Adaptive Part Inspection Through Developmental Vision

Gil Abramovich
Department of Mechanical Engineering, The University of Michigan, Ann Arbor, MI 48109
e-mail: gabramov@engine.umich.edu

Juyang Weng
Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824
e-mail: weng@cse.msu.edu

Debasish Dutta
Department of Mechanical Engineering, The University of Michigan, Ann Arbor, MI 48109
e-mail: dutta@engine.umich.edu

We present a novel online inspection method for manufacturing processes that automatically adapts to variations in part and environmental properties. This method is based on a developmental learning architecture comprising a procedure that focuses attention to apparently defective regions, a recognition method that performs automatic feature derivation based on a set of training images and hierarchical classification, and an action step that controls attention and further decision processes. The method adapts to variations incrementally by updating rather than recreating the training information. Also, the method is capable of inspecting and training simultaneously. Addressing new inspection tasks requires neither re-programming and compatibility tests, nor quantitative knowledge about the image set, from a human developer. Instead, automatic or manual training of the inspection system according to simple guidelines is applied. These attributes allow the method to improve online performance with minimal ramp-up time. Our system performed inspection of three applications with low error rate and fast recognition, confirming its suitability for general-purpose, real-time, online inspection.
[DOI: 10.1115/1.2039103]

Keywords: Inspection, Pattern Recognition, Developmental Vision, Appearance-Based Method, Invariance, Feature Derivation, Classification, Complexity

1 Introduction

1.1 General. The work presented in this paper is dedicated to defining the structure of automated system architecture for online part inspection in a manufacturing line. This architecture embodies a novel implementation of a developmental learning concept in machine vision inspection. We use state-of-the-art algorithms to demonstrate capabilities of this architecture.

1.1.1 Machine Defect Inspection. The essence of recognition-based inspection is assigning images of parts, or regions of parts, to one of several classes, where a class is a group of objects sharing an attribute (traditionally, manufactured components are called "parts"). One form of pattern recognition uses an approach called Supervised Learning, which creates a model by feeding a system with pairs of inputs and desired outputs. In a defect inspection application, a human trainer, or the inspection machine (autonomously) captures images ("inputs") and feeds also their associated class labels ("outputs"). A class label is a descriptor for the class, which can be a name, a number, or a vector, such as "Defect-Free," "Containing Defect Number 1," "Defect Number 2," "[1, 0, 0, 0]," etc. The computer inspection program creates a representation of the different classes and the decision rules for classification, using the training images and their classes. It later classifies inspected images (i.e., matches class labels to them) according to these decision rules.

The inspection system is stationed along the production line of a manufacturing system, inspects and classifies manufactured parts, and physically sorts or sends commands to the manufacturing system to sort each of them. Defect inspection is common in various manufacturing industries: part machining [1], semiconductor wafer production [2], wood manufacturing [3,4], ceramics manufacturing [5], manufacturing of fabrics [6], and others. In this paper we concentrate on visible defects the images of which are captured by cameras. However, the method introduced here is useful for detection of internal defects as well, using images captured with other devices, such as ultrasound or x-ray.

1.1.2 The Environment: RMS and its Demands. A Reconfigurable Manufacturing System (RMS) imposes special inspection requirements. An RMS is "one designed at the outset for rapid change in its structure, as well as its hardware and software components, in order to quickly adjust its production capacity and functionality in response to sudden, unpredictable market changes as well as introduction of new products or new process technology" [7]. An inspection system for an RMS must adapt in real-time to new parts or inspection tasks. Traditional inspection methods that incorporate manual feature selection, such as filters, morphological operations, correlation, spectral methods, and model-based pattern matching, are labor-intensive and time consuming, since they require developing a separate algorithm for each task, and, therefore, cannot successfully address these demands.

1.1.3 Using Manual Visual Inspection as a Guide to Designing Machine Vision Inspection. Manual inspection is labor intensive, subject to human errors and to worker-to-worker discrepancy, and is often slow. However, the human inspector is versatile and is capable of specializing to perform new tasks by learning new visual scenes and reducing the sensitivity to unused visual information. The human visual system accumulates training experience as time evolves, and acquires the property of partial invariance to translation [8], scale [9], or orientation [10]. We incorporate these attributes in the proposed synthetic vision inspection system.

1.1.4 Structure of the Approach: Artificial Developmental Learning (ADL). Artificial Developmental Learning is an approach in machine learning that achieves specialization of a machine vision system for specific tasks, in three stages:

1. Programming stage (human operated): The computer program that operates the machine vision system is tailored around constraints, such as the physical domain where the system operates. Such hard-coded constraints include image capture rate, image intensity range, magnification of the inspected object, and computer hardware limitations of
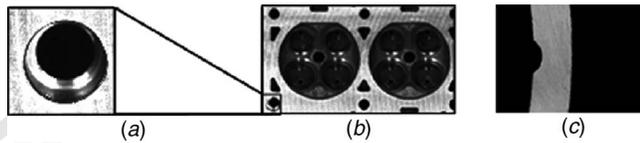
Fig. 1 Dimensional landmarks: (*a*) Locating hole (left) on an engine head (right); (*b*) circular part—the notch can serve as an orientation and a positioning landmark

Fig. 2 Arrangement of different applications according to position relevance

memory and processing speed. The programming stage is the only labor intensive stage of ADL.

2. Developing stage (autonomous or supervised): The program specializes in a specific task, such as finding defects on textured surfaces by visual exploration (i.e., creating new decision rules using previous recognition results) or by supervised training. The program can discard from the memory unnecessary tasks that have not been encountered recently, and thus, it dedicates resources only to relevant tasks. Depending on the application, this stage is either autonomous or requires minor human involvement.

3. Performance stage (automatic): The system performs the task that it has specialized in.

We focus on a concept we name "developmental vision," which is an autonomous process of developing a specialization for new tasks by visual experience. We describe this concept in detail in Sec. 2.

1.1.5 Variations in Manufacturing Flaws. Typical flaws can roughly be classified into (a) part-dimension errors, (b) surface defects, and (c) internal defects. Surface defects can further be classified into distinct textural patches [see Fig. 10(*a*)] and into smaller defects, which appear as irregularities in the texture (e.g., porosity). Textural surface defects appear as textural patches with contrasting surrounding defect-free background.

Dimension measurement is used for creating a database or for comparison with an existing database. One implementation of dimension measurement is manual or automated detection of geometrical features, followed by measurement of the distances between them. For example, dimensional verification of an engine head requires detecting locating holes [Fig. 1(*a*)], while the measurement of another circular part [Fig. 1(*b*)] requires detecting and locating the notch for positioning and orientation detection. We call such geometrical features "dimensional landmarks." Dimensional landmarks are non-textural objects or patterns, which are identified by their shape and location.

1.2 Combined Inspection Requirements and Related Training.

1.2.1 Maximizing Similarity of Algorithms. Usually, inspection is carried out for more than one property of a part. Inspection tasks include, for instance, landmark detection together with surface inspection. We attempt to accelerate the ramp-up of the inspection system from reconfiguration to operation by creating a method that handles, with only minor changes, different recognition procedures for a specific part. In practice, multiple versions of the algorithm, with minor variations, are implemented in a computer inspection program, and are run simultaneously.

1.2.2 Key Example: Dependence of an Application on Position. Images can be clustered into different types according to their level of dependence on a specific feature, such as position (Fig. 2). A periodic texture is position-independent since position information is not relevant to the recognition task and unnecessarily increases the variations within each class (right side of Fig. 2). A general texture is a repetition of basic elements containing several pixels with quasi-periodic or random distribution. For instance, the image of a grinded surface may contain position-
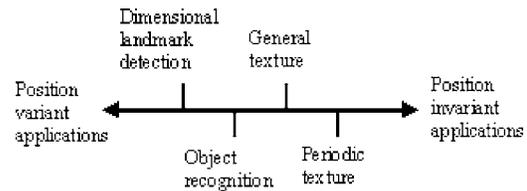
dependent variations. This may be due to variations of the machining process across the part or because of perspective distortion in the image, created by the viewpoint of the camera relative to the inspected surface. On the other extreme, landmark detection is a position-dependent application, since position is part of the identity of a landmark (left side of Fig. 2).

Throughout this paper we deal with tasks with different significance of the position, in order to demonstrate our system's capability of handling variations in inspection tasks. Traditionally, solutions to different inspection tasks are mutually exclusive. To overcome this limitation and create a unified technique, we replace dedicated mutually exclusive techniques, such as the Harris Corner Detector [11], or the Hough Transform for shape recognition [12], with a general-purpose method that allows training of an inspection system to perform one of a wide variety of inspection tasks.

1.2.3 Invariance Methods for Training a Machine. Although machine training is usually an offline procedure, it influences the real-time performance because a large training set may slow the inspection process. A way to reduce the number of training images is to resort to a process by which the influence of a specific feature on every captured image is ignored. Such a process is called an invariance process. In pattern recognition, a feature that should be ignored is one that is not useful for discriminating between classes, and typically has large variations. It thus increases the image set variation within each class and reduces the separation between classes, which are undesirable consequences.

Invariance algorithms for position, rotation or scaling include Gabor filters [13,14], FFT-based techniques [15,16], fractal dimensions [17], steerable pyramids [18], and Fourier Radially-Mellin descriptors [19]. These "hard-wired" methods require prior knowledge of the inspected images (for example, some existing narrow frequency-bands in a textural image), and are not compatible with a general, unknown setting (see Table 1 Column 3). Many of these methods require pre-determined feature correspondence, which is inconsistent with the goal of automatic inspection under varying conditions.

1.3 Automated Feature Derivation and Classification. Traditional automatic inspection methods incorporate a manual process of matching and modifying a combination of algorithms to carry out a specific inspection task. These methods require quantitative knowledge of all the variations in the application in order to choose a suitable method, which is not always possible. The necessity of using this manual process can be removed by using artificial neural networks. These have been already applied to surface inspection [20–23], or to object recognition and segmentation [24]. However, neural networks typically do not perform well in high-dimensional scenes, such as those encountered in the case of high-resolution images.

1.3.1 Appearance-Based Methods: General Structure. Appearance-based methods were introduced in computer vision to address the issue of automating the selection of discriminant features for high-dimensional scenes. Appearance-based methods amount to the following steps:

1. Training step:

Table 1 Training considerations: Three ways of addressing invariance for specific features variations

	1. Exhaustive training for invariance	2. Division into subclasses to address variance of non-discriminating characteristics	3. Feature-based pre-processes
Application Example	Variations in non-measurable features Each surface class has variations in appearance	Variations of measurable features Textured surface in random orientation; Object at different positions	Explicit a priori knowledge about the image Periodic texture
Advantages	Handles variations which are not explicitly quantifiable	Small number of images is required for every subclass	Small number of images is required for every subclass
Disadvantages	Large number of images in every class potentially results in a high dimensional problem	Large number of subclasses potentially results in a high dimensional problem	Non-adaptive; require compatibility tests

(a) Concatenate each training image into a training vector by linking the rows serially.

(b) Create or update an automatic projection procedure from the training vectors from (a) (as a batch or one at a time). The resultant subspace is called the feature subspace and contains projections of the training vectors or some representation of their population. The set of the feature vectors is a basis of this new feature subspace.

(c) The projected training vectors desirably create more distinct groups of similar images than in their original form.

2. Performance (test) step:

Repeat (a) for the inspected images and project them onto the feature subspace. Automatic classification is performed in the feature subspace.

Feature derivation is, therefore, an operation where the features are automatically learned from the training image set. The appearance-based methods extract information about the correlation between nearby pixels as well as that between distant pixels. The process of automatic feature derivation is presented in Fig. 3.

1.3.2 The Choice of the Appearance-Based Method. The choice of an automatic feature-derivation and classification method is critical for a good performance of the inspection architecture. The guidelines for selecting of such a method ensure the capability of an automatic adaptation to a new set of training images, a low error rate, high-sensitivity to variations in the image, high processing speed, and low memory consumption. Compatibility with online inspection tasks requires that the method incrementally update the database using newly acquired training images, without the need to retrain the system by using the whole training database.

Several appearance-based methods are listed in the following:

(1) Principal Component Analysis (PCA). This method projects the samples onto the directions, along which the scatter of the data points is the greatest [25]. The feature vectors are the eigenvectors of the scatter matrix built from training images that correspond to the largest eigenvalues. A disadvantage of PCA is that it may capture variations that are useless for classification. Furthermore, the PCA is not a supervised learning method and therefore does not use class labels when these are available. (2) Linear Discriminant Analysis (LDA) [26] combines dimensionality reduction and
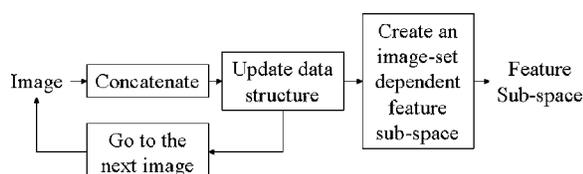
classification. It seeks projection directions along which the ratio of the scatter of class means to the scatter within the classes is maximal. (3) Independent Component Analysis (ICA) [27]. (4) Multidimensional Scaling (MDS) [28]. (5) Local Non-Negative Matrix Factorization (LNMF) [29]. (6) Hierarchical Discriminant Regression (HDR) [30]. The latter, in its incremental form, the IHDR, is described next, and was selected for this work. Details can be found in [31,32].

1.4 The IHDR Method. In this paper, we use the term *sample* when referring to a single image of a part, or an image of a region on the part that includes a defect or a section of a defect. A *cluster* is a group of samples having some similarity. A *pure* cluster is one whose samples belong to a single class. A *node* is a decision junction in a decision tree, which divides its input data into multiple branches. *Leaf* nodes are the final nodes in the tree, beyond which there are no further nodes. Figure 4 presents two clustering hierarchies [(*a1*) and (*a2*) vs (*b1*) and (*b2*)], and the corresponding tree nodes, encircled in (*a3*) and (*b3*), respectively. Note in (*a1*) and (*b1*) that a cluster mean marked by "⊕" is the average of the means of all its samples. The IHDR is a regression
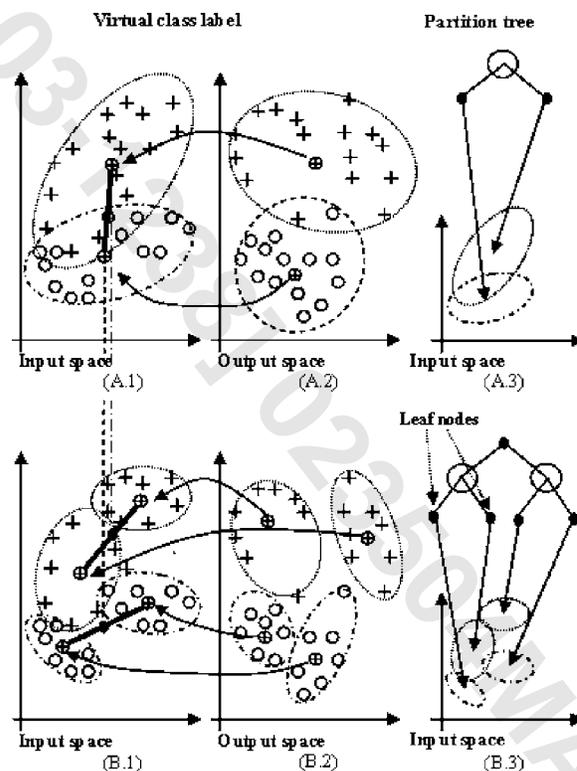

Fig. 3 Automatic feature derivation

Fig. 4 (*a*) Coarse clustering; (*b*) fine clustering. (*a.1*) and (*b.1*) represent the input space, (*a.2*) and (*b.2*) represent the output space, (*a.3*) and (*b.3*) represent the IHDR tree node space

tree, which associates to each input vector with an output vector in such a way that the mapping from the input to the output space is smooth. Since we are dealing with a classification problem, we need to introduce special procedures to convert the classification problem into a regression problem, so that the training images are not labeled by a multivariate vector rather than a single value.

1.4.1 Matching Samples. In the IHDR, a *training* image is introduced and the decision tree is updated for one image at a time. This is achieved by finding the best-matching leaf node and the closest cluster in the node. The best matching cluster is found by computing the average of the normalized distance in the output space:

$$m = \arg \min_{1 \leq i \leq q} \left(\frac{\|\mathbf{y} - \bar{\mathbf{y}}_i\|}{|\mathbf{y}|_{\text{est}}} \right) \quad (1)$$

where $\mathbf{y}$ is the output label vector, $\bar{\mathbf{y}}_i$ is the ith cluster mean in the output space, and $|\mathbf{y}|_{\text{est}}$ is the estimated average norms of $\mathbf{y}$. The clustering is done in the output space, and is used to form the corresponding $\mathbf{x}$ cluster. This procedure is called double clustering. Then the new sample is used to update the means of the mth cluster in both input and output space:

$$\bar{\mathbf{x}}_m(n) = \frac{n-1-\mu(n)}{n} \bar{\mathbf{x}}_m(n-1) + \frac{1+\mu(n)}{n} \mathbf{x}$$

$$\bar{\mathbf{y}}_m(n) = \frac{n-1-\mu(n)}{n} \bar{\mathbf{y}}_m(n-1) + \frac{1+\mu(n)}{n} \mathbf{y} \quad (2)$$

where $\mathbf{x}$ is an input sample, n is the index number of a sample arriving at the mth cluster, $\bar{\mathbf{x}}_m(n-1)$, $\bar{\mathbf{y}}_m(n-1)$ are the estimates prior to the update, and μ is an update rate function. In the case where the number of training prototypes in the node exceeds a predefined value, the node stops serving as a leaf node, and sprouts new leaf nodes. Outdated nodes that have not been recently updated by new prototypes are eliminated.

1.4.2 Creation of a Feature Subspace. In order to have a compact and rapid classification procedure, only the mean and the covariance matrix of each cluster, as well as a projection matrix to the local feature subspace are preserved at each node. The projection matrix to the local subspace D is computed by the Gram-Schmidt Orthogonalization (GSO) algorithm (see, e.g., [30]). The local discriminating feature subspaces are hyper-planes that pass through the means of all clusters belonging to a specific node [see the thick lines in Fig. 4 (*b1*)]. When a new *inspected* sample is captured, it chooses the closest cluster of samples according to the Size-Dependent Negative-Log-Likelihood (SDNLL) metric, defined by an approximate Gaussian density of dimension q-1:

$$l_i(x) = \frac{1}{2}(x-c_i)^T W_i^{-1}(x-c_i) + \frac{1}{2}\ln(|W_i|) \quad (3)$$

where l_i is the distance and c_i is the cluster sample mean and sample covariance matrix. W_i is a weighted mean of covariance matrices of the Euclidean Negative-Log-Likelihood (NLL), the Mahalanobis NLL, and the Gaussian NLL. Each of these performs best for a different range of numbers of samples. Keeping the most representative metric is ensured by weights incorporated into W_i [30]. The advantage of the SDNLL metric is in its ability to efficiently handle small, large or unbalanced clusters of samples. When using natural images in general, and manufacturing flaw images in particular, different classes may contain different numbers of samples. The ability of the IHDR tree to deal with a varying number of samples is, therefore, beneficial to inspection applications.

1.5 Structure of the Paper. In Sec. 2 we propose specific architecture and describe how it handles invariance. Section 3 gives general system training guidelines with the goal of achieving invariance in various inspection cases. Section 4 provides an analysis of the process complexity, showing that the method (ar-
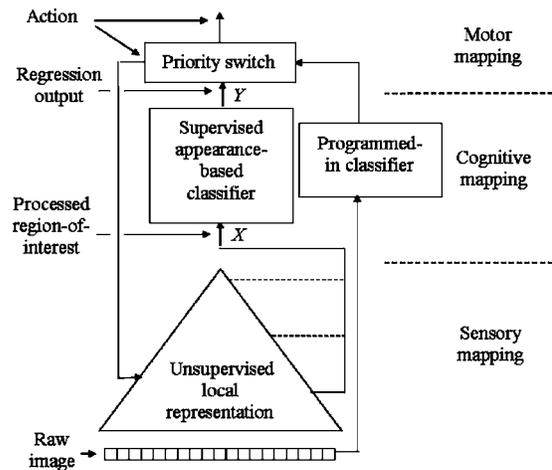

Fig. 5 A general-purpose developmental vision architecture, divided into sensory, cognitive, and motor maps

chitecture and chosen algorithms) is computationally compact and that, therefore, it is compatible with real-time inspection requirements. We present in Sec. 5 three different case studies, representing the range of position relevance—textural surface defect inspection, landmark detection, and object orientation recognition. We provide experimental results and discuss them. Finally, we provide conclusions in Sec. 6.

2 A New Concept: Adaptive Inspection Through Developmental Vision

2.1 Developmental Vision. Developmental Vision is an approach that utilizes developmental learning for visual tasks. Automated developmental vision enables a vision system to learn new tasks (not predicted by the programmer) autonomously by experiencing its environment. It consists of programming, and an autonomous and supervised development, which allow the system to explore and learn visual scenes, and gain expertise. Actual performance (recognition and decision making) is intertwined with incremental training and exploration at this stage. A program that guides this development of a vision system is called a developmental program. The tasks that the vision system will perform are unknown to the programmer at the time of writing. This capability matches in particular, the requirements of an RMS, where the application is unknown at the time of the RMS design, but a wide range of tasks may arise during the life of the system.

2.2 System Architecture. The architecture of our proposed developmental vision system, as shown in Fig. 5, consists of a local representation method of the image, that allows the system to selectively focus attention on regions-of-interest (described in the following), for example, on potential flaws; an appearance-based feature deriver and classifier; and a parallel route of programmed-in (not learning-based) classifier. The developmental vision architecture incorporates algorithms for selective attention to candidate regions, for recognition of the chosen regions and for an action process (referred to here as motor mapping) that translates classification results to mechanical part sorting or attention control (i.e., it controls the local image representation method to determine what is the next region-of-interest in the image). It comprises the following mappings: sensory mapping, cognitive mapping, and motor mapping.

2.2.1 The Sensory Mapping—Selecting Features for Attention. The sensory mapping is defined as a mapping applied directly to sensory output (the acquired image). In the instance we describe here, it is a set of specific derivation filters applied to sub-images at different scales (i.e., digital zoom setting). Figure 6 displays a
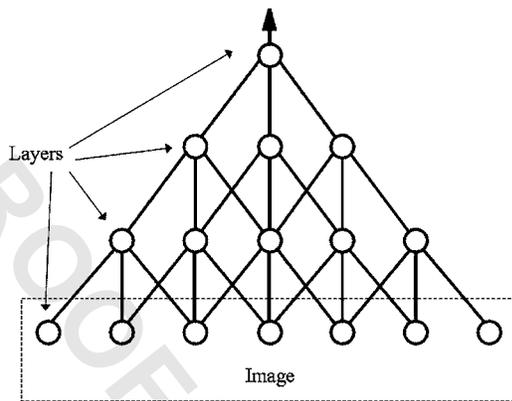
Fig. 6 Hierarchical sensory mapping

multi-layer sensory mapping. In each layer, the field-of-view has a certain size—the higher the layer in the hierarchy, the larger the field-of-view. Every layer and position in the layer defines a field-of-view, which is filtered. Here, attention selection is the process of activating specific fields-of-view.

These derived features are dedicated to attention selection of regions of interest, such as potential defects. The sensory mapping is implemented by: (1) A hard-coded set of filters that create saliency of regions-of-interest relative to the background (an example is presented in [33]). For instance, in the lower image of Fig. 9(b), automatic visual attention to an object on the conveyer is created by intensity contrast, and the result is an object sub-image with minimal background. This type of hand-designed saliency is effective for high object-to-background contrast, but may not perform well for unpredictable environment or inspected components. (2) An unsupervised learning mechanism that segments the image into a region-of-interest (e.g., "candidate defect") and background (e.g., "defect-free"). The features that distinguish between these two regions are derived automatically. A hierarchical developmental sensory mapping of the second type was proposed in [34].

2.2.2 The Cognitive Mapping for Feature Derivation and Recognition. The cognitive mapping represents the image understanding stage. It encompasses procedures that derive discriminating features from the outputs of the sensory mapping and classifiers in the feature space. In the proposed approach the cognitive mapping is realized by the IHDR and in parallel, by a module of a classifier that does not require learning (see Fig. 7). The latter is active when the former is being trained, prior to accumulation of a sufficient amount of learning experience.
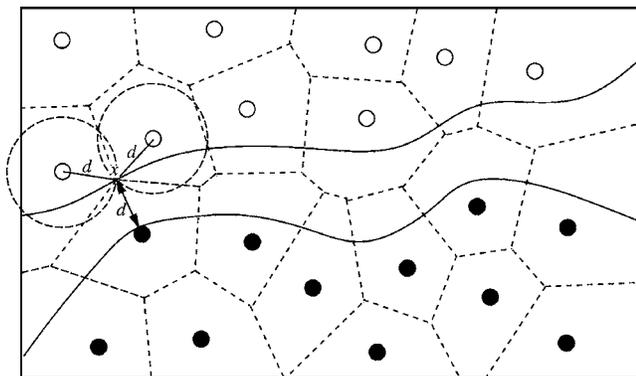

Fig. 7 Positively separable regions: The black and white dots denote two different classes. *d* is the minimal separation zone breadth

2.2.3 The Motor Mapping. The motor mapping is the action process that is applied after the cognitive mapping. Based on classification results, it controls and chooses the recognition path (developmental or programmed-in), controls the attention path according to previous recognition, or performs the mechanical sorting of the parts.

3 General Training Guidelines for Handling Flaw Variations

A factor to be considered when one defines a training procedure concerns the amount of prior knowledge that directly relates to the image. This information includes ranges of variations in shape, dimensions, texture characteristics, and position. For instance, defects on grinded surfaces may contain major variations in texture characteristics and in the orientations of the cutter marks in defective areas and in the non-defective areas [see Fig. 10(a)].

Other types of data that need to be considered are lighting, camera viewing angles, and camera characteristics. All possible types of acceptable images should be represented in the training set, or be handled by invariance procedures applied on the input images (see Sec. 1.2 for definitions and further details).

The proposed architecture (Sec. 2) and algorithms circumvent these problems and lead to faster ramp-up to operation and to possible improvements in inspection performance. In the next sections we discuss basic definitions and theorems, propose a training strategy for three general cases, and provide a formula for a sufficient number of training images.

3.1 Feature Sufficiency and Invariance. Feature sufficiency was defined in [35] in the following way: A classification method is *feature sufficient* for an environment E if in its feature space L, for any two objects O_1 and O_2 to be distinguished in the environment E, the corresponding feature value sets are disjoint: $T(O_1) \cap T(O_2) = \theta$ [35]. Therefore the feature vectors are sufficient for discriminating any two objects in the environment. The Feature Sufficiency Theorem [35] asserts that if C is a classification system that is error-free in an environment then C uses enough discriminating features in that environment.

The Perfect Invariance Theorem [35] asserts that given (a) an environment E represented by a finite number of features, (b) a feature sufficient classifier in E, and assuming that (c) there are a finite number of objects in the environment whose feature sets are mutually positively separable, then, there is a mapping that classifies all the possible samples with zero error; in other words, perfect invariance to non-discriminating features in E. Further, the classifier can be constructed from *a finite* number of training samples by an explicit algorithm.

Each training sample (image) has a neighborhood whose population of images will be correctly classified. In Fig. 7, the black and white dots represent training samples of two classes. The solid curves denote the boundary of the positively separable regions. The dashed lines indicate the decision boundaries, which lie at equal distance (d in the figure) from the neighboring training samples.

Definition 1: Two regions, r_1 and r_2, are *positively separable* in the space L if the distance between them, $d = \inf\{\|x_1 - x_2\| | x_1 \in r_1, x_2 \in r_2\}$, is a non-zero positive number, where x_1 and x_2 denote samples belonging to class 1 and 2, respectively. That is, a finite set of training samples that define positively separable regions is sufficient for achieving zero classification error for positively separated regions.

Using the feature sufficiency and invariance concepts, we define training guidelines for typical inspection scenarios. These guidelines can be followed by a line operator and require only simple, qualitative understanding of the inspection task and its requirements. The idea is to ensure that invariance holds for a general inspection task and limit the dimension of the feature subspace and the number of training images by the available quantitative information without losing discriminant information.
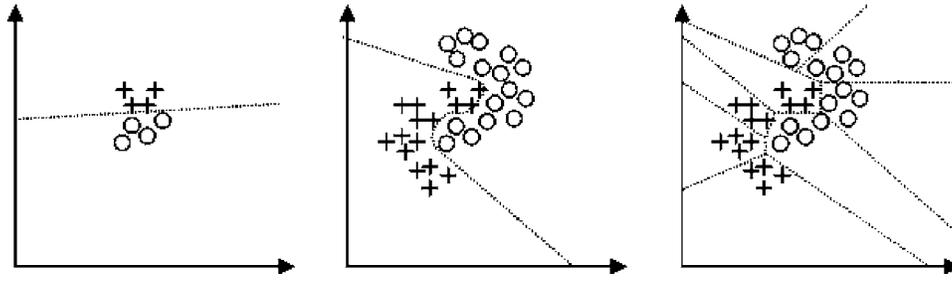
Fig. 8 Sample space and decision boundaries: (*a*) Separable classes; (*b*) same classes, with the introduction of variation in a specific property; (*c*) same as (*b*), with separation into subclasses resulting in low-degree boundaries between subclasses

3.2 Proposed Strategy. There are two types of non-discriminating features for which invariance needs to be yielded: (a) non-measurable (appearance-based) features and (b) directly measurable features (e.g., texture orientation). Case (a) requires exhaustive training of all the available ranges of image appearance. In the general case, samples of individual classes may be widely scattered according to non-Gaussian distribution. Hierarchical clustering autonomously creates positively separable subregions, so that a non-overlapping mixture of Gaussians is achieved (see the description of IHDR in Sec. 1.4). To describe how we handle case (b), assume that we are dealing with two classes (of texture, objects, or landmarks) which are easily separable, so that there is no overlap between classes, as shown in Fig. 8(*a*). Figure 8(*b*) shows the distribution of samples of the same two classes when a value range of a feature is measurable and is shared by all the classes. The human trainer divides each class into subclasses, where each subclass has only a narrow range of that feature [Fig. 8(*c*) and Table 1 Column 2]. This procedure replaces the first layer of autonomous hierarchical clustering in the IHDR. This clustering procedure is error-free and the resultant subclasses have significantly reduced variations. The practical improvement is that less training images are needed to train the system without losing discriminant information.

When even further quantitative knowledge is available, such as specific dominant frequencies, feature-based invariance methods are preferred (see Table 1 Column 3). For example, milled engine heads have a characteristic distance between marks, which depends on the cutter pitch and its speed relative to the head. For this periodic texture, the magnitude of an FFT operator achieves the position invariance. In [35], the FFT operator is applied on the well-known Brodatz set with excellent results.

Note that the method replaces software re-programming, which is necessary for conventional methods, by training. We acknowledge that training can be considered as a form of programming (though not in the software development sense). However, training according to the aforementioned guidelines can be described as a low level programming that can be easily performed by a person who is not highly specialized, using the inspection software as a "black box." Also, training according to these guidelines consumes substantially less time than software development. An additional effort that one needs to take into account in our method is the collection of training images. This, however, is a trivial effort. It may involve programming of the acquisition software to capture images at a specific timing, location, and orientation, but it does not require inspection algorithm re-programming.

3.3 Training Sufficiency. An important question is how many training images are required for a specific inspection task. In this section we establish a general estimate for this number. The formula for this estimate is dependent on the aforementioned training guidelines. It applies to each of the three cases described in the previous section.

Given s classes, either the system or the human operator determines the practical number of training images to use per class.

This number depends on two task-specific factors: (1) the number of widely varying non-discriminating features and, (2) the ability to measure the range in which each feature can vary. In any case, the training images must create positively separable regions. This requirement is complemented by the following definitions.

Definition 2: The Recognition Resolution Increment Δ_j, $j \in \{1, 2, \ldots, M\}$, is the minimal distance between values of a varying non-discriminating feature (where M is the number of these features) of two adjacent training images, that ensures that the training images can serve as distinct prototypes, without redundancy (see Fig. 7).

The recognition resolution may be non-uniform, depending on the concentration of the available training prototypes. Note that the distance is taken directly between measurable feature values and not between the images. The choice of a training set means that the continuous range of variations is represented by a finite number of images.

Definition 3: The Variation Quantization is the mapping of an infinite set of vectors (images) that represent a continuous range of a specific feature into a finite set of vectors (images) that represent discrete levels of that feature.

We define the following values:

j = the index number of a feature
i = corresponding quantization level index of the jth feature
x = an input image (in vector form) from a set that represents a continuous range of a feature
$\hat{x}$ = a representative training image picked by quantization
Q = the quantization function
$r'_{i,j}$ = the ith quantization level of the jth feature
N_j = the number of quantization levels of the jth feature

Then, the variation quantization is expressed as

$$\hat{x} = Q(x) = r'_{i,j}, \quad i \in \{1,2,\ldots,N_j\}, \quad i = i(j), \quad j \in \{1,2,\ldots,M\}. \tag{4}$$

To estimate the required number of training prototypes, assume uniform quantization with respect to the range of values R_j of a feature j. The number of training images that represent this range is

$$N_j = R_j/\Delta_j, \quad j \in \{1,2,\ldots,M\}. \tag{5}$$

Therefore, the overall number of training images is

$$p = \Pi_{j=1}^{M} N_j. \tag{6}$$

Here p is the number of training prototypes per class of images. When samples of a specific class are distributed according to a non-Gaussian law [Figs. 8(*b*) and 8(*c*)], a few clustering hierarchies are required and the minimal total number of training prototypes per class is

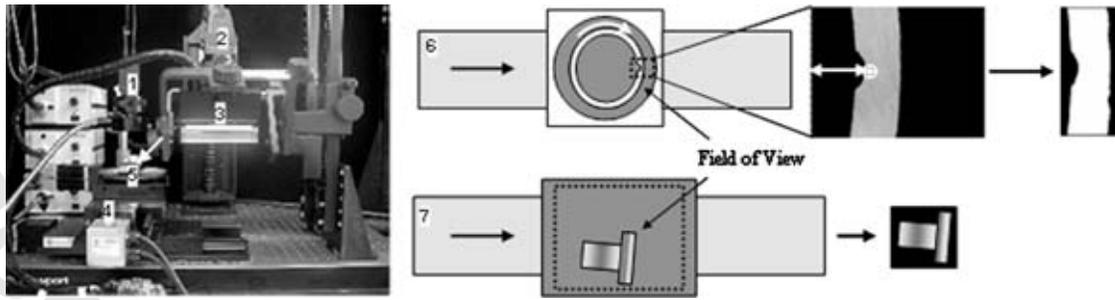$$p = K\Pi_{j=1}^{M} N_j, \quad K \gg 1, \tag{7}$$

Fig. 9 (*a*) Experimental setup. (*b*) acquisition strategy

where K is an integer representing the number of hierarchies.

We estimated the number of required training images for a specific task. This estimate will be used in the complexity considerations presented in the next section.

4 Complexity Considerations

The purpose of this section is to show that the proposed inspection architecture and specific implemented algorithms (particularly the IHDR) are compatible with real-time inspection tasks. We analyze the complexity of the method using symbolic notations that apply to any computing platform.

Our approach to complexity focuses on bounded available resources, which are the processing time and the required memory for computation. The bounds are determined by the inspection requirements (the rate of inspected parts, the region-of-interest resolution, required recognition rate), and by the available computer speed and memory. In this section, we estimate in a symbolic form the memory consumption and processing time for recognition. We assume that a decision tree classifier serves for the purpose of cognitive mapping.

Assume a d-dimensional training sub-image, and let the number of classes be s, and the number of prototypes per class p. Then, there are $n = sp$ training prototypes (distinguishable training images). It has been established [30] that the time complexity (O notation) for classifying a single image is $O(2qd \log(n))$ (where q is the splitting parameter). This logarithmic time complexity with respect to the number of training prototypes is extremely low. Assuming L sub-images per image, and using the formula of the number of required training images [Eq. (8)], the time required to retrieve the matched prototypes is

$$T = O(qd \log(Lsp)) = O(qd \log(Ls\Pi_{j=1}^{M}N_j)) \qquad (8)$$

and the space complexity, which is the memory required for storing the IHDR tree, is

$$S = O(dLsp) = O(dLs\Pi_{j=1}^{M}N_j). \qquad (9)$$

For example, assume a set of 100×100−pixel sub-images, which consumes 4 bytes per pixel in floating-point representation. Also assume that there are 100 sub-images per image, both rotation and scaling quantized to 10 different levels each, and the number of classes is $s = 5$. The IHDR tree, which is the major memory consumer, requires about 400 Mbytes, which is reasonable with today's personal computers. This analysis shows that in our approach, the recognition time is short provided sufficient memory is available. This applies to the two training approaches as described in columns 1 and 2 of Table 1.

When an external position invariant method is applied prior to cognitive mapping (Table 1 Column 3), it consumes time and memory space, but it may reduce the classifier time and memory consumption due to the decrease of the required number of training images. For example, the magnitude of the two-dimensional FFT has the following time complexity:

$$T_{\text{2D-FFT}} = O(d \log_2(d)), \qquad (10)$$

which is negligible. This operator only creates a new d-dimensional transformed image in two stages for every input image.

Using the case studies presented in the next section, we demonstrate the suitability of our approach for various inspection applications having real-time performance. These inspection applications differ from each other by their dependence on position information. However, they practically differ only in their training strategy.

5 Experiments

The purpose of the experiments is to demonstrate the versatility of the method by testing its performance in three inspection tasks: classification of textural surface defects, object orientation recognition for robot pick-and-place, and dimensional landmark detection. Specifically, with these case studies we show that the method can handle different levels of feature-relevance (position-relevance in this case). All the tests are disjoint, meaning that the test images are different from the training images. The tests were repeated using a cross-validation technique (see, e.g., [36]) from the same image set, each time different subsets of images were used for training and for testing to avoid possible bias in partitioning the image set into training and test sets.

5.1 System Setup and Acquisition Considerations. Considering the different possible production line settings, our experimental setup consists of two systems. The first system includes a linear camera, with two partially diffuse linear lights symmetrically positioned on both sides of the camera. The second system includes an area camera with a ring light. In both cases, the camera was positioned in a normal-to-surface orientation. Approximately normal-to-surface diffuse illumination was chosen in order to reduce reflection variations, which are caused by the interaction of illumination with the direction of the cutter marks. Figure 9(*a*) shows the various parts in the setup: (1) Area CCD camera with a ring light; (2) CCD linescan camera; (3) two symmetrically oriented line illuminators; (4) three-stage motion system which is shared by the two acquisition systems; and (5) a part which in this figure is incrementally scanned by the area camera setup to train the system for landmark detection. Figure 9(*b*) shows two forms of our acquisition strategy: at the top—a system with an area camera, a motion stage (conveyer), and a rotary motion-stage for active vision-based orientation detection (see Case Study 2, Sec. 5.3). At the bottom of Fig. 9, the system incorporates digital attention to the part and recognition. We used a regular Windows Personal Computer, with a 2.4 GHz Intel® Pentium® IV processor, and 1Gbytes RAM.

5.2 Performance Indicators. Table 2 summarizes the values used to evaluate the performance of our system. Conventional synonyms for each error term are enclosed in parentheses.

Table 3 represents schematically the form of the confusion ma-

Table 2 Performance terms

Performance terms	Acronym	Definition
Error rate	-	The ratio between the number of wrongly classified samples to the number of samples.
True accept rate (true negative) rate	TN	The rate of correct classifications of non-flaws, calculated as the ratio between the number of true accepts and the overall number of non-defective parts.
True reject (true positive) rate	TP	The rate of correct classifications of flaws, calculated as the ratio between the number of true positives and the overall number of defective parts.
False reject (false positive, type I error) rate	FP	The rate of misclassifications of non-flaws as flaws, calculated as the ratio between the number of false rejects and the overall number of non-defective parts.
False accept or false negative rate	FN	The rate of misclassifications of flaws as non-flaws, calculated as the ratio between the number of false accepts and the overall number of defective parts.
False subclass classification	FSC	Misclassification of one subclass as another subclass in the same super-class, for instance, classifying the defective surface correctly while misclassifying its orientation.
False classification	FC	Misclassification of one super-class as another super-class while retaining the corresponding super-class, e.g., classification of a defect as non-defect while recognizing the texture orientation correctly.
Training time per image/sub-image	-	The overall training time divided by the number of training images / sub-images.
Testing time per image/sub-image	-	The overall testing time divided by the number of training images /sub-images.
Confusion matrix	-	A matrix that displays the actual data labels against the classifier outputs. The matrix dimension is c^*c, where c is the number of classes.

trix for two-level classification used in this work. The confusion matrix has the dimensions $(N_1+N_2) \times (N_1+N_2)$, where N_1 is the number of subclasses in class 1 ("non-defects") and N_2 is the number of subclasses in class 2 ("defects"). Specific values, defined in Table 2 and represented in Table 3, were used for evaluating the inspection performance of the case studies represented next.

5.3 Case Studies.

5.3.1 Case Study 1–Textural Surface Defects on Machined Parts. For surface inspection, we focus on grinded metallic surfaces. These are challenging surfaces to inspect because the appearance of the surface is sensitive to lighting conditions. Specular reflection from a metallic surface is uneven and depends on the direction of the cutter marks. Also, surface defects often involve large variations in appearances of each class that are related to the uncontrolled process that caused them. In industrial settings, surfaces may also appear in any orientation, which is interpreted by an appearance-based method as a distinct texture class. Note that the decision boundaries between classes may vary with changing illumination conditions and that therefore the training procedure may vary as well. In our experimental setup, the appearance of the defective areas had a slightly different appearance relative to that of non-defective areas [see Fig. 10(a)].

We define a defective region as one class and the non-defective region as the other. We separate classes into subclasses according to texture orientation. Each band of 15 deg is regarded as a different subclass, corresponding to the class type and the orientation. The architecture is presented in Fig. 11(a).

Preparation of training images was done as follows: The system acquires images of defect-free training parts. The program segments each part from the background and extracts overlapping sub-images. This procedure is repeated by automatic rotation of each part into twelve different orientations, in an attempt to extract a sufficient number of sub-images for every texture orientation. The time for collecting defect-free training sub-images was negligible when compared to the training time and recognition time. The collection of training sub-images for defective areas was more involved. After the extraction of sub-images, sections of the image were manually sorted into defect-free and defective regions using a visualization program. Division into subclasses was manual, following the same procedure.

Table 3 A general confusion matrix for two-level classification using super-classes and subclasses

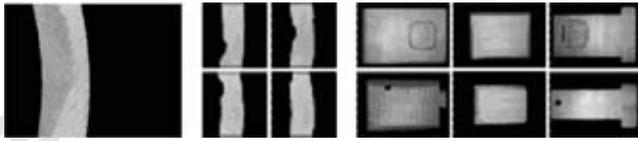|		Actual data						
		Negative (1)	Negative (2)	$\cdots$	Negative (N_1)	Positive (1)	Positive (2)	$\cdots$	Positive (N_2)
Classified data	Negative (1)	TN	FSC	$\cdots$	FSC	FN/FC	FN	$\cdots$	FN
	Negative (2)	FSC	TN	$\cdots$	FSC	FN	FN/FC	$\cdots$	FN
	Negative (N_1)	FSC	FSC	$\cdots$	TN	FN	FN	$\cdots$	FN/FC
	Positive (1)	FP/FC	FP	$\cdots$	FP	TP	FSC	$\cdots$	FSC
	Positive (2)	FP	FP/FC	$\cdots$	FP	FSC	TP	$\cdots$	FSC
	Positive (N_2)	FP	FP	$\cdots$	FP/FC	FSC	FSC	$\cdots$	TP

Fig. 10 Three case studies: (*a*) textural surface defects; (*b*) dimensional landmarks—notch of a piston ring; (*c*) object orientation detection

Our test set included 255 sub-images. They were divided into two super-classes—defects and non-defects. The "non-defect" super-class was further divided into twelve sub-classes, while the "defect" super-class was divided into five subclasses, making together seventeen subclasses. We consider samples assigned to different subclasses of the correct class to be correctly classified. We applied shift-invariant pre-processing, returning the magnitude of two-dimensional FFT of each raw image. For training, we sampled the field-of-view with overlapping sub-images, which correspond to the "staggered" sub-images, suggested by the proposed architecture of the sensory mapping [34]. We conducted a different test to confirm that the proposed training approach of dividing into subclasses enhances performance given a limited number of training images. We did it by repeating the experiment and assigning the images to classes "defect" or "non-defect" without breaking into subclasses.

Table 3 is a confusion matrix based on terms defined in Table 2, which includes a list of different possible types of correct and incorrect classification. The results are shown in the upper section of Table 4. These were selected from confusion matrices of the form presented in Table 3. 75 training sub-images and 15 test sub-images represented each subclass. The same experiment was repeated, this time without division into subclasses. A result of eliminating sub-classes and merging into only two super-classes was an increase in error rate, from 3.36% to 8.0%. The false accept rate increased from 2.86% to 21.43% [Table 4, experiments 1 and 2]. The average test (retrieval) time per sub-image dropped from 42 to 26 ms. This is due to the fact that in a 17-class case we used a higher value of splitting parameter q in the IHDR tree for superior recognition rate, while the value of q for the two-class case was 2. The increase in the value of q is accompanied by an increase in complexity [see Eq. (8)].

In order to determine which features are dominant—textural or orientation-related, we constructed a second set of images, where each image has a uniform intensity, equal to the mean intensity of the corresponding original image. By doing so, we removed the textural information. The error rate increased to 13.50%, with a 19.64% false reject rate (Table 4 Row 3), suggesting that texture contributes here discriminant features.

Next, the mean intensity of each image was digitally shifted to ensure identical means for all the images. The error rate increased
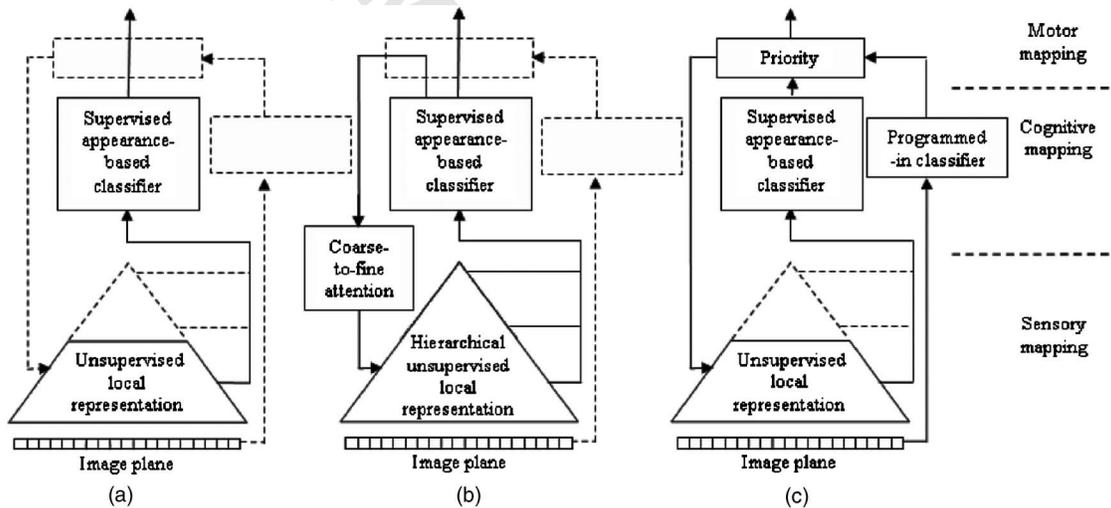

Fig. 11 Partial versions of our architecture: (*a*) architecture for surface defect inspection by texture analysis and object positioning; (*b*) architecture for landmark detection and location; (*c*) landmark detection by active vision with both low level "innate" position information and a developmental appearance-based classifier

Table 4   A. Surface defect detection; B. landmark detection and part positioning recognition

(A)

Experiment number and name	Number classes or subclasses	Error rate (%)	False accept rate (%)	False reject rate (%)	Training time per sub-image [s]	Test time per sub-image [s]	IHDR tree memory consumption [Mbytes]
1. Original texture sub-images	17	3.36	2.86	3.57	0.0754	0.042	5.2
2. Original texture sub-images	2	8.00	21.43	2.38	0.043	0.026	16.6
3. Synthetic mean sub-images	17	13.50	4.29	19.64	0.0398	0.0302	1.4
4. Shifted-means images	17	10.10	30.00	1.79	0.0262	0.0221	7.4

(B)

Application number and name	Number of classes or subclasses	Error rate (%)	Error A/B/C (%)	Training time per image [s]	Test time per image [s]	IHDR tree memory consumption [Mbytes]
1. Landmark detection - Notch	12	9.31	0/0/3.45	0.0850	0.0323	39.9
2. Object identification and orientation recognition	48	0	N/A	0.0133	0.021225	3.09

to 10.1%, and the false accept rate increased to 30%. The results suggest that both textural information and intensity in this particular case contain significant discriminating features between defective and non-defective regions.

5.3.2 Case Study 2—Landmark Detection. The goal of this experiment was to test our approach for the detection of landmarks used for shape recognition and position or orientation information. The chosen landmark was a notch [see Fig. 10(*b*)]. It was used to identify position and rotational attitude of the part. This version of our architecture is presented in Fig. 11(*b*). The system was trained to locate the notch in this circular part by feeding it narrow rotation ranges (5 deg) when the notch was present in the frame and wide rotation ranges (20 deg) when the notch was out of the frame.

Preparation of training images was done automatically as follows: Each training part was located on a rotary motion stage with the notch at position "0 deg" (3 o'clock). The system rotated the part, captured an image, and labeled the captured sub-images according to orientation. Note that the motion programming and capturing is simple, and can be done by a line worker using the generic motion and acquisition system. Training does not involve dedicated software developer's programming for this specific application or for a specific landmark.

According to the orientation of the notch in the sub-image, and the sub-image position within the captured image, the system determined the part position and orientation of a part. Although in this specific case one can use simple image processing techniques [such as minimal-thickness finder in a binarized (black-white) image], our method is general enough for any shape of dimensional landmarks in any part. In future implementations, finer hierarchies of orientation recognition may be applied once the notch is in the field-of-view.

Class-specific error definitions were used in our data analysis:

- Error A—classification of an image that includes a notch as an image that excludes a notch.
- Error B—classification of an image that excludes a notch as an image that includes a notch.
- Error C—classification of an image that includes a notch as an image that includes a notch but at a wrong orientation.
- Error D—classification of an image that excludes a notch as an image that excludes a notch but at a wrong orientation.

Since the part is approximately axisymmetric, error D is expected to be large. Therefore, it was impractical to use the appearance of the notch-free images for direct orientation detection. Instead, the part is rotated rapidly until the notch is detected in the image. Then, it is rotated in small increments until the notch is detected at the center.

Table 4 (upper row) summarizes the performance results of the notch detection experiment. Since the number of classes is 12, the maximum recognition time until a notch is found is eight times the test time per image (seven classes without a notch and one with a notch). This sums to 0.258 s per part. Additional rotating is due to moving the part to its final orientation. A part that is not axisymmetric, will require only two alignment steps, which result in a reduction of the overall process time.

5.3.3 Case Study 3—Object Identification and Orientation Recognition. Object identification lies between the position-invariant texture analysis and the position-relevant landmark detection (Sec. 1.1). The system automatically creates a well-framed image, where the object is positioned and oriented canonically, by intensity-based attention selection. The architecture that handles this case is presented in Fig. 11(*c*). Examples of the resultant images are presented in Fig. 10(*c*). There are four different part classes that have six possible faces to face the camera. Each face has two possible orientations with a relative angle of 180 deg. Four different images of each case were taken, at different locations in the field-of-view (FOV) of the camera, in order to train

for the variations in illumination across the FOV.

Collection of training images was done automatically by placing the training parts on the $x-y$ stage, one at a time. Each image was captured and attention to the part was given automatically, as described earlier. The program labeled each training image according to the part type and according to one of 12 discrete orientations.

The results show perfect recognition of the parts and their orientations. This is due to the fact that the variations between parts of the same type are small and consequently, a small number of training samples can guarantee perfect classification.

6 Conclusions

We propose a developmental approach for adaptive general-purpose machine vision inspection. The architecture outlines an unsupervised sensory mapping followed by a supervised cognitive mapping for feature derivation and classification. The results are fed to the motor mapping, which in turn generates attention control to the sensory mapping and gives commands for further actions, such as part sorting. A programmed-in classifier may be integrated to operate before the main route has been trained sufficiently. The essence of developmental vision is that attention and recognition are autonomously generated (developed) from an acquired training set. By utilizing this approach, a human developer does not need to get involved in the system ramp-up since re-programming of the system is not needed in order to match it with a new task, a new part, or changing environmental conditions. These attributes result in cost reduction and shortened ramp-up time to renewed operation. Therefore, our system is particularly suitable for a reconfigurable manufacturing system that is characterized by frequent and major changes.

We present guidelines for training in three cases that are characterized by different levels of available prior knowledge about the inspected part. These cases embody measurable and non-measurable variations. We propose estimates of the required number of training images, which prove feasibility of system operation in real-time conditions. The off-line procedures needed for this method (i.e., collection of training images and training) replace algorithm modification via re-programming that is required by conventional methods. We conclude that the proposed training method has the advantage of simplicity and convenience compared to re-programming. It can be performed on the production line. The process of collecting training images, however, may be either quick and automatic, or manually involving, depending on the application. Yet, image collection and training require a lower level of expertise than does re-programming.

We developed complexity formulas based on the number of training images required for the proposed training strategy. We structured them around time and memory resource bounds, imposed by the inspection task requirements and the available computer memory. The formulas show that our approach enjoys low time and space complexities and suggest that it is potentially compatible with real-time inspection needs.

The experiments demonstrated that the approach is both general (addresses various tasks) and adaptive (addresses major variations in a particular task). We provided three case studies, which differ in their dependence on position information. With almost identical algorithm implementations, the system performed fast with low error rate in all the cases. We also demonstrated the ability to automatically create different degrees of invariance to position. This can be generalized to invariance to rotation, scaling, illumination, and also to more complex properties.

Acknowledgments

Nomenclature

FN = false negative
FOV = field-of-view
FP = false positive
FSC = false subclass classification
FC = false super-class classification
HDR = hierarchical discriminant regression
ICA = independent component analysis
IHDR = incremental hierarchical discriminant regression
LDA = linear discriminant analysis
LNMF = local non-negative matrix factorization
PCA = principal component analysis
RMS = reconfigurable manufacturing system

References

[1] Al-kindi, G. A., Baul, R. M., and Gill, K. F., 1992, "An Application of Machine Vision in the Automated Inspection of Engineering Surfaces," Int. J. Prod. Res., **30**, pp. 241–253.

[2] Rao, A. R., and Jain, R., 1990, "A Classification Scheme for Visual Defects Arising in Semiconductor Wafer Inspection," J. Cryst. Growth, **103**, pp. 398–406.

[3] Lampinen J., and Smolander S., 1994, "Wood Defect Recognition with Self-Organizing Feature Selection," *Proc. SPIE Conf. on Intelligent Robots and Computer Vision XIII: Algorithms and Computer Vision*, D. P. Casasent, and E. L. Hall, eds., Vol. 2353, pp. 385–395.

[4] Niskanen M., Silvén O., and Kauppinen, H., 2001, "Experiments with SOM Based Inspection of Wood," *Proc. Intl. Conf. on Quality Control by Artificial Vision*, Vol. 2, pp. 311–316.

[5] Silva, J. A., Pais, C. P., Freitas, J. C., Carvalho, F. D., and Rodrigues, F. C., 1992, "Detection and Automatic Classification of Defects in Ceramic Products," *Proc. SPIE Intl. Conf. on Manufacturing Automation*, W. Zhiling et al., eds., Vol. 1713, pp. 22–28.

[6] Sari-Sarraf, H., and Goddard, J. S., Jr., 1999, "Vision System for On-Loom Fabric Inspection," IEEE Trans. Ind. Appl., **35**, pp. 1252–1259.

[7] Koren, Y., Jovane, F., Heisel, U., Pritschow, G., Ulsoy, G., and Van Brussel, H., 1999, "Reconfigurable Manufacturing Systems," CIRP Ann., **48**, pp. 6–12.

[8] Nazir, T., and O'Regan, J. K., 1990, "Some Results on Translation Invariance in the Human Visual System," Spatial Vis., **5**, pp. 81–100.

[9] Kolers, P. A., Duchnicky, R. L., and Sundstroem, G., 1985, "Size in Visual Processing of Faces and Words," J. Exp. Psychol. Hum. Percept. Perform., **11**, pp. 726–751.

[10] Thompson, P., 1980, "Margaret Thatcher: A New Illusion," Perception, **9**, pp. 483–484.

[11] Harris, C., and Stephens, M., 1988, "A Combined Corner and Edge Detector," *Proc. of the Fourth Alvey Vision Conference*, pp. 147–151.

[12] Hough, P. V. C., 1992, "Method and Means of Recognizing Complex Patterns," *US Patent* 3,069,654.

[13] Long, H., Tan, C. W., and Leow, W. K., 2001, "Invariant and Perceptually Consistent Texture Mapping for Content-Based Image Retrieval," *Proc. 2001 Int. Conf. on Image Processing*, Vol. 2, pp. 117–120.

[14] Teuner, A., Pichler, O., Sandos Conde, J. E., and Hosticka, B. J., 1997, "Orientation- and Scale-Invariant Recognition of Textures in Multi-Object Scenes," *Proc. 1997 Int. Conf. on Image Processing*, Vol. 3, pp. 174–177.

[15] Picard, R. W., and Kabir, T., 1993, "Finding Similar Patterns in Large Image Databases," *1993 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Vol. 5, pp. 161–164.

[16] Reddy, B. S., and Chatterji, B. N., 1996, "An FFT-Based Technique for Translation, Rotation, and Scale-Invariant Image Registration," IEEE Trans. Image Process., **5**, pp. 1266–1271.

[17] Charalampidis, D., and Kasparis, T., 2002, "Rotation Invariant Roughness Features for Texture Classification," *2002 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Vol. 4, pp. 3672–3675.

[18] Greenspan, H., Belongie, S., Goodman, R., Perona, P., Rakshit, S., and Anderson, C. H., 1994, "Overcomplete Steerable Pyramid Filters and Rotation Invariance," *Proc. 1994 Computer Vision and Pattern Recognition*, pp. 222–228.

[19] Götze, N., Drüe, S., and Hartmann, G., 2000, "Invariant Object Recognition with Discriminant Features Based on Local Fast-Fourier Mellin Transform," *Proc. 15th Int. Conf. on Pattern Recognition*, Vol. 1, pp. 948–951.

[20] Neubauer, C., 1991, "Fast Detection and Classification of Defects on Treated Metal Surfaces Using a Backpropagation Neural Network," *IEEE Int. Joint Conf. Neural Networks*, Vol. 2, pp. 1148–1153.

[21] Caleb, P., and Steuer, M., 2000, "Classification of Surface Defects on Hot Rolled Steel Using Adaptive Learning Methods," *Proc. IEEE Fourth Int. Conf. on Knowledge-Based Intelligent Engineering Systems & Allied Technologies*, Vol. 1, pp. 103–108.

[22] Wu, X., Wang, J., Flitman, A., and Thomson, P., 1999, "Neural and Machine Learning to the Surface Defect Investigation in Sheet Metal Forming," *Proc. of the Neural Information Processing*, Vol. 3, pp. 1088–1093.

[23] Sinha, S. K., Karray, F., and Fieguth, P. W., 1999, "Underground Pipe Cracks Classification Using Image Analysis and Neuro-Fuzzy Algorithm," *Proc. of IEEE Int. Symp. on Intelligent Control/ Intelligent Systems and Semiotics*, pp. 399–404.

[24] Weng, J., Ahuja, N., and Huang, T. S., 1997, "Learning Recognition and Segmentation Using the Cresceptron," Int. J. Comput. Vis., **25**, pp. 109–143.

[25] Kirby, M., and Sirovich, L., 1990, "Application of the Karhunen-Loeve Procedure for the Characterization of Human Faces," IEEE Trans. Pattern Anal. Mach. Intell., **12**, pp. 103–108.

[26] Etemad, K., and Chellappa, R., 1994, "Discriminant Analysis for Recognition of Human Face Images," J. Opt. Soc. Am. A, **14**, pp. 1724–1733.

[27] Comon, P., 1994, "Independent Component Analysis—a New Concept?," Signal Process., **36**, pp. 287–314.

[28] Tenenbaum, J. B., de Silva, V., and Langford, J. C., 2000, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," Science, **290**, pp. 2319–2323

[29] Feng, T., Li, S. Z., Shum, H.-Y., and Zhang, H. J., 2002, "Local Non-Negative Matrix Factorization as a Visual Representation," *Proc. IEEE Int. Conf. on Development and Learning*, pp. 178–183.

[30] Hwang, W. S., and Weng, J., 2000, "Hierarchical Discriminant Regression," IEEE Trans. Pattern Anal. Mach. Intell., **22**, pp. 1277–1293.

[31] Weng, J., and Hwang, W. S., 2000, "An Incremental Learning Algorithm with Automatically Derived Discriminating Features," *Proc. Asian Conf. on Computer Vision*, pp. 426–431.

[32] Zeng, S., and Weng, J., 2003, "Online-learning and Attention-based Approach to Obstacle Avoidance Behavior," Technical Report No. MSU-CSE-03-26, Michigan State University, Lansing, MI.

[33] Itti, L., Koch, C., and Niebur, E., 1998, "A Model of Saliency-Based Visual Attention for Rapid Scene Analysis," IEEE Trans. Pattern Anal. Mach. Intell., **20**, pp. 1254–1259.

[34] Zhang, N., Weng, J., and Zhang, Z., 2002, "A Developing Sensory Mapping for Robots," *Proc. Second Int. Conf. on Development and Learning*, pp. 13–20.

[35] Weng, J., Abramovich, G., and Dutta, D., 2003, "Developmental Vision: Architecture, Invariance, Attention, and Complexity," Technical Report No. MSU-CSE-03-19, Michigan State University, Lansing, MI.

[36] Duda, R. O., Hart, P. E., and Stork, D. G., 2001, *Pattern Classification*, J. Wiley, New York, 483 pp.