LEARNING STRUCTURED VISUAL DETECTORS FROM USER INPUT AT MULTIPLE LEVELS

ALEJANDRO JAIMES
Department of Electrical Engineering, Columbia University, 500 West 120th street MC 4712
New York, NY 10027, USA
E-mail: ajaimes@ee.columbia.edu
Web: www.ee.columbia.edu/~ajaimes

SHIH-FU CHANG
Department of Electrical Engineering, Columbia University
E-mail: sfchang@ee.columbia.edu
Web: www.ee.columbia.edu/~sfchang

Revised (March 28, 2001)

In this paper, we propose a new framework for the *dynamic* construction of structured visual object/scene detectors for content-based retrieval. In the *Visual Apprentice*, a user defines visual object/scene models via a multiple-level *definition hierarchy*: a *scene* consists of *objects,* which consist of *object-parts*, which consist of *perceptual-areas*, which consist of *regions*. The user trains the system by providing example images/videos and labeling components according to the *hierarchy* she defines (e.g., image of two people shaking hands contains two faces and a handshake). As the user trains the system, visual features (e.g., color, texture, motion, etc.) are extracted from each example provided, for each node of the hierarchy (defined by the user). Various machine learning algorithms are then applied to the training data, at each node, to learn classifiers. The best classifiers and features are then automatically selected for each node (using cross-validation on the training data). The process yields a *Visual Object/Scene Detector* (e.g., for a handshake), which consists of an hierarchy of classifiers as it was defined by the user. The *Visual Detector* classifies new images/videos by first automatically segmenting them, and applying the classifiers according to the hierarchy: *regions* are classified first, followed by the classification of *perceptual-areas*, *object-parts*, and *objects*. We discuss how the concept of *Recurrent Visual Semantics* can be used to identify domains in which learning techniques such as the one presented can be applied. We then present experimental results using several hierarchies for classifying images and video shots (e.g., Baseball video, images that contain handshakes, skies, etc.). These results, which show good performance, demonstrate the feasibility, and usefulness of dynamic approaches for constructing structured visual object/scene detectors from user input at multiple levels.

1. INTRODUCTION

In the last few years, there has been a tremendous growth in the availability of multimedia data. This is partly due to better and less expensive technologies to facilitate digital content creation (e.g., digital cameras), acquisition (e.g., scanners, recording devices), and access (e.g., the world wide web). Therefore, one of the most important technical challenges is the development of techniques to access the resulting digital (image/video) collections. Traditional text-based approaches to access images and video, although very useful (and necessary in many applications), cannot be used to fully describe all levels of visual information.[30] Consequently, a large amount of research in Content Based Retrieval (CBR) [6,13,67,62,78,11,9] has focused on indexing visual information (images/video) using syntax (visual features: color, texture, etc.) at different levels.[24] While syntactic information is useful, users are mostly interested in the meaning of the information (not the form). In particular, users are interested in semantics (objects,

scenes, events, etc.).[33] In this regard, textual descriptions of visual information can be more useful, in many applications. Manually annotating images or videos, however, is a costly endeavor, specially given the rate at which digital content is produced.

Consequently, most recent efforts in CBR attempt to automatically assign semantic labels to images/videos. Proposed techniques range from classification mechanisms, to approaches that structure and describe data. Automatic classification of images, for example, can be done at the object (e.g., the image contains a horse, or a naked body [18]), or scene level (indoor and outdoor,[74] mountain scene,[77] etc.). Classification of video, can also be performed at the scene/object levels. Several other approaches attempt to automatically structure data (image collections or videos). Additionally, there are major efforts to create structures and standards for the description of multimedia content. MPEG-7, [7,48] for example, aims at standardizing a framework for describing audio-visual content.

One active research area is classification. In particular, it is desirable to construct systems that can automatically examine visual content, and label it based on the semantic information it contains. Many of the current systems perform this task by classifying images/video and assigning them semantic labels. Typically such classifiers are built (by experts) to perform specific tasks (e.g., indoor vs. outdoor image classification). The classifiers index image/video data, and users can then utilize the corresponding labels for searching and browsing. Different users, however, search for information in different ways, and their search criterion may change over time. Therefore, many of the current automatic classification approaches suffer from two disadvantages: (1) they do not accommodate *subjectivity* (i.e., the expert decides which classifiers to construct), (2) they do not allow the construction of structured models from user input at multiple-levels.

Manual construction of image/video classifiers can produce systems that are accurate and work well in specific domains. If the number of objects/scenes to classify is large, however, such approach becomes impractical. Furthermore, class definitions depend on the experts that build the systems, and any modification to the class definitions must be performed by the experts themselves. In addition, users may have interests that are different from those of the experts building such systems. The definition of a "handshake image class," for example, may vary among different individuals: for one user the class may include images that show the hands of two individuals, but nothing else. For another user, it may include only images in which people are pictured, from a certain distance, shaking hands. While specialized algorithms can be very useful in some domains (e.g., face recognition), we argue that successful content-based retrieval systems should be *dynamic*, to allow construction of classifiers that cater to different users' needs. Algorithms should be as general as possible so that they can be applied in several domains, and they must exhibit enough flexibility to allow users to determine the classes in which they are interested. In addition, they should allow the definition of complex multiple-level models that can accurately represent (and capture) real world structures. One way to enhance the capability of such systems is to construct flexible frameworks that use machine learning techniques. [46,5,15,1]

In this paper, we present a new approach to content-based retrieval. In the *Visual Apprentice (VA)*[27,28,29] framework users are able to build *Visual Object/Scene Detectors* (classifiers) by defining a model for their classes of interest and providing training examples. The model in this framework is defined by the user based on an *hierarchy* that

contains several levels (*scene, object, object-part, perceptual-area, region*). More specifically, the user defines a model (*definition hierarchy*) for an object or scene, and labels regions (from automatic segmentation) of image/video examples, according to the model (*hierarchy*). The system uses several machine learning algorithms to learn classifiers corresponding to nodes in the hierarchy. The resulting classifiers are applied to new images/videos, and a decision is made (about the class of the image/video) based on the new image/video's segmentation and results of classification and grouping at multiple levels (according to the hierarchy and training provided by the user). In comparison to other work in CBR, one of the important advantages of our framework is the flexibility of defining object/scene hierarchies and detailed user input at multiple levels. The approach allows users to specify multiple level composition models, which are absent in most existing approaches to CBR.

1.1 Related Work

Research in the area of CBR has grown tremendously in the last few years (for recent reviews, and references see [6,67,62,78,13,11,9]). Many of the systems (e.g., QBIC,[53] VisualSEEk,[70] VideoQ,[8] Virage,[3] Spire,[4] etc.) have used query-by-example ("show me images like this one"), and query-by-sketch ("show me images that look like this sketch"). Some systems have enhanced capabilities for query formulation (e.g., in Spire, users can perform queries using examples from different images; in Semantic Visual Templates[10] the system tries to help the user formulate queries, using relevance feedback). Others have focused on classification, using visual features only, textual features only (e.g., WebSEEk[69]), or combinations of different types of features (e.g., textual and visual in news images;[55] visual and audio in [50]). Distinctions between different approaches can be made in many different ways. In [24], for example, distinctions are made based on the level of description used,[30] interface, type of features, etc.

Approaches that perform classification of visual content based on visual features, like the *VA*, can be divided into those that perform automatic classification at the scene level (indoor vs. outdoor,[74,55] city vs. landscape[77]), and at the object level (faces,[19] naked people, and horses[18]).

Scene level classifiers determine the class of the input image as a whole.[77,74,55] In many of the approaches the image is divided into blocks and the final classification decision is based on a global measure over the blocks. They differ from the *VA* since images are classified based on their global features- not on the structure of local components (i.e., a user defined model of scene structure). In addition, the algorithms proposed in many of those systems are specific to the classifiers being built. For example, in the work of,[77] the features that the algorithms use were chosen by the authors, based on the qualitative features of the different classes being considered (indoor, outdoor, city, landscape, sunset, forest, and mountain).

Other related approaches perform scene level classification based on regions obtained from automatic segmentation.[68,40] The configuration of regions in different scene classes is used during classification. A typical beach scene, for example, contains blue regions at the top (sky), and yellow regions at the bottom (sand). This type of information is used in a training stage, and the configuration of regions in new images is used to determine the images' class. The structure, however, is limited to the global configuration regions in the images, and structured object (or scene) models are not used.

A related approach for object classification uses body-plans[18] in the construction of object models. Specialized filters, for detection of naked people and horses, are used first to select relevant regions in the image. Search for groups that match the body-plan is then performed over those regions. Although this approach allows the construction of multiple-level composition models (like the *VA*), it system differs from the *VA* because it uses specialized algorithms (e.g., filters), and object models built by experts. Likewise, the approach in [19] utilizes a specialized face detection algorithm.[61]

In terms of adaptive systems, most of the related work has been done in Computer Vision.[23,16,20] The main difference between the *VA* approach and previous work in Computer Vision is the role the user plays in defining objects, and the lack of constraints imposed by the *VA* system (e.g., no constraints on lighting conditions, etc.). Other differences range from the representation of the data (e.g., features used), to the learning algorithms, application domain, and operational requirements (e.g., speed, computational complexity). A discussion in [24] outlines differences between CBR and object recognition.

The FourEyes system,[45] learns from labels assigned by a user. User input, however, consists of labeling of regions (not definition of models based on multiple levels like in the *VA*). Although multiple feature models (for feature extraction) are incorporated in that system, different learning algorithms are not used.

Another approach to detecting events on scenes in specific domains is to explore the unique structures and knowledge in the domain. A system developed in [80] includes multiple models (for handling color variations within the same type of sport game- e.g., different colors of sand in tennis) and uses manually constructed region-level rules (for exploring the scene structure). High accuracy was reported in detecting batting scenes in baseball, and tennis serving scenes. This work differs from the *VA* framework in several aspects. In particular, that approach uses domain knowledge programmed by an expert (specific rules for baseball/tennis). In addition, it includes an initial filtering stage based on a global color measure. In other words, the video scene is first analyzed at the global level using color histograms, and then detailed scene analysis is performed. The detailed scene analysis differs in the two approaches (use of rules constructed by experts[80] vs. no expert input in the construction of classifiers in the *VA*). The initial filtering, however, could complement the *VA* framework (e.g., like in [80], a filtering stage could be used to select different detectors, to deal with variations across different games, as outlined in section 4.1.1).

Alternative models that represent objects and scenes in terms of their parts have also been proposed in the CBR community.[17,71,48] The definition of Composite Visual Objects[17], for example, is similar to the *definition hierarchy* of the *VA* framework, with the difference that classifiers in the *Visual Apprentice* are learned automatically. It is also useful to note the similarity between the definition hierarchy and structures used in MPEG-7.

1.2 Outline

The paper is organized as follows. In section 2 we briefly discuss the application of machine learning in CBR. In section 3 we discuss the *Visual Apprentice* framework. In particular we discuss user input, feature extraction and learning, and classification. Then we present experimental results, a general discussion of important issues within the framework, and possible extensions. We conclude in section 6.

2. USING LEARNING IN CBR

As discussed earlier, different users may have different interests, and those interests (for a single user) may vary over time.[57] Using this premise, it is preferable to construct systems that can adapt to users' interests. One possibility is to build systems that can adapt by learning from users. An important issue, therefore, in the application of learning techniques in CBR, is deciding where learning techniques are suitable. The concept of *Recurrent Visual Semantics* (*RVS*)[29,24] is helpful in identifying domains in which to apply learning techniques in the context of CBR.

RVS is defined as the repetitive appearance of elements (e.g., objects, scenes, or shots) that are *visually similar* and have a common level of meaning within a specific context. Examples of domains in which *Recurrent Visual Semantics* can be easily identified include news, consumer photography,[26] and sports. In professional Baseball television broadcast, for example, repetition occurs at various levels: *objects* (e.g., players), *scenes* (e.g., a batting scene), *shots* (e.g., the camera motion after a homerun occurs), and *shot sequence structure* (e.g., a homerun shot sequence often includes a batting scene, a scene of the player running, etc.).

The existence of *RVS* motivates the approach of using learning techniques in content-based retrieval. Using this concept, it is possible to identify domains in which learning techniques can be used to build automatic classifiers (for objects or scenes). The existence of repetition facilitates training, and the domain constrains the future data inputs to the classifiers learned. Once a domain is selected, identification of its repetitive (but semantically meaningful) elements increases the possibilities of successfully applying machine learning techniques in the specific domain. At the same time, application of learning within the domain (e.g., baseball video only, versus all types of video) decreases the possibility of errors.

In section 4 we discuss how this concept was used to select the domains for our experiments.

3. THE VISUAL APPRENTICE

3.1 OVERVIEW

The *Visual Apprentice (VA)* framework consists of three stages: *(1) user input, (2) feature extraction and learning,* and *(3) classification.* In the first stage, the user explicitly defines object/scene models according to her interests, and labels training examples (images or videos). In particular, each example image/video is segmented automatically by the system, and the results of the segmentation are manually labeled by the user according to an *hierarchy* defined by the user. In the second stage, the system extracts features (e.g., color, texture, motion, etc.) from each image/video example provided by the user. Then it learns classifiers based on those examples producing an hierarchy of classifiers (a *Visual Object/Scene Detector- VOD*). In the third stage, the classifiers (the *VOD*) is applied to unseen images/videos. The *Visual Object/Scene Detector* performs automatic classification by first automatically segmenting the image/video, and then combining classifiers and grouping at different levels.

The Definition Hierarchy

Studies in cognition and human vision have shown that during visual recognition, humans perform grouping of features at different levels.[41,2] The highest level of grouping is semantic: areas that belong to an *object* are grouped together. An *object*, however, can be separated into *object-parts,* which consist of *perceptual-areas*: areas that we perceive categorically. Categorical perception refers to the qualitative difference of elements across different categories. Colors, for example, are often "grouped" [73,22]- we say the shir is green, although it may have different shades of green. These different levels of grouping motivate the model-based approach to the construction of *Visual Object Detectors (VOD)*[a] in the *Visual Apprentice*. In this framework, a *VOD* is defined as a collection of classifiers organized in a *definition hierarchy*[b] consisting of the following levels (Fig. 1): (1) *region*; (2) *perceptual*; (3) *object-part*; (4) *object* and (5) *scene*.

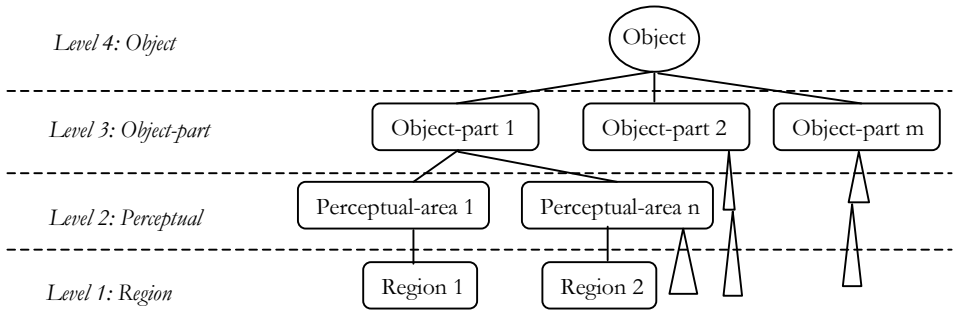

Fig. 1. Definition Hierarchy. Note that a scene (not shown in the figure) is a collection of objects and corresponds to the highest level.

More specifically, a *definition hierarchy* is defined in terms of the following elements:

(5) *Scene:* structured[c] set of objects.
(4) *Object:* structured set of adjoining object-parts.
(3) *Object-part:* structured set of perceptual-areas.
(2) *Perceptual-area:* set of regions that are contiguous to each other and homogeneous within the set.
(1) *Region:* set of connected[d] pixels

According to this definition, every node *e* in our *definition-hierarchy* has a conceptual interpretation (e.g., "object"), and represents a set of connected pixels in an image/video. Nodes are image areas and arcs indicate parent-child relationships (from top to bottom)-

[a] The *detectors* we describe refer to objects and scenes. We use the name VOD, however, for simplicity and to emphasize the local structure of the classifiers.
[b] We have chosen only five levels for the hierarchy because they provide an intuitive representation that is useful in practice.
[c] The word *structured* is used to emphasize the importance of spatial relationships between elements in the particular set.
[d] Regions, which are at the lowest level of the hierarchy, constitute the basic units in the framework and can be extracted using any segmentation algorithm based on low-level features such as color, texture, or motion.

a node is composed of all of its children. For example, in Fig. 1 object-part1 is an area composed of n perceptual areas, each of which is composed of a number of regions.

In addition, the following restrictions are placed on the construction of *valid hierarchies* (please refer to Fig. 1, where each node represents a set): (1) a set of level *i* (i≠5) is a subset of only one set of level *i+1 (e.g.,* an *object-part* can only belong to one *object*; a node in the hierarchy can only have one parent); (2) a set of level *i* cannot be a subset of a set of level *i-1*, unless the two sets are equal (e.g. an *object* cannot be part of a *perceptual-area*; a face can be equal to a single perceptual area); (3) sets at the same level are disjoint (i.e., intersection of two sets of the same level is empty; two *object-parts* cannot intersect); (4) *regions* do not contain subsets (i.e. *regions* are the basic units and cannot be separated); (5) No. sets at level i <= No. sets at level i-1; (6) all sets are finite and can contain one or more elements; (7) every set is equal to the union of its children.

Fig. 2 shows a batting scene as defined by a user. Note that every node in an hierarchy has a conceptual meaning (e.g., pitcher), corresponds to an image area in a training example (e.g., a set of connected pixels in each image), and, as will be shown later, corresponds to a classifier (e.g., pitcher *object-part* classifier). The user, in this case, has decided to model the scene using only four levels (*region, perceptual-area, object-part*, and *object*).

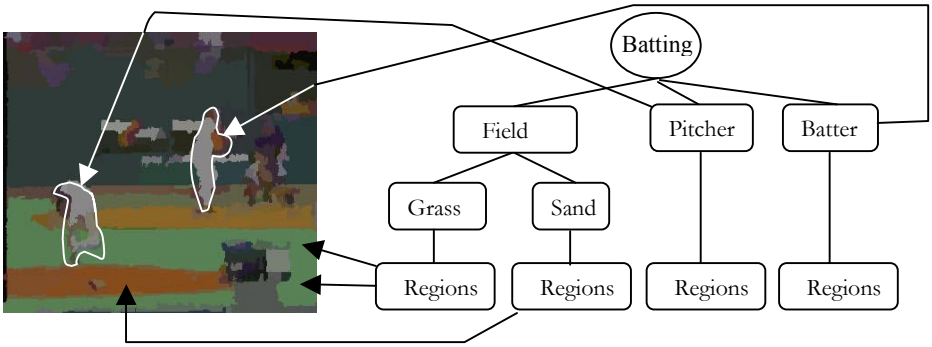

Fig. 2. Automatically segmented Baseball image. This example shows how a scene can be modeled using the *hierarchy*. The white outlines were drawn manually to illustrate how the regions map to the hierarchy. Note that the user decided to model the scene using only four levels.

After the user defines an hierarchy and provides the training examples, features are extracted and classifiers are learned (stage 2). Classification (stage 3) occurs at the levels of Fig. 1: *regions* are classified first and combined to obtain *perceptual-areas,* which are used by *object-part* classifiers. *Object-parts*, in turn, are combined and the results are used by *object* classifiers, etc.

In the sections that follow we discuss, in detail, each of the three stages of the *VA* (user input, feature extraction and learning, and classification).

3.2 USER INPUT

Different users have different interests. In order to accommodate this subjectivity we allow users to build different models (i.e., *definition hierarchies*) based on their

individual preferences. The way in which *VODs* are constructed, therefore, is subjective and may vary between users (or for a single user over time). The main goal of this stage is to let the user construct a *Visual Object Detector*, without any low-level knowledge about features, or learning algorithms[e].

During training, the user performs the following tasks: (1) creation of a *definition hierarchy* by defining the labels to be used for each node; (2) labeling of *areas* (e.g., regions, perceptual-areas, etc.) in each training image/video according to the *hierarchy*.

Using the interface, the user defines the hierarchy by creating labels for nodes and expressing the connections between them. The label "batter region of batter *object-part*," (Fig. 2) for example, clearly defines the connections between the batter *region*, the batter *object-part*, and the batting *object*. Using a simple user interface (Fig. 3), the user can set the corresponding labels (e.g., the "*object-part*" field would say "batter", the "scene" field would say batting, etc.). Once the labels are created, during training, the user labels *regions, perceptual-areas, object-parts*, and *objects,* in *each* training image/video. In the current implementation an image/video example corresponding to a particular hierarchy must contain all of the nodes defined in the hierarchy (e.g., all batting scene examples must contain a field, a pitcher, a batter, etc.).

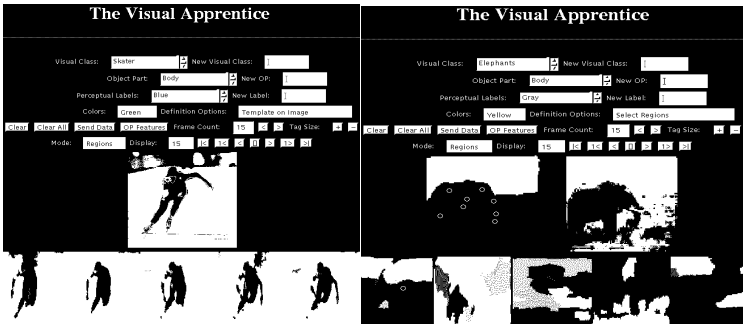

Fig. 3. *Visual Apprentice* Graphical User Interface.

Labeling of image/video examples according to the hierarchy can be done in several ways: (1) by clicking on regions obtained from automatic segmentation, (2) by outlining areas in the segmented/original images/videos. Usually, labeling is almost exclusively done on the segmented images directly. Furthermore, in most cases it is only necessary to label individual regions (without outlining areas), because the interface of the *VA* facilitates training by automatically grouping regions that are connected and have the same label. The groups generated are assigned the label of the parent node of the regions used in the grouping. For example, in Fig. 2, the user labels all of the pitcher regions (from automatic segmentation) with the name "pitcher region". Then the system automatically groups all contiguous "pitcher regions" (those that are connected) and labels that group "pitcher *object-part*" (since the parent of the "pitcher regions" label is "pitcher *object-part*"). In some cases, however, the user may wish to manually outline

[e] In section 5 we discuss possibilities of additional user input (e.g., decisions on learning algorithms to use, etc.).

objects, *object-parts* or *perceptual areas* (note manual outlines in white in Fig. 2) and bypass the automatic grouping algorithm. The difference between using the automatic grouping provided by the system and manually outlining components is that manual outlining eliminates segmentation errors that would otherwise be incorporated. Again in Fig. 2, note that in the segmented image a pitcher region contains some pixels that belong to the background. Manually outlining the pitcher eliminates that error, since the user drawn outline excludes those background pixels in the "pitcher *object-part*" example.

User input for video is similar since only the first frame, in each example video shot, must be labeled- in the segmentation algorithm used[79], regions are automatically segmented and tracked in each video frame. On that first frame, the user identifies regions that correspond to each node in her *definition hierarchy*: all of the sand *regions*, all of the sand *perceptual-areas*, *object-parts*, etc. The labeled region is tracked by the system in subsequent frames. For each region, then, it is possible to extract motion-related features (discussed below).

As a result of user interaction, we obtain the following sets for a defined class *j* (e.g., batting scene of Fig. 2):

• Conceptual *definition hierarchy*: H_j.

• Example Element Set: $EES_j = \{\{(e_{11}, l_{11}), (e_{12}, l_{12}), ..., (e_{1n}, l_{1n})\}, ..., \{(e_{k1}, l_{k1}), (e_{k2}, l_{k2}), ..., (e_{kp}, l_{kp})\}, ..., \{(e_{m1}, l_{m1}), ..., (e_{mq}, l_{mq})\}$ where in each tuple, e_{ki} corresponds to the i^{th} element (i.e., an area of a training image) of level k and l_{ki} is a label of level k associated with the element (e.g., (op_{31}, l_{31}) = (pitcher *object-part*, pitcher label)). Label level distinctions emphasize that labels must be different at different levels of the *hierarchy*. Regions in the example images/videos that are not labeled by the user are automatically assigned the label "unknown" and included in the set EES_j. This way, using the closed-world assumption [5], those regions can be used as negative examples during training.

Note that the user also has the option of including additional images/videos/regions to be used as negative examples, and that each image/video example for a given hierarchy must contain all the nodes in the hierarchy defined by the user. In other words an image/video example corresponding to a particular hierarchy must contain all of the nodes defined in the hierarchy (e.g., all batting scene examples must contain a field, a pitcher, a batter, etc.). As discussed in section 5, it would be possible to modify this constraint to provide further flexibility.

In the training stage, then, user input consists solely of defining the *definition hierarchy* (by creating the appropriate labels), and labeling example image/video areas according to the hierarchy. The labeling is done by clicking on image regions, or outlining image areas in each image/video example.

3.2.1 FEATURE EXTRACTION AND LEARNING

3.2.2 Feature Extraction

As discussed earlier, an element e_{ki} of our model (node in the *hierarchy*) is a set of connected pixels (i.e., an area of the image). Therefore, user input produces, for each example image/video, a set of image/video areas, labeled according to the *hierarchy* defined by the user. For each element in the Example Element Set, we compute a *feature*

vector, which is an attribute-value tuple representation of the features of the element (e.g., color, shape, etc.). By computing feature vectors for all elements in the set EES_j, we obtain a training set of examples for each class *j* (e.g., batting scene of Fig. 2):

• $TS_j = \{\{(fv_{11}, l_{11}), (fv_{12}, l_{12}), ..., (fv_{1n}, l_{1n})\}, ..., \{(fv_{k1}, l_{k1}), (fv_{k2}, l_{k2}), ..., (fv_{kp}, l_{kp})\}, ..., \{(fv_{m1}, l_{m1}), ..., fv_{mq}, l_{mq})\}$ where fv_{ki} corresponds to the i^{th} feature vector element of level k and l_{ki} is a label of level k associated with the feature vector (e.g., (op_{31}, l_{31}) = (pitcher *object-part* feature vector, pitcher label)). Note that all examples for a particular node in the hierarchy (e.g., pitcher region) will have the same label.

Two types of feature vectors are used in the framework, those that contain *raw features*, and those that contain *spatial relationships* (described below). The raw vectors consist of a *superset* of 43 features. These features can be placed into five different groups.

- *Area and location:* area, bounding box center (x, and y), orientation, major axis length, major axis angle, minor axis length.[64]
- *Color:* average L, U, and V, dominant L, U, and V (LUV quantized to 166 colors [70]).
- *Shape:* perimeter, form factor, roundness, bounding box aspect ratio, compactness, extent.[64]
- *Texture:* mean Maximum Difference, mean Minimum Total Variation (MTV), horizontal, vertical, diagonal, and anti-diagonal Mean Local Directed Standard Deviation (MLDSD), edge direction histogram (see [60,77]).
- *Motion trajectory:* maximum/minimum horizontal and vertical displacement, absolute horizontal/vertical displacement, trajectory length, displacement distance, average motion angle, average horizontal/vertical speed/acceleration.

Feature extraction occurs for all nodes, according to the hierarchy defined by the user. By computing feature vectors for each element, a *training set* is obtained for *every node* of the hierarchy. Recall that during user input (section 3.2), grouping occurs between regions that are connected and have the same label (e.g., in Fig. 2 pitcher regions form a pitcher *object-part*; sand regions are grouped at the sand *perceptual-area* node). For each image example, when the grouping is performed (or a manual outline is used), a new area of the image is used for feature extraction. In other words, the features of the *regions* of the pitcher are used at the pitcher region node, but at the parent node (pitcher *object-part* in this case) a new set of features is computed from the image area that results from merging all connected pitcher regions together. The connected (labeled) pitcher regions, then, serve as a mask that is used to extract new features for the parent node of the region node used in the grouping (again in Fig. 2, pitcher *object-part* for pitcher regions, sand *perceptual-area* for sand regions, and so on).

Elements of the hierarchy that are structured (i.e., *scenes, objects*, and *object-parts* in the definition hierarchy of section 3.1), and have more than one element (i.e., field *object-part* and batting *object* in Fig. 2) are treated differently during feature extraction in the sense that they are characterized in terms of the elements they contain and the spatial relationships between those elements. For example, in Fig. 2, the feature vector for the field *object-part* does not contain the 43 features discussed earlier. Instead, it contains

two elements (grass, sand), and their spatial relationships. Note the difference between the following feature vectors:

Pitcher *region* = {label = pitcher, color = white, texture = coarse, etc.} (i.e., a region and its 43 features from the set described above)

Field *object-part* = {label = field_object_part, grass *perceptual-area* contains sand *perceptual-area*} (e.g., an *object-part* in terms of its perceptual areas and their spatial relationships)

To represent the structural relationships in structured sets that have more than one element, (e.g., between *perceptual-areas* within *object-parts*, or *object-parts* within *objects*, etc.), *Attributed Relational Graphs (ARG)* [56,46,65] are constructed. In an *ARG*, nodes represent elements and arcs between nodes represent relationships between elements. In the *VA*, nodes in the *ARGs* correspond to labeled elements from the *hierarchy* being considered, and arcs represent simple spatial relationships between those elements. In particular, the following relationships are used: above/below; right of/left of; near; far; touching; inside/contains.

It is important to note that using this representation an *ARG* will contain labeled elements only (see Field feature vector above), and their relationships. This is important because in the classification stage matching of graphs that contain unlabeled objects, which is a hard combinatorial problem, is avoided. Additionally, to avoid the difficulties of searching in complex relational representations (e.g., Horn clauses), the *ARG* is converted from its original relational representation to an attribute-value representation:[46] elements in the *ARG* are ordered (according to their label) and a feature vector is generated. With such transformation, existing learning techniques that use feature vector representations can be applied directly (e.g., decision trees, lazy learners, etc.).

The result of the feature extraction stage, then, is a set of feature vectors *for each node* of the corresponding hierarchy. Note that in the set TS_j the positive examples for a particular node are those feature vectors in TS_j that have the label for the corresponding node. The rest of feature vectors in the set TS_j are negative examples, for that node, under the closed-world assumption [5]. In essence, if there are *n* nodes in the hierarchy, there will be *n+1* different labels (including the "unknown" label) in the set TS_j. This means that there will be *n* different classes (one for each node), and therefore *n* different classification problems, each of which contains a set of positive and negative examples. This is important because it emphasizes that the result of the training stage is a set of *different classification problems*, one for each node.

3.2.3 Learning of Classifiers and Feature Selection

A *classifier* is a function that, given an input, assigns it to one of *k* classes. A *learning algorithm* is a function that, given a set of examples and their classes, constructs a classifier [14]. These two definitions are of extreme importance in Machine Learning and in particular in the framework of the *VA*. Using the labeled feature vectors, learning algorithms are applied at *each node* of the hierarchy defined by the user to obtain classifiers. This is done for each node at the five levels defined above: *(1) region, (2) perceptual, (3) object-part, (4) object* and *(5) scene.*

As depicted in Fig. 4, all classifiers in an hierarchy could be constructed independently using a single learning algorithm. For example, it would be possible to choose one of the

most widely used learning algorithms[5,46] (e.g., decision trees, lazy learners,[1] neural networks, etc.) and apply it at each node to obtain the corresponding classifiers. The difficulty with this approach is that no algorithm will outperform (in terms of classification accuracy) all other algorithms in all tasks. In other words, since the *VA* is meant to allow users to define their own classes, it is not possible to choose, a priori, a learning algorithm that will produce classifiers that will always perform better than classifiers produced by any other learning algorithm. Of course, other factors could be considered (discussed further in section 5), including availability of resources (computational, number of training examples, etc.), speed requirements (during training and during classification), and desired accuracy.

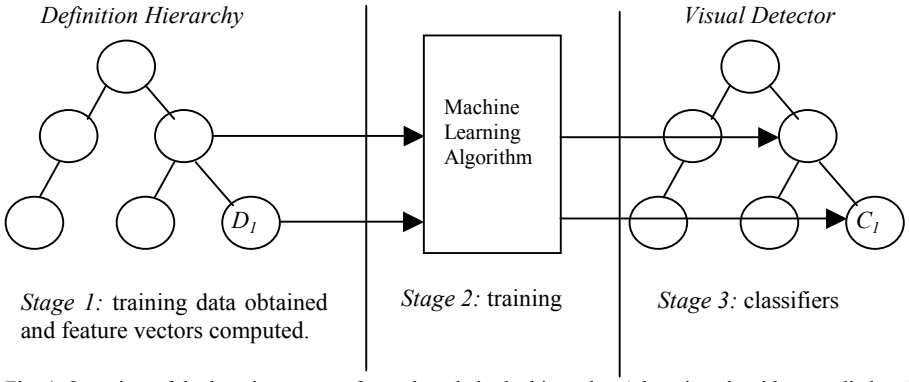

Fig. 4. Overview of the learning process for each node in the hierarchy. A learning algorithm, applied to the training data for each node, produces a classifier for the corresponding node.

In order to allow flexibility, we propose a different approach (discussed in the next section), which consists of applying several *learning algorithms* to the same training data (at each node), to obtain a collection of binary classifiers for each node.

Regardless of the approach chosen to construct classifiers (using one or several learning algorithms), it is well known that selection of features can have a strong impact on classifier performance, even with learning algorithms that incorporate some form of feature selection. The justification and benefits in performance of selecting features in decision trees, for example, is given in [52,39,32]. This is because different features may be better for representing different concepts. For example, "good" features to represent a field might be color and texture, but good features to represent pitcher might be spatial location and aspect ratio (see Fig. 2). Therefore, using the same features for all hierarchies (or for all nodes within a given hierarchy) may not yield the best results.

In many content-based approaches, specifically in interactive ones (query-by-sketch and query by example techniques [8,70,53]), users typically select the features to use. This, however, is often a difficult task. Automatic feature selection, used in the *VA* framework, serves to shield the user from the difficulties inherent in deciding which features are more important for a specific task (i.e., node in a hierarchy, or *VOD*). Given a set of features A (e.g., the *superset* described in section 3.2.1) with cardinality n, we wish to find a set B such that $B \subseteq A$, and where B is a better feature set than A[31]. The criterion for a "better" feature set S can be defined in terms of a criterion function $C(S)$, which gives high values

for better feature sets and lower values for worse feature sets. One possibility is to define the function as (1-P_e), where P_e is the probability of error of a classifier. In such case, the value of the function $C(S)$ depends on the *learning algorithm*, the *training* set, and *test* set used.

Since the goal is to find a set B, such that $B \subseteq A$, *feature subset selection (FSS)* [31] can be characterized as a search problem. The search for a better feature set can be conducted using several heuristics that aim to avoid exhaustively analyzing all possible feature sets. In particular, the search can look for optimal or sub-optimal results, and can be based on a filter or wrapper approach [36]. In the filter approach, the search for best features is independent of the learning algorithm and classifier that will be used. In the wrapper approach, which is used in the *VA*, the criterion function (1- P_e) is dependent on the *learning algorithm* and *data* used. Feature selection, therefore, is performed with respect to a particular algorithm and data set. In particular, a learning algorithm is repeatedly run on a data set using various feature subsets so that each run produces a classifier that uses a different set of features. The performance of the classifiers learned using each feature set is measured (using k-fold cross-validation, described below), and the best feature subset is chosen according to the performance. Once the features are chosen, the learning algorithm is used to construct a classifier using only those features. In the *VA*, best-first forward search[65] (a sub-optimal non-exhaustive technique) is used to find the best features.

The search for a better feature set is performed, using a given learning algorithm, by building different classifiers using different subsets of the original feature set. Once the best feature set is found (for that particular algorithm), a classifier, using that algorithm is constructed. Since we use several algorithms (next section), it is then necessary to compare the different classifiers constructed by the different algorithms, at each node.

3.2.4 Selection and Combination of Classifiers

In the machine-learning community, an important goal is often to compare the performance of different learning algorithms.[14,66] The criterion in those cases is often the performance of the algorithms on standard data sets (e.g., UC Irvine repository[49]), or in particular domains. Instead of trying to find the best algorithms for classifying visual information, the goals in the *VA* framework center on determining the best *classifiers* for specific tasks (e.g., nodes of the *hierarchy*). The goal of the training stage is to obtain the best possible classifier (*not* learning algorithm) for each node. Since different learning algorithms may produce classifiers that perform differently on the same training data, the system simultaneously trains several algorithms (ID3, Naïve-Bayes, IB, MC4)[38] and selects the classifier that produces the best results. Note that, as discussed in the previous section, *FSS* is performed with respect to each algorithm, so when the system compares classifiers (this stage) it is already using the best features found, for each algorithm, in the previous step.

The performance of each classifier is measured using *k-fold cross-validation:* [37,47] the set of training examples is randomly split into k mutually exclusive sets (folds) of approximately equal size. The learning algorithm is trained and tested k times; each time tested on a fold and trained on the data set minus the fold. The *cross-validation* estimate of accuracy is the average of the estimated accuracies from the k folds. In the *VA* accuracy is determined as the overall number of correct classifications, divided by the

number of instances in the data set. The process is repeated for each classifier being considered.

The best classifier is chosen according to its performance estimate given by the cross-validation accuracy: for each node, the classifier with the highest accuracy is selected. The process occurs for every node of the *hierarchy* defined by the user.

An alternative to selecting the best classifier is to combine all or some of the classifiers resulting from the cross validation process. In [28] other ways in which classifiers can interact in the *VA* framework were presented (also see [54] for a different combination strategy in a different framework).

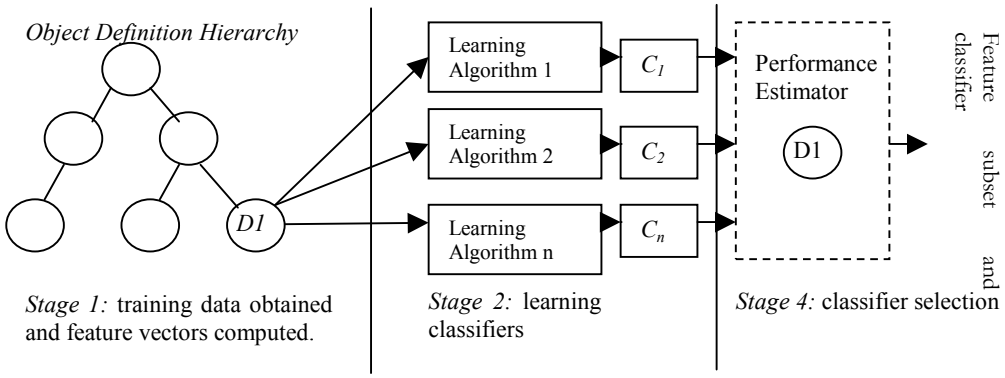

Fig. 5. Overview of the learning process for each node in the hierarchy. Note that each classifier is produced with the best feature set for the particular learning algorithm being applied.

3.3 CLASSIFICATION

When a *VOD* is applied to a new image/video, the first step is automatic segmentation of the image/video. Classification then follows the bottom up order of the *hierarchy* defined by the user (Fig. 3). First, individual *regions* are classified (in a selection process similar to [73]) and, then, *perceptual-areas* formed (i.e., *regions* are classified *perceptually* and *groups* are found). Those *groups* are then combined to form prospective *object-parts*, which form *objects* that form *scenes*. Classification, however, depends on the specific hierarchy defined by the user. To detect a pitcher *object-part* (Fig. 2), for example, the corresponding *VOD* would find pitcher regions first, and then try to find groups of pitcher regions that may correspond to the pitcher *object-part*. The process would be similar for grass and sand *perceptual areas*- regions are selected and groups of regions are used by the parent classifier (of the corresponding region classifier). Note that this is similar to the grouping performed by the system in the training phase (section 3.2). During training, *regions* are labeled by the user, so the system knows exactly which regions (those labeled) must be taken as a group at the parent node. In the classification stage, the labels are assigned by a region classifier. The classifier of the parent node, therefore, must search the space of possible groups.

In the first step, then, regions are selected by a region classifier. Given a universe U of elements, a function $c_j(i)$ is a classifier for class j that determines membership of i ($i \in U$)

in the set *j*. In binary classifiers, $c_j: U \rightarrow \{0,1\}$ where, $\forall\, i \in U$, $c_j(i) = 1$ if $i \in j$ and $c_j(i) = 0$ if $i \notin j$. In fuzzy-classifiers[35] the function is not binary, but continuous, thus $c_j: U \rightarrow [0,1]$. In this case, *j* is a fuzzy-set since each element in *j* has been assigned a degree of membership in that set (e.g., if $c_j(i) = 0.75$ and $c_j(l) = 0.68$ we say that *i* is a *stronger* member of class *j* than *l)*.

Region classification results in a set of region-membership tuples $R_{op} = \{(r_1, m_1, s_1), (r_2, m_2, s_2), ..., (r_n, m_n, s_n)\}$ where in each tuple (r_i, m_i, s_i), r_i is a *region* that belongs to the current *region* class with degree of membership m_i. The variable s_i is used to differentiate *weak* (s_i=0) and *strong* members of the class (s_i=1). This is useful because weak isolated regions can be discarded.

We apply a grouping function *g* to R_{op}, to obtain a set $PG = \{g_1, g_2, .., g_n\}$ where every element g_i is a *group* of adjoining *regions* (e.g. *group* of pitcher *regions*). The goal then becomes to find the most likely group candidates from the set *PG* (e.g., determine which group of pitcher *regions* is more likely to be a pitcher *object-part*). Each group g_i may contain strong and weak *region* candidates, or just strong candidates. Groups of only weak regions are not considered because it is very unlikely for such groups to be important (e.g., unlikely pitcher *regions* are unlikely to form a pitcher *object-part*).

A search, then, must be performed over the space of possible *groups* of regions from the set *PG* to find the best possible ones. This can be treated as a classical search problem in Artificial Intelligence [20,65], and therefore, we can use heuristic techniques to reduce the search space. In particular, we use an Evolution Algorithm[44], treating each element g_i in the set *PG* as an individual in a population. Individuals evolve from one generation to the next through genetic operations such as mutation (an individual's characteristics are changed) and cross-over (two or more individuals combined to produce a new one). During the evolution process (generation to generation), only "strong" individuals survive- that is, individuals that meet certain fitness criteria.

What follows is a description of our algorithm:

1. Initialize population (P = PG).

2. Evaluate individuals in P using as a fitness function, the classifier of the parent node of the function used to select the regions to form PG. If the maximum number of iterations has been reached, or an element in P satisfies the criterion function, stop. Otherwise, continue to step 3.

3. Select and mutate individuals (e.g., remove a region from a selected group, etc.).

4. Go to step 2.

To summarize, strong *region* candidates are first grouped and then merged with adjoining weak candidates. This eliminates from consideration isolated weak candidate *regions.* The evolution program then considers each *group* of regions. At every generation step, each *group* is mutated, thus generating a new individual. A new individual's fitness in the population is measured by the region candidate's parent node classifier. Note that each individual in the population (a group of regions) corresponds to an area in the image/video being considered. Therefore, features (recall raw feature set of section 3.2.1) are extracted from that area, and the classifier that was learned during training is applied. Examining the example of Fig. 2 again, region classifiers for the following nodes are applied: grass, sand, pitcher, and batter. The grouping using the evolution program is

performed for each of these, and the groups are judged by the corresponding parent classifiers (grass and sand perceptual-areas; pitcher and batter object-parts). The field classifier, then, receives as input a feature vector that contains the grass and sand perceptual areas found (with their spatial relationships, as explained in section 3.2.1). The batting classifier, then, receives as input three elements and their spatial relationships (field, pitcher, and batter).

A final decision is made by the *Visual Object Detector (VOD)*, then, based on the decisions of all of its classifiers. In particular, all elements of the hierarchy must be present for a *VOD* to detect the object. For the batting scene of Fig. 2 to be found, all elements must be found (i.e., pitcher, field and its parts, etc.).

4. EXPERIMENTAL RESULTS

Applying CBR techniques, and in particular those that use learning, in a real world scenario can be a challenging task for many different reasons. Therefore, we will describe some of the issues we encountered applying the VA, and experimental results. In each of the experiments reported in this section (baseball, handshakes, skies) the amount of time required to complete the training stage was less than two hours.

4.1.1 Baseball Video

Television broadcasts of professional Baseball games were selected for these experiments because (as suggested by the concept of *RVS* of section 2), it was possible to identify meaningful objects and scenes that are visually similar and repeat. First, we identified the batting scene of Fig. 2 as a meaningful candidate for a *VOD*. Then we collected and examined data.

Table 1. Some quality factors found in professional Baseball broadcast.

Visual Appearance	Signal quality
Time of game (day: natural light, evening: artificial light)	Reception (from Cable TV, antenna, satellite, etc.)
Daytime weather (sunny, cloudy, raining, etc.)	Origin (live game, videotaped game)
Evening weather (raining, foggy, etc.)	Internal-network transmission (via satellite, etc.)
Stadium (natural vs. artificial field, sand color)	Analog recording (VHS, S-VHS, EP/SP mode, etc.)
Teams (color of uniforms)	Digital encoding (MPEG-1,2, parameters, etc.)
Broadcast Network (camera angles, text-on-screen, etc.)	Noise, human error

In constructing a classifier for the batting scene of Fig. 2, we encountered several issues of importance (see Table 1 discussed in previous work [29]). We divided such factors into those that are related to visual appearance (i.e., independent of the signal), and those that

are related to the broadcast signal itself. These factors cause variations in the visual appearance of the *content* used by the algorithms, and therefore on the value of the *features* (segmentation, color, shape, etc.) used. The effect varies from minor to significant. For example, the time of day (morning, afternoon, night) can significantly affect the lighting conditions, which have an impact on the perception (and encoding) of color and texture. It is interesting to note that, due to the length of some Baseball games (several hours), it is possible to observe significant variations in weather (from sunny to overcast to rainy) and lighting (it is not uncommon for a game to start in the afternoon and end in the evening). Other factors, such as the players' uniform and the field remain constant within a game, but can vary significantly across different games. The way the games are broadcast (e.g., number of cameras, angles, etc.), on the other hand, is fairly standard within a game and across different broadcasters. Analyzing the data carefully, however, it is easy to observe variations that although minor for humans, can have a severe impact on CBR algorithms. Examples include "small" variations in camera angles (or camera distance), text on the screen, and others.

The second set of factors was also surprisingly important. Variations in the signal, even within the same game were sometimes significant. Colors changed, and noise was visible in many cases. Some of these are due to human error at origin, while others are related to the broadcast mechanism itself (live over satellite, etc.). Of course, variations in digitization of the signals can also have a strong impact.

For humans, most of those factors have no impact on the ability to recognize different scenes. Issues such as clutter (i.e., presence of unknown/unmodeled objects), occlusion, variations in lighting, and others,[24] are well known in Computer Vision and can have a strong impact on automatic algorithms. Most of these issues, however, are ignored in most of the experiments reported in CBR, mainly because in most cases the data used for training/testing comes from a single “collection” (e.g., a particular news source, etc.).

In order to test the framework with a batting scene detector, we used videos from several broadcasters. The set used in the experiments included games played in different stadiums (natural and artificial turf), times of day (night/day), weather conditions (overcast, sunny), etc.

Innings from 6 different games were digitized in MPEG1 format at 1.5 MBps (30 frames/sec, at resolution 352x240 pixels). All scene cuts were obtained manually (scene cuts could be detected automatically using [43]), and each shot was forced to a length of 30 frames, which is long enough to represent a batting (pitching) action. The batting (or pitching scene) lasts approximately 1 second in a television broadcast.

A set of 376 baseball video shots (of 30 frames each) was used for the experiments. The set contained 125 batting scene shots (Fig. 2). The set of 376 was divided into independent training and testing sets. The training set consisted of 60 batting scene shots. The test set, then, consisted of 316 shots (65 batting scenes and 251 other types of scenes), and *different* games were used in the training and test sets. The definition hierarchy used differed slightly from the one in Fig. 2: the field *object-part* was divided into three perceptual areas: mound, top grass, and bottom grass.

Since classification starts at the region level, we examine classification results for different region classes. As discussed in section 3.2.4, different classifiers may perform differently on the same training data, and therefore it may be beneficial to use different

algorithms for different classification problems. For our particular experiments, this is illustrated in Fig. 6. The figure shows the learning curves for *region* node classifiers constructed using the ID3 algorithm, and using a k-Nearest Neighbor classifier (k=5). In the *VA* framework, cross-validation accuracy is used, over the *training set*, to select the best features and classifiers. For illustration purposes here (in Fig. 6 *only*), we show the learning performance over the *entire set* (training and testing sets together). The overall batting scene classifier *does not have access to the test set* during training, but we show it here because the differences between the curves are easier to observe than on the training set alone- the point of this discussion is to emphasize that an algorithm will perform differently on different sets of data. The curve for ID3, for example, suggests that an ID3 classifier will perform better on pitcher and grass nodes. An IB-5 classifier shows similar performance variations on different sets of data. At the same time, the plots show that the ID3 algorithm is more likely to perform better for the batter regions than the IB5 classifier. In the actual cross-validation experiments over the training set (not shown), different algorithms and features were selected for the construction of classifiers at different nodes (some examples are presented in [29]). Performance variations over the training set varied depending on the node, and most of the region level classifiers achieved around 80% accuracy on the independent test set.

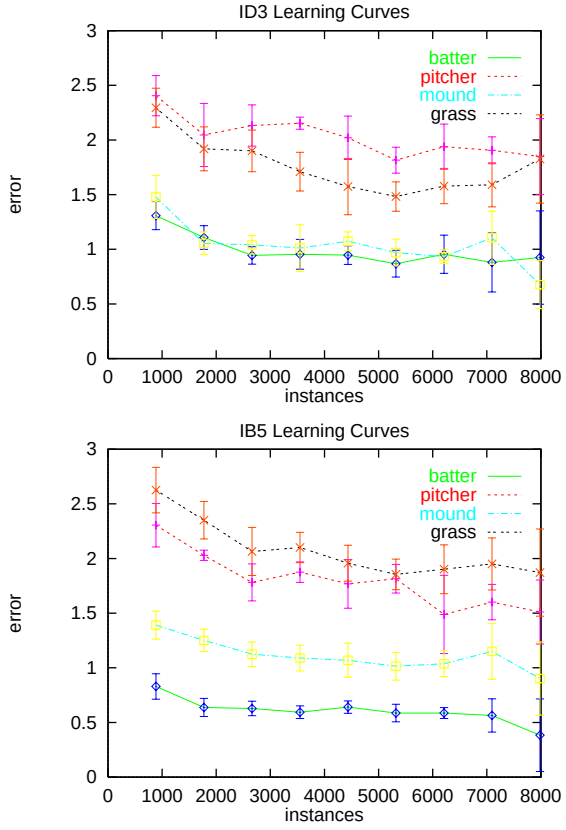

Fig. 6. Learning curves that show number of training examples vs. error rate, for two different algorithms (top: ID3 and bottom: K-Nearest Neighbor, K=5) on the same set of data. The error bars represent 95% confidence intervals, and the error corresponds to the total percentage of misclassifications.

Detection of video shots of the batting scene (Fig. 2) using the entire hierarchy resulted in an overall accuracy of 92% (overall % of correct classifications) with 64% recall, and 100% precision on the independent test set of 316 shots (65 batting scenes and 251 non-batting scenes). High precision was achieved in the *VA* in these experiments because the current implementation of the framework requires the detection of all nodes in the hierarchy. In other words, a batting scene can only be detected if all components of the hierarchy are found in the scene/shot. Therefore, a detector for this scene is unlikely to encounter false positives for all of the components of the hierarchy within a single scene. This mechanism, however, also causes a drop in recall: a classification error (miss) in one node of the hierarchy can cause a dismissal of a batting scene shot. In general, therefore, a hierarchy with more nodes is likely to yield higher precision and lower recall. Fewer nodes are more likely to yield lower precision and higher recall. Indeed, the shots that were missed by the classifier were missed because not all of the nodes were present. In particular, in most of the misclassifications the smaller elements (e.g., mound, batter) could not be found. This was due to segmentation errors, and errors at different levels of the hierarchy. In some cases text (and borders) surrounding the scene caused the errors- it is not uncommon for the entire scene to be reduced by a border with text (e.g., statistics or information from other games being played at the same time), making detection very difficult.

Detection of the batting scene across different games (with the variations outlined in Table 1) is a difficult problem on which the *VA* has performed well. Preliminary experiments using global features (quantized LUV color histogram and coarseness) as well as block-based global scene classification (breaking up each image into 16 blocks, classifying the blocks and assigning the image the majority of the block labels[74]) produced poor performance. Although more experiments are required to compare the *VA*'s performance with other approaches (e.g., using the same features as in [74,77] and testing the implementation with a set of similar images), an analysis of the data and the preliminary experiments suggest that scene-level classification (i.e., not using structure information) may not yield good results for this particular problem. One of the biggest difficulties is the variation that occurs across different games. The important components of the batting scene (i.e., those included in the hierarchy of Fig. 2) usually occupy around one third of the image (scene). A global approach to classification, therefore, is likely to be affected by the remaining two thirds of each scene. Because of variations in the stadium, for example, the background (e.g., wall behind the pitcher) can be significantly different across different scenes. The *VA* framework takes this into account in the sense that if the background is not included in the hierarchy, it may not have a direct impact on the detection of the object/scene. A related observation is that, in this particular application, there are many similar scenes that do not match the model. There are many scenes that show a field and a crowd in the background. Such scenes, however, do not contain a batter (and pitcher), so a *VOD* that includes such elements would be able to differentiate between one of those "field" shots and a batting scene. It would be more unlikely for a global (or block-based) classifier, on the other hand, to be able to make such distinctions.

A possibility to alleviate the problem of variation present in the data, is to perform a filtering of the shots.[80] In that approach, incoming video scenes are first automatically assigned to a "color model" based on unsupervised learning (e.g., different models for different games- night, sunny, etc.), and a subsequent process uses manually constructed

rules at the segmented region level (to account for local scene structure) to perform classification. Promising preliminary results (precision 96%, recall 97%) was reported in detecting batting scenes in broadcast videos.[80] However, note that unlike the *VA* the approach uses manually constructed region-level rules, and adaptive filtering that automatically selects the global color model depending on color variations of the new video. Indeed, it would be a promising direction to incorporate the adaptive filtering as a pre-filter before applying the *VA* detector.

4.1.2 Handshakes and skies

Detectors were also constructed for handshake, and sky images (see object hierarchies for handshakes and skies in Fig. 7). For the handshake tests, 80 training images, and an independent test set of 733 news images were used. Out of the 733 images, 85 were handshakes. An overall accuracy of 94% (94% of the set of 733 images, were correctly classified) was achieved (74% recall, 70% precision) with 89 images automatically labeled as handshake by the system. Sky detection was performed on a set of 1,300 images that contained 128 skies (with an independent training set of 40 images, see [55]). An accuracy of 94% was achieved (50% recall, 87% precision), in a set of 134 images retrieved.

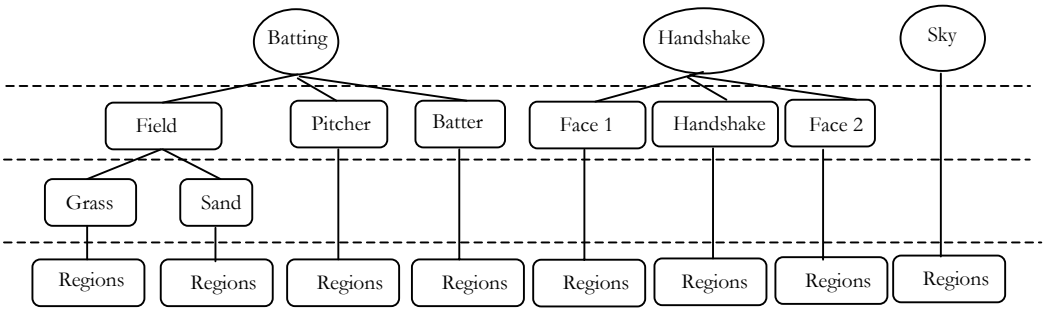

Fig. 7. Example object definition hierarchies. The first hierarchy was not used in experiments, but shows how the close-up of a player could be modeled. The other two hierarchies were used in the experiments reported.

The results reported for the different types of hierarchies show that the Visual Apprentice framework is flexible, allowing the construction of different types of detectors. More importantly, performance in each case was similar to performance reported for similar classifiers using other techniques (overall accuracy around 90% and higher). The experiments show encouraging results for the construction of dynamic approaches to classification. Next we describe some possible extensions, and improvements.

5. DISCUSSION

5.1 Extensions to the framework

The framework of the *VA* shows several desirable characteristics of CBR systems. The system uses learning techniques to automatically build classifiers, and therefore detectors can be easily constructed without the need for specialized algorithms. Since classifiers are built independently (for each node of the hierarchy), however, specialized algorithms can be easily incorporated. For example, in the handshake classifier, a face detection module could be used instead of the face node classifiers. Similarly, a domain-specific segmentation algorithm could be used to improve performance. In the current implementation a "standard" set of parameters is used with the segmentation algorithm. The parameters, however, could depend on the specific class (and hierarchy) being constructed by the user, or even learned by the system based on correct/incorrect segmentation results (labeled by the user).

The construction of the hierarchy, as discussed earlier, is subjective and will depend on the user. Therefore, two hierarchies for the same class (e.g., batting scene) may lead to different classification results. It is conceptually possible to build an hierarchy automatically, or semi-automatically. This issue is somewhat related to the learning of belief networks[54], and research in which the goal is to automatically detect Regions of Interest (ROIs). ROIs are areas that would roughly correspond to nodes in an hierarchy (i.e., areas of the image which are more important than others[59]). In [25], for example, experiments were presented to explore the use of eye-tracking results for automatic classification. Potentially this type of interaction could replace the current mode of interaction in the training stage of the *VA*, and help in the automatic or semi-automatic construction of hierarchies. A related issue is allowing more flexibility in the construction of hierarchies, and the application of the *VODs*. For example, instead of requiring all nodes to be labeled (and present during classification), it would be possible to extend the framework to allow the omission of nodes. A batting scene, then, could be detected (with a smaller confidence score), even if a pitcher is not detected.

Another possible extension of the *VA*, could include (at the user's expense) additional input parameters that could be used by the system to guide the training process. Information on issues such as desired computational efficiency (e.g., training/classification speed), for example, could be used internally in the selection of classifiers, and in providing training guidelines (e.g., size of training set, etc.).

5.2 On Applications

With the *VA* it is possible, to construct classifiers for objects/scenes that are visually similar, *and* have a structure that can be clearly defined. Applications in sports commercial domains (e.g., databases of retail objects) seem promising. The approach, however, may be unsuitable for classes in which variation in visual appearance is too significant, or in which a well-defined structure is not easily identified (e.g., indoor, outdoor images). Although it is conceivably possible to build several disjoint hierarchies for such classes (rather than having a single one), it is likely for other approaches that have produced promising results (e.g., [74,55]) to be more suitable.

It is also important to point out that in some domains, specialized algorithms may be better than flexible frameworks (e.g., the *VA*). An interesting possibility, however, is the combination of approaches like the *VA* with approaches that use expert knowledge. The *VA*, for example, could be used by an expert to construct rule-based classifiers, and those classifiers could be manually refined by the expert to improve their performance in a specific application. The framework could also be used to quickly examine feature variations for different types of objects (e.g., analyzing the training data), and to construct basic components to use in expert-constructed systems (e.g., use of a sky detector in a larger framework). The batting scene rules in the sports event detection discussed earlier,[80] for example, were constructed by an expert by analyzing features extracted by the *VA* during training. High accuracy was achieved in that system using this approach, suggesting that the *VA* framework can also be a useful tool for experts constructing domain-specific classifiers.

6. CONCLUSIONS AND FUTURE DIRECTIONS

We presented a new approach to the construction of dynamic classifiers for CBR. In the *Visual Apprentice (VA),* a user defines visual object/scene models, that depend on the classes in which she is interested, via a multiple-level *definition hierarchy* (*region, perceptual-area, object part, object*, and *scene*). As the user provides examples from images or video, visual features are extracted and classifiers are learned for each node of the *hierarchy*. At each node, the best features and classifiers are selected based on their performance, using k-fold cross-validation over the training set. The resulting structured collection of classifiers (a *Visual Scene/Object Detector*) can then be applied to new images/videos. A new image/video is first segmented automatically, and then the classifiers (*region, perceptual-area, object part, object, scene*) are applied according to the hierarchy.

The concept of *Recurrent Visual Semantics (RVS)* was also discussed. *RVS* is defined as the repetitive appearance of elements (e.g., objects, scenes, or shots) that are *visually similar* and have a common level of meaning within a specific context. Using that concept, it is possible to identify where and when learning techniques can be used in CBR.

Experimental results were presented in the detection of baseball batting scenes, handshake images, and skies. The results of the experiments are promising. The framework is flexible (users are allowed to construct their own classifiers, accommodating subjectivity); no input is required on difficult issues, such as the importance of low-level features, and selection of learning algorithms; and performance is comparable to that of other approaches. One of the main advantages of our framework is the flexibility of defining object/scene hierarchies and detailed user input at multiple levels. The approach allows users to specify multiple level composition models, which are absent in most existing approaches to CBR.

Future work includes further research into classifier combination, semi-automatic hierarchy construction, and MPEG-7 compatibility for the generation of features during learning and classification. Other topics of future research also include feature and classifier selection in with a small number of samples, the development of a theoretical

framework for the hierarchical classification scheme we propose, and the inclusion of additional multimedia features (e.g., audio).

7. REFERENCES

1. D. Aha, Editor. *Lazy Learning*. Kluwer Academic Publishers, The Netherlands, 1997.
2. A. Amir, and M. Lindenbaum, "A generic grouping algorithm and its quantitative analysis," *IEEE Transactions on Pattern Recognition and Machine Intelligence (PAMI)*, 20(2):186-192, February 1998.
3. J.R. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R.C. Jain, and C. Shu, "The VIRAGE Image Search Engine: An Open Framework for Image Management," in *proceedings of SPIE Storage and Retrieval for Still Image and Video Databases IV*, vol. 2670:76-87, February 1996.
4. L.D. Bergman, V. Castelli, C.-S. Li and J.R. Smith, "SPIRE, a Digital Library for Scientific Information," *International Journal of Digital Libraries, Special Issue* "In the Tradition of the Alexandrian Scholars", 3(1):85-99, July 2000.
5. G. Briscoe and T. Caelli, editors. A *Compendium of Machine Learning,* Ablex series in Artificial Intelligence. Norwood, NJ, 1996.
6. L.D. Bergman, and V. Castelli, editors. *Image Databases, Search and Retrieval of Digital Imagery*. John Wiley & Sons, New York (forthcoming).
7. S.-F. Chang, T. Sikora, and A. Puri, "Overview of the MPEG-7 Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, special issue on MPEG-7, June 2001 (to appear).
8. S.-F. Chang, W. Chen, H.J. Meng, H. Sundaram, and D. Zhong, "A Fully Automatic Content-Based Video Search Engine Supporting Multi-Object Spatio-temporal Queries," *IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on Image and Video Processing for Interactive Multimedia*, 8(5):602-615, September 1998.
9. S.-F. Chang, J.R. Smith, M. Beigi and A. Benitez, "Visual Information Retrieval from Large Distributed On-line Repositories," *Communications of the ACM*, 40(12):63-71, December, 1997.
10. S.-F. Chang, B. Chen and H. Sundaram, "Semantic Visual Templates: Linking Visual Features to Semantics," *in proceedings of International Conference on Image Processing (ICIP '98), Workshop on Content Based Video Search and Retrieval*, pages 531-535, Chicago IL, October, 1998.
11. S.-F. Chang, Q. Huang, T.S. Huang, A. Puri, and B. Shahraray., "Multimedia Search and Retrieval," in A. Puri and T. Chen, eds., *Advances in Multimedia: Systems, Standards, and Networks*, Marcel Dekker, 1999.
12. W. Cohen, "Learning Trees and Rules with Set-valued Features", *Thirteenth National Conference on Artificial Intelligence, AAAI 1996*, Portland, Oregon, August 1996.
13. A. Del Bimbo. *Visual Information Retrieval*. Morgan Kaufmann Publishers, San Francisco, USA, 1999.
14. T.G. Dietterich, "Proper Statistical Tests for Comparing Supervised Classification Learning Algorithms," *Technical Report*, Department of Computer Science, Oregon State University, 1996.
15. R.O. Duda, P.E. Hart, D.G. Stork, *Pattern Classification*, John Wiley & Sons, New York, 2001.
16. B.A. Draper, "Learning Object Recognition Strategies," *Ph.D. thesis*, Computer Science Department, University of Massachusetts, MA, 1993.
17. G. Durand, C. Thienot, and P. Faudemay, "Extraction of Composite Visual Objects from Audiovisual Materials," in *proceedings of SPIE Multimedia and Archiving Systems IV*, vol. 3846:194-203, Boston, MA, 1999.

18. D.A. Forsyth and M. Fleck, "Body Plans," in proceedings of *IEEE Computer Vision and Pattern Recognition (CVPR '97)*, pages 678-683, San Juan, Puerto Rico, 1997.
19. C. Frankel, M.J. Swain and V. Athitsos, "WebSeer: An Image Search Engine for the World Wide Web," *University of Chicago Technical Report TR-96-14*, July 31, 1996.
20. W.E.L. Grimson. *Object Recognition by Computer: The Role of Geometric Constraints.* MIT Press, Cambridge, MA 1990.
21. B. Gunsel, A.M. Ferman, and A.M. Tekalp, "Temporal video segmentation using unsupervised clustering and semantic object tracking," *IS&T/SPIE Journal of Electronic Imaging,* 7(3):592-604, July 1998.
22. D. Healey, "Preattentive Processing in Visualization," *http://www.cs.berkeley.edu/~healey/PP/PP.shtml*
23. K. Ikeuchi, and M. Veloso, editors. *Symbolic Visual Learning*. Oxford University Press, New York, 1997.
24. A. Jaimes, and S.-F. Chang, "Concepts and Techniques for Indexing Visual Semantics", in L.D. Bergman, and V. Castelli, editors. *Image Databases, Search and Retrieval of Digital Imagery*. John Wiley & Sons, New York (forthcoming).
25. A. Jaimes, J.B. Pelz, T. Grabowski, J. Babcock, and S.-F. Chang, "Using Human Observers' Eye Movements in Automatic Image Classifiers" in *proceedings of SPIE Human Vision and Electronic Imaging VI*, San Jose, CA, 2001.
26. A. Jaimes, A.B. Benitez, S.-F. Chang, and A.C. Loui, "Discovering Recurrent Visual Semantics in Consumer Photographs," invited paper, *International Conference on Image Processing (ICIP 2000)*, *Special Session on Semantic Feature Extraction in Consumer Contents*, Vancouver, Canada, September 10-13, 2000.
27. A. Jaimes and S.-F. Chang, "Model-Based Classification of Visual Information for Content-Based Retrieval," in *proceedings of SPIE Storage and Retrieval for Image and Video Databases VII*, vol. 3656:402-414, San Jose, CA, January 1999.
28. A. Jaimes and S.-F. Chang, "Integrating Multiple Classifiers in Visual Object Detectors Learned from User Input," Invited paper, session on Image and Video Databases*, 4th Asian Conference on Computer Vision (ACCV 2000)*, vol. 1:376-381, Taipei, Taiwan, January 8-11, 2000.
29. A. Jaimes and S.-F. Chang, "Automatic Selection of Visual Features and Classifiers," in *proceedings of SPIE Storage and Retrieval for Media Databases 2000*, vol. 3972:346-358, San Jose, CA, January 2000.
30. A. Jaimes and S.-F. Chang, "A Conceptual Framework for Indexing Visual Information at Multiple Levels," in *proceedings of SPIE Internet Imaging 2000*, vol. 3964:2-15. San Jose, CA, January 2000.
31. A. Jain and D. Zongker, "Feature Selection: Evaluation, Application, and Small Sample Performance," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 19(2):153-158, February 1997.
32. G.H. John, R. Kohavi, and K. Pfleger, "Irrelevant Features and the Subset Selection Problem," in *proceedings of 11th International Conference on Machine Learning (ICML '94)*, pages 121-129, 1994.
33. C. Jorgensen, A. Jaimes, A. B. Benitez, and S.-F. Chang, "A Conceptual Framework and Research for Classifying Visual Descriptors," invited paper, *Journal of the American Society for Information Science (JASIS), special issue on "Image Access: Bridging Multiple Needs and Multiple Perspectives,"* Spring 2001 (to appear).
34. R. Kasturi and R.C. Jain, editors*. Computer Vision: Principles*. IEEE Computer Society Press, 1991.
35. J.M. Keller, M.R. Gray and J.A. Givens Jr., "A fuzzy k-nearest neighbor algorithm," *IEEE Transactions on Systems Man, and Cybernetics,* 15(4):580-585, July/August 1985.
36. R. Kohavi, "Feature Subset Selection Using the Wrapper Model: Overfitting and Dynamic Search Space Topology," in *proceedings of First International Conference on Knowledge Discovery and Data Mining*, pages 192-197, 1995.
37. R. Kohavi, "A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model

Selection," in *proceedings of the 14th International Joint Conference on Artificial Intelligence (JCAI '95)*, pages 1137-1143, Morgan Kaufmann Publishers, Inc., 1995.
38. R. Kohavi, G. John, R. Long, D. Manley, and K. Pfleger, "MLC++: A Machine Learning Library in C++," in *proceedings of Conference on Tools with Artificial Intelligence 94*, 1994.
39. D. Koller and M. Sahami, "Toward Optimal Feature Selection," in *proceedings of 13th International Conference on Machine Learning (ICML)*, Bary, Italy, July 1996.
40. P. Lipson, "Context and Configuration Based Scene Classification," *Ph.D. thesis,* MIT Electrical and Computer Science Department, September 1996.
41. D.G. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, Boston, 1985.
42. D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information.* W. H. Freeman and Company, San Francisco, 1982.
43. J. Meng and S.-F. Chang, "Tools for Compressed-Domain Video Indexing and Editing," in *proceedings of SPIE Conference on Storage and Retrieval for Image and Video Database*, vol. 2670:180-191, San Jose, February 1996.
44. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, New York, 1992.
45. T. Minka, and R. Picard, "Interactive Learning Using a Society of Models," *Pattern Recognition,* 30(4), 1997.
46. T. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.
47. A.W. Moore and M.S. Lee, "Efficient Algorithms for Minimizing Cross Validation Error," in *proceedings of the 11th International Conference on Machine Learning*, Morgan Kaufmann, 1994.
48. MPEG Multimedia Description Scheme Group, "Text of ISO/IEC CD 15938-5 Information technology - Multimedia content description interface: Multimedia description schemes", ISO/IEC JTC1/SC29/WG11 MPEG00/N3705, La Baule, France, Oct. 2000 (see also MPEG-7 website: http://drogo.cselt.stet.it)
49. P.M. Murphy, *UCI Repository of Machine Learning Databases- a machine-readable data repository*, Maintained at the department of Information and Computer Science, University of California, Irvine. Anonymous FTP from ics.uci.edu in the directory pub/machine-learning-databases, 1995.
50. M.R. Napahde, and T.S. Huang, "A Probabilistic Framework for Semantic Video Indexing, Filtering, and Retrieval," *IEEE Transactions on Multimedia,* 3(1): 141-551, March 2001.
51. S. Nayar and T. Poggio, editors. *Early Visual Learning*. Oxford University Press, New York, 1996.
52. A.Y. Ng, "On Feature Selection: Learning with Exponentially many Irrelevant Features as Training Examples," in *proceedings of International Conference on Machine Learning (ICML)*, 1998.
53. W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos and G. Taubin, "The QBIC Project: Quering Images by Content Using Color, Texture, and Shape," in *proceedings* of *SPIE Storage and Retrieval for Image and Video Databases*, vol. 1908:173-187, February 1993.
54. S. Paek, and S.-F. Chang, "The case for Image Classification Systems Based on Probabilistic Reasoning," in *proceedings, IEEE International Conference on Multimedia and Expo (ICME 2000)*, New York City, July, 2000.
55. S. Paek, C. L. Sable, V. Hatzivassiloglou, A. Jaimes, B. H. Schiffman, S.-F. Chang, K. R. McKeown, "Integration of Visual and Text based Approaches for the Content Labeling and Classification of Photographs," in *proceedings of ACM SIGIR '99 Workshop on Multimedia Indexing and Retrieval*, Berkeley, CA. August, 1999.
56. E.G.M. Petrakis and C. Faloutsos, "Similarity searching in large image databases," *University of Maryland Department of Computer Science, Technical Report,* No. 3388, 1995.
57. R.W. Picard, "Computer Learning of Subjectivity," *ACM Computing surveys*, 27(4):621-623, December 1995.
58. R.W. Picard, "A Society of Models for Video and Image Libraries," *MIT Media Laboratory*

Perceptual Computing Section Technical Report, No. 360, Cambridge, Massachusetts, 1996.
59. C.M. Privitera, and L.W. Stark, "Algorithms for Defining Visual Regions-of-Interest: Comparison with Eye Fixations," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22(9):970-981, September 2000.
60. T.R. Reed and J.M. Hans Du Buf, "A Review of Recent Texture Segmentation and Feature Extraction Techniques," *Computer Vision, Graphics, and Image Processing (CVGIP): Image Understanding*, 57(3), May 1993.
61. H.A. Rowley, S. Baluja, and T. Kanade, "Human Face Detection in Visual Scenes," *Carnigie Mellon University Technical Report CMU-CS-95, 158,* 1995.
62. Y. Rui, T.S. Huang, and S.-F. Chang, "Image Retrieval: Current Directions, Promising Techniques, and Open Issues," *Journal of Visual Communication and Image Representation*, No. 10:1-23, 1999.
63. Y. Rui, T.S. Huang, M. Ortega, and S. Mehrotra, "Relevance Feedback: A Power Tool for Interactive Content-Based Image Retrieval," *IEEE Transactions on Circuits and Video Technology*, 8(5):644-655, September 1998.
64. J. Russ. *The Image Processing Handbook*. 3rd edition, CRC Press, Boca Raton, Fl, 1999.
65. S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall series in artificial intelligence, Prentice Hall, Englewood Cliffs, N.J., 1995.
66. S.L. Salzberg, "On Comparing Classifiers: A Critique of Current Research and Methods," *Data Mining and Knowledge Discovery*, 1:1-12, 1999.
67. A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-Based Image Retrieval at the End of the Early Years", in *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22(12):1349-1380, December 2000.
68. J.R. Smith, and S.-F. Chang, "Multi-stage Classification of Images from Features and Related Text," in *proceedings Fourth DELOS workshop*, Pisa, Italy, August, 1997.
69. J.R. Smith and S.-F. Chang, "An Image and Video Search Engine for the World-Wide Web," in *proceedings of SPIE Storage & Retrieval for Image and Video Databases V*, vol. 3022:84-95, San Jose, CA, February 1997.
70. J.R. Smith and S.-F. Chang. "VisualSEEk: a fully automated content-based image query system," in *proceedings of the ACM Conference on Multimedia (ACM MM '96)*, pages 87-98, November, 1996.
71. U. Srinivasan, C. Lindley, and B. Simpson-Young, "A Multi-Model Framework for Video Information Systems," *in Database Semantics: Issues in Multimedia Systems*, pages 85-108, Kluwer Academic Publishers, January 1999.
72. D.L. Swets and J.J. Weng, "Efficient Content-Based Image Retrieval using Automatic Feature Selection," in *proceedings of International Conference on Computer Vision (ICCV '95)*, Coral Gables, Florida, November 1995.
73. T.F. Syeda-Mahmood, "Data and Model-Driven Selection Using Color Regions," *International Journal of Computer Vision*, 21(1/2):9-36, 1997.
74. M. Szummer and R.W. Picard, "Indoor-Outdoor Image Classification," in *proceedings of IEEE International Workshop on Content-based Access of Image and Video Databases*, pages 42-51, Bombay, India, 1998.
75. H. Tamura, S. Mori, and T. Yamawaki, "Textural Features Corresponding to Visual Perception," *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-8:6, June 1978.
76. A. Triesman, "Preattentive Processing in Vision," *Computer Vision, Graphics, and Image Processing (CGVIP),* 31:156-177, 1985.
77. A. Vailaya, M. Figueiredo, A. Jain, and H.J. Zhang, "Image Classification for Content-Based Indexing," *IEEE Transactions on Image Processing*, 10(1): 117-130, January, 2001.
78. A. Yoshitaka, and T. Ichikawa, "A Survey on Content-Based Retrieval for Multimedia Databases," *IEEE Transactions on Knowledge and Data Engineering*, 11(1):81-93, January/February 1999.
79. D. Zhong and S.-F. Chang, "Video Object Model and Segmentation for Content-Based Video Indexing," in *proceedings of IEEE International Conference on Circuits and Systems, Special session on Networked Multimedia Technology & Applications*, vol. 2:1492-1495, Hong Kong,

June, 1997.
80. D. Zhong and S.-F. Chang, “Structure Analysis of Sports Video Using Domain Models”, *IEEE International Conference on Multimedia and Expo (ICME 2001)*, Tokyo, Japan, 2001 (submitted).

Alejandro Jaimes received the Computing Systems Engineering degree from Universidad de los Andes (Bogotá, Colombia) in 1994, and the M.S. in Computer Science from Columbia University (New York City) in 1997.

At Columbia University he is currently pursing a Ph.D. degree in Electrical Engineering. Before joining the Ph.D. program in 1997, he was a member of Columbia’s Robotics and Computer Graphics groups. He was also a member of Columbia's Digital Libraries group. His recent work has focused on the use of learning techniques in content-based retrieval, the development of interactive frameworks for organizing digital image collections, MPEG-7, and human understanding of image and video content.

Shih-Fu Chang is an Associate Professor of Electrical Engineering at Columbia University. Prof. Chang received a Ph.D. degree of EECS from U.C. Berkeley in 1993. He currently leads Columbia's ADVENT industry-university research consortium, which focuses on representation, manipulation, searching, and transmission of multimedia content.

Prof. Chang also leads digital video research within several cross-disciplinary projects at Columbia, including Columbia's Health Care Digital Library Project supported by NSF's DLI Phase II initiative. He actively participates in international conferences and standardization efforts, such as MPEG-7. Prof. Chang has been a general co-chair of ACM Multimedia Conference 2000, an associate editor for several journals, and a consultant in several new media startup companies.

Prof. Chang has been awarded a Navy ONR Young Investigator Award in 1998, a Faculty Development Award from IBM in 1995, a CAREER Award from the National Science Foundation in 1995, and three best paper awards in the areas of video representation and searching. He is currently a Distinguished Lecturer of IEEE Circuits and Systems Society in the area of multimedia technologies and applications.