects, y-coord
7	Maximal intensity of objects
8	Center position of object with max. intensity, x-coord
9	Center position of object with max. intensity, y-coord
10	Max. gray value in the image
11	Average gray value of the image
12	Total area of all objects
13	Sum of gray values in all objects
14	Maximal local density of objects
15	Average local density of objects
16	Average area of the objects
17	Variance of the area of the objects

These features have been chosen because of their relevance for a very wide range of surface inspection applications

out [2] in order to include the nature of the objects in the whole image classifier. One applied an empirical evaluation of image classifier accuracies on three different data sets to show the impact of the various object extraction approaches on the classification performance. These were an artificial data set (where good/bad decision rules at a real industrial surface inspection are imitated), one set recorded on-line at a CD imprint production process and another from an egg inspection system. We used the same parameter settings for all clustering algorithms as a priori manually tuned based on representative exemplars from the image data sets.

5.1 Characteristics of the data sets

5.1.1 Artificial surface inspection data set

Five sets of artificial test data were used, each with 20,000 images that were labeled automatically either as good (accept) or as bad (reject) with about 10,000 images in each class. No labels were given for the single objects. In order to generate the labels, a set of rules was used for each set of test images. The rules were based on descriptions that are regularly found in quality control instructions, such as "part is bad, if there is a fault with size >1.5 mm". The rules also included more complicated combinations, such as "part is bad, if there is a cluster of four faults, each with a size > 0.75 mm". Three to five such rules were logically combined for each set of images. The images and the rules were chosen to have some resemblance to inspection of machined parts. The first two data sets (ArtifData01 and ArtifData02) contained quite simple rules on the whole image, where the other three data sets (ArtifData03-05) contained more complex rules which also depend on the distribution of the objects in an image. Two examples of artificial images (size 128×128 pixels) are shown in Fig. 13, the left image shows a bad one, the right image a good one, even including some pixels marking significant deviations.

5.1.2 Egg inspection

In this real-world application, hen's eggs are inspected in order to identify dirt and yolk that might cover parts of the eggshell. The images have a size of 313×262 pixels and are gray scale with 256 gray levels. There are three sets with 4,342 images available: one with dirt only, one with yolk only and a third one with both kinds of defilement. Thus, it was possible to label the object feature vectors according to dirt and yolk. The images were pre-labeled by the experts with 'good' and 'bad'. In fact most of the images were labeled as

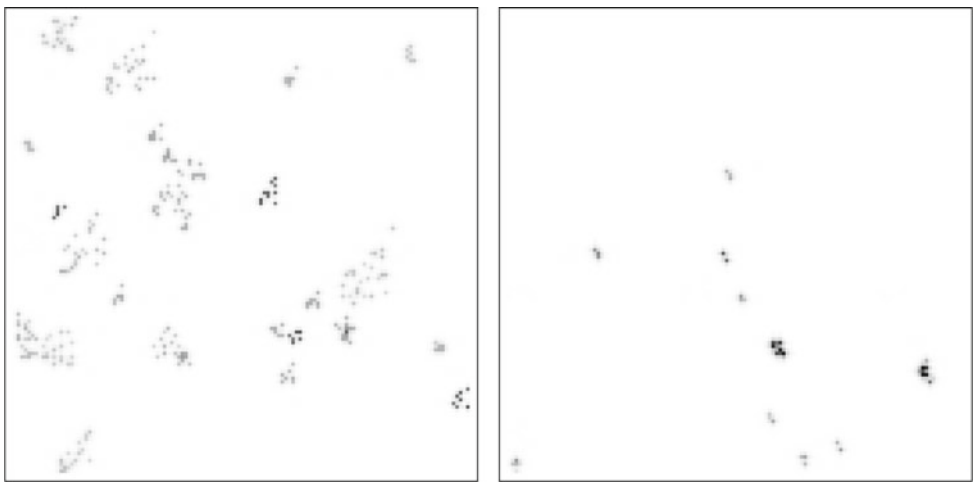

Fig. 13 *Left* artificial images labeled as bad, *right* artificial image labeled as good

good, therefore for classifier training the input was quite an imbalanced data set. Due to the nature of dirt and yolk, their shape can be arbitrary. The correct number of objects in the images is unknown, therefore an object extraction approach supporting the extraction of objects with arbitrary shape is required. Example deviation images from the egg inspection system are shown in Figs. 2a, 3a, 4a, b.

5.1.3 CD imprint inspection

The images in this application example come from a CD imprint inspection process. In order to guarantee the quality of the imprints, a color matrix camera is installed in the production line that takes an image of each CD. These images are compared with the image from the corresponding fault-free master CD ($\pm$ a threshold) and each deviation is marked as a probable defect. The outcome is a gray-scale image of size 768×576 pixels, where the gray levels correspond to the amount of deviation from the master. Obviously, the defective areas have arbitrary size and shape and often exhibit some kind of scattered structures, for example as they occur in the case of ink splashes. The major effort was to distinguish between real and pseudo defects. Real defects include a color drift during offset print, a pinhole caused by a dirty sieve ($\rightarrow$ color cannot go through), occurrence of colors on dirt, palettes running out of ink. The so-called pseudo-errors include shifts of CDs in the tray (disk not centered correctly), causing longer arc-type objects (e.g. see left side of Fig. 11) or masking problems at the edges of the image or illumination problems ($\rightarrow$ causing reflections). The CDs showing the latter artifacts should be recognized as 'good'. After object extraction, features are calculated for the objects found. They serve as input for the classifier that decides whether the imprint is acceptable or has to be rejected.

5.2 Experimental setup

For the artificial data sets the good/bad labels of the images were automatically given by the pre-defined rules. The egg data sets containing 4,342 images were labeled by one expert at the system. The CD imprint data set containing 1,534 non-black images (purely black images can be trivially identified as fault-free) which were labeled by 4 different operators with a slightly different view on the objects and whole images. As it was done by real experts, the labeling represents the characteristics of good and bad images quite well. One hypothesizes that wrongly identified objects will have an influence on the classification accuracy of the classifiers built using the expert labels, as feature vectors extracted from the objects are disturbed. There are 57 object features, and 17 aggregated features which characterizing images as a whole as shown in Table 1. For the artificial and CD imprint data sets unsupervised information from the objects are used, by aggregating the maximum value observed for each object feature in each image and appending them to the aggregated features for that image. For the egg data set object labels were used to directly to train an object classifier distinguishing between dirt and yolk. According to an interpretation of Wolpert's [33] "no free lunch theorem" which says that for different data sets different types of machine learning methods (and also classifiers) perform best, one does not stick to one fixed classification method. Instead one exploits three different well-known classification methods, namely decision trees generated by the Classification And Regression Trees (CART) approach [1] (implementation in MATLAB^{TM} statistics toolbox), support vector machines (SVMs) [22,29] (lib-SVM implementation) and possibilistic neural networks (NN) [32] (implementation in the MATLAB^{TM} neural network toolbox). This also allows to see whether, and how, the different object extraction approaches affect a variety of different classifiers in terms of their predictive accuracy. If assuming that connected components generates a quite high disturbance of the feature vectors, since it usually produces too many objects as shown above, this also yields a kind of 'robustness' analysis with respect to noise for the various classification methods. For all classifiers 10-fold cross-validation step [27] is applied which is coupled with a best parameter grid search scenario to elicit a good estimation of the expected prediction error on new unseen samples, see [8]. The search over a parameter grid finds the optimal parameter setting of each single classifier (optimal in terms of minimizing the CV error), which is the pruning level for CART, C and γ for SVMs and the spread of the neurons for possibilistic NN.

5.3 Results

The results are presented in separate tables for the three different classifiers (CART, SVMs and possibilistic NN), where the rows represent the various data sets and the columns represent the various object extraction approaches. The single values in the tables represent the 10-fold CV accuracy. For all the classifiers, the standard deviation among the ten different folds is quite low for all data sets, i.e. smaller than 0.5% difference in accuracy. This means that an increase of more than 0.5% accuracy between different methods is already significant. Table 2 shows us the results obtained when using CART classifier. It is evident that for the artificial data set all methods perform equally, except the normalized cut approach that brings significant improvement (up to 5%) compared to the other methods. This is not a big surprise when taking into account that the mean absolute error on the number of objects achieved by this method is about half as high as for the other methods (note that the number of objects is known in this data set as artificially generated). For the CD imprint data the situation is a bit different, as DBSCAN can

Table 2 Classification accuracies achieved by CART classifier when using different object extraction methods, the best performing ones given in bold font

CART	CC	HC	DBSCAN	RDG	NCUT
Art. #1	87.86	87.99	88.18	87.05	**92.76**
Art. #2	93.18	93.28	93.45	93.52	**94.25**
Art. #3	94.28	94.40	93.70	93.55	**95.42**
Art. #4	89.32	89.32	86.90	86.43	**91.63**
Art. #5	91.59	91.88	89.87	89.90	**92.20**
Print #1	88.25	93.20	**94.56**	92.20	NA
Print #2	90.64	95.56	**95.99**	93.33	NA
Print #3	91.68	94.51	**95.71**	93.55	NA
Print #4	90.84	94.90	**95.15**	94.11	NA
Egg	92.08	98.89	**99.54**	99.13	97.95

Table 3 Classification accuracies achieved by SVM classifier when using different object extraction methods, the best performing ones given in bold font

SVMs	CC	HC	DBSCAN	RDG	NCUT
Art. #1	87.68	87.89	88.50	87.80	**91.02**
Art. #2	93.45	93.55	93.98	93.52	**94.64**
Art. #3	93.70	93.66	93.52	93.32	**94.38**
Art. #4	90.20	90.43	88.43	88.73	**93.32**
Art. #5	90.77	91.11	90.33	88.72	**93.32**
Print #1	88.74	93.59	**94.92**	93.22	NA
Print #2	93.45	96.73	**96.82**	95.05	NA
Print #3	92.03	94.90	**95.71**	94.10	NA
Print #4	95.10	95.10	**96.25**	94.77	NA
Egg	92.53	98.48	**99.60**	98.98	97.56

outperform all other clustering approaches and the conventional connected components algorithm is far behind the others. This is because usually objects contain pixels which are not connected, and hence connected components generates by far too many objects (as e.g. shown in Fig. 4c, d). For the egg data the situation is similar (connected components falls short in classification accuracy by more than 6%), whereby DBSCAN can outperform the other clustering approaches. In fact, it is remarkable that DBSCAN can even improve the already high accuracy of 98.89% from hierarchical clustering to 99.54%. This means that the error (100% accuracy) is reduced by more than a half.

Now, the remaining question is: are these results really significant? Or the other way round, is the classification method maybe biased in a way that it favors a specific object extraction method? To answer this question two other types of classifiers are applied, SVMs and possibilistic NN, with the results shown in Tables 3 and 4. These tables confirm the statements made above for the different data sets (and hence no favor of any object recognition approach towards a specific classification method is observed): for artificial data normalized cut performs best, while for CD imprint and egg data DBSCAN can outperform all others, leaving connected components far behind. Note that in all tables no evaluation with the normalized cut clustering algorithm on the CD imprint data could be carried out (hence shown as NA). This was due to the heavy memory usage (which caused virtual memory overflow) and too high computational effort (during eigenvector calculations) inherent to this kind of algorithm when a high number of pixels has to be processed (note that some CD imprint images contain up to a few hundred thousands of bright pixels). However, experiments have been carried out with contrast images, containing only a small number of bright pixels. These show that the normalized cut has the potential to improve the classification accuracy (at least) in a similar way as DBSCAN does.

Table 4 Classification accuracies achieved by possibilistic NN classifier when using different object extraction methods, the best performing ones given in bold font

Possibilistic NN	CC	HC	DBSCAN	RDG	NCUT
Art. #1	83.08	83.22	82.80	82.00	**86.06**
Art. #2	89.10	89.31	88.52	88.38	**89.32**
Art. #3	91.38	91.12	91.88	90.95	**92.60**
Art. #4	84.86	85.17	86.30	84.37	**90.50**
Art. #5	90.02	90.31	88.88	88.13	**90.39**
Print #1	87.20	93.20	**94.56**	92.98	NA
Print #2	93.64	96.01	**95.99**	95.43	NA
Print #3	91.82	93.79	**95.71**	92.34	NA
Print #4	91.96	**95.56**	95.15	92.83	NA
Egg	91.07	98.62	**99.74**	99.39	98.12

6 Conclusion

In this paper, it was examined how various object extraction approaches may influence the predictive performance of image classifiers, when they are trained using features extracted from the objects found in a pre-labeled set of contrast images. This is an important aspect at the surface inspection system, as the image classifiers are responsible for the quality of the production items and hence the costs for re-production and customer complaints. The final conclusion is that clustering approaches can significantly outperform standard methods such as connected components algorithm and that different clustering algorithms may result in different classifier accuracies. Basically, there is a favor for DBSCAN algorithm as it was among the top performers for all data sets and classification methods. Furthermore, this clustering approach is optimized for processing a large number of data points that is usually needed when grouping large clouds of pixels. Three different well-known classification methods were used

in order to eliminate an eventual bias of a classifier when coupled with a certain object extraction method. In this light, and as ten different data sets were used, the final statements about the impact of the object extraction approaches on classifier performance can be seen as quite significant. Nevertheless, in future work other clustering methods have to be analyzed, especially those which have been developed for time-critical applications such as grid-based clustering [31]. Also, there may be an improvement when clustering the objects found by the connected component algorithm instead of the original bright pixels (which means using the connected component algorithm as a preprocessing step before clustering). Future enhancements may also include the incorporation of application-specific background knowledge in terms of clustering constraints [30] and/or the combination of different clustering results using so-called cluster ensembles [28].

Acknowledgments This work was supported by the European Commission (project Contract No. STRP016429, acronym DynaVis). This publication reflects only the authors' views.

References

1. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A.: Classification and Regression Trees. Chapman & Hall, Boca Raton (1993)
2. Caleb-Solly, P., Smith, J.E.: Adaptive surface inspection via interactive evolution. Image Vis. Comput. **25**(7), 1058–1072 (2007)
3. Daszykowski, M., Walczak, B., Massart, D.L.: Looking for natural patterns in data. Part 1: Density based approach. Chemom. Intell. Lab. Syst. **56**(2), 83–92 (2001)
4. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Second International Conference on Knowledge Discovery and Data Mining, pp. 226–231. AAAI Press, Menlo Park (1996)
5. Gan, G., Ma, C., Wu, J.: Data Clustering: Theory, Algorithms and Applications. Society for Industrial and Applied Mathematics (SIAM), American Statistical Association, Philadelphia (2007)
6. Gonzalez, R.C., Woods, R.E.: Digital Image Processing, 2nd edn. Prentice-Hall, New Jersey (2002)
7. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On clustering validation techniques. J. Intell. Inf. Syst. **17**(2/3), 107–145 (2001)
8. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference and Prediction. Springer, New York (2001)
9. Hopkins, B.: A new method for determining the type of distribution of plant individuals. Ann. Bot. **18**, 213–226 (1954)
10. Iivarinen, J., Rauhamaa, J.: Surface inspection of web materials using the self-organizing map. In: Casasent, D.P. (ed.) Proceedings of the SPIE, vol. 3522, Intelligent Robots and Computer Vision XVII: Algorithms, Techniques, and Active Vision, pp. 96–103 (1998)
11. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Comput. Surv. **31**(3), 264–323 (1999)
12. Hashim, A.A., Campbell, J.G., Murtagh, F.: Flaw detection in woven textiles using space-dependent fourier transform. In: Owens, F.J. (ed.) ISSC '97, Irish Signals and Systems Conference, pp. 241–252
13. Kim, C.W., Koivo, A.J.: Hierarchical classification of surface defects on dusty wood boards. Pattern Recognit. Lett. **15**, 712–713 (1994)
14. Kubota, T., Talekar, P., Ma, X., Sudarshan, T.S.: A nondestructive automated defect detection system for silicon carbide wafers. Mach. Vis. Appl. **16**, 170–176 (2005)
15. Niskanen, M.: A visual training based approach to surface inspection. PhD thesis, Department of Electrical and Information Engineering, University of Oulu (2003)
16. Ozdemir, S., Baykut, A., Meylani, R., Ertüzün, A., Erçil, A.: Comparative evaluation of texture analysis algorithms for defect inspection of textile products. In: Proceedings of the International Conference on Pattern Recognition, pp. 1738–1741
17. Papari, G., Petkov, N.: Algorithm that mimics human perceptual grouping of dot patterns. In: Proceedings of the First International Symposium on Brain, Vision and Artificial Intelligence BVAI, Naples, October 2005, vol. 3704, pp. 497–506. Springer, Berlin (2005)
18. Pernkopf, F.: 3D surface analysis using coupled hmms. Mach. Vis. Appl. **16**(5), 298–305 (2005)
19. Runkler, T.A.: Information Mining: Methoden, Algorithmen und Anwendungen intelligenter Datenanalyse. Vieweg, Gabler—Computational Intelligence, April 2000
20. Salvador, S., Chan, P.: Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms. In: ICTAI 04 (2004)
21. Schael, M.: Texture fault detection using invariant textural features. In: Radig, B., Florczyk, S. (eds.) Proceedings of DAGM 2001, Pattern Recognition. Lecture Notes in Computer Science, vol. 2191, pp. 17–24. Springer, Berlin (2001)
22. Schölkopf, B., Smola, A.J.: Learning with Kernels—Support Vector Machines, Regularization, Optimization and Beyond. MIT Press, London (2002)
23. Shapiro, L.G., Stockman, G.C.: Computer Vision. Prentice-Hall, New Jersey (2001)
24. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **22**(8), 888–905 (2000)
25. Sibson, R.: Slink: an optimally efficient algorithm for the single-link cluster method. Comput. J. **16**(1), 30–34 (1973)
26. Smith, M.L.: Surface Inspection Techniques: Using the Integration of Innovative Machine Vision and Modelling Techniques. Professional Engineering Publishing, London (2000)
27. Stone, M.: Cross-validatory choice and assessment of statistical predictions. Journal of the Royal Statistical Society **36**, 111–147 (1974)
28. Strehl, A., Ghosh, J.: Cluster ensembles—a knowledge reuse framework for combining multiple partitions. J. Mach. Learn. Res. **3**, 583–617 (2002)
29. Vapnik, V.: Statistical Learning Theory. Wiley, New York (1998)
30. Wagstaff, K., Cardie, C.: Clustering with instance-level constraints. In: Proceedings of the Seventeenth International Conference on Machine Learning, pp. 1103–1110 (2000)
31. Wang, W., Yang, J., Muntz, R.R.: STING: a statistical information grid approach to spatial data mining. In: Twenty-Third International Conference on Very Large Data Bases, Athens, Greece, pp. 186–195. Morgan Kaufmann, San Francisco (1997)
32. Wasserman, P.D.: Advanced Methods in Neural Computing. Van Nostrand Reinhold, New York (1993)
33. Wolpert, D.H.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1997)