IS&T/SPIE Storage and Retrieval for Image and Video Databases VIII, Vol. 3972, San Jose, CA, Jan. 2000

Automatic Selection of Visual Features and Classifiers

Alejandro Jaimes and Shih-Fu Chang[1]
Image and Advanced TV Laboratory, Department of Electrical Engineering
Columbia University, 1312 S.W. Mudd Building Mail code 4712 Box F8
New York, N.Y. 10027

ABSTRACT

In this paper, we propose a *dynamic* approach to feature and classifier selection. In our approach, based on performance, visual *features* and *classifiers* are selected automatically. In earlier work, we presented the *Visual Apprentice*, in which users can define visual object models via a multiple-level *object definition hierarchy* (*region, perceptual-area, object-part, and object*). *Visual Object Detectors* are learned, using various *learning algorithms*- as the user provides examples from images or video, visual *features* are extracted and multiple *classifiers* are learned for each node of the *hierarchy*. In this paper, *features* and *classifiers* are selected automatically at each node, depending on their performance over the training set provided by the user. Thus, changes in the training data yield *dynamic* changes in the features and classifiers used. We introduce the concept of *Recurrent Visual Semantics* and show how it can be used to identify domains in which performance-based learning techniques such as the one presented can be applied. We then show experimental results in detecting Baseball video shots, images that contain handshakes, and images that contain skies. These results demonstrate the importance, feasibility, and usefulness of dynamic *feature*/*classifier* selection for classification of visual information, and the performance benefits of using multiple *learning algorithms* to build *classifiers*. Based on our experiments, we also discuss some of the issues that arise when applying learning techniques in real-world content-based applications.

Keywords: Content-based retrieval, automatic classification, feature selection, machine learning, object recognition, classifier selection.

1. INTRODUCTION

Over the last few years, an enormous amount of visual information has become easily accessible in digital form in the World Wide Web and through other sources. In addition, new technologies such as those used in TIVO and ReplayTV, seem promising in the sense that users are allowed to have more control of the content they receive and how they can use it. One of the biggest challenges for researchers, then, is the creation of techniques to automatically index content in a way that it fits the specific user's needs. Most of the recent research in content-based retrieval, however, has focused on manually creating classifiers for specific tasks- classifying specific types of scenes, or detecting specific objects (e.g., faces). These approaches are very useful, but suffer from two major drawbacks: (1) new classifiers have to be manually constructed, and (2) classifiers built are *static* and lack flexibility.

Additionally, in most automatic content-based classification approaches that use Machine Learning techniques, experts heuristically select the *algorithms* and *features* used. Decisions are usually based either on domain specific knowledge, or on past performance of *features* and *learning algorithms* (often on how well they have performed on standard data sets in the Machine Learning literature). It is thus generally assumed that a good heuristic choice of *learning algorithms* and *features* will yield good *classifiers*. Additionally, the *classifiers* and *features* used are static and cannot be changed.

While this approach has proven successful in some domains, it is clear that feature and classifier selection continue to be important aspects that have not been studied in depth in content-based retrieval. In this paper, we focus on these two fundamental issues in the context of content-based retrieval: (1) the automatic selection of features, and (2) the automatic selection of classifiers[2]. We present a new approach, based on performance, in which visual features and classifiers are selected automatically in building *Visual Object Detectors*. In the *Visual Apprentice* framework presented in [7], a user defines visual object models according to his interests via a multiple-level *object-definition hierarchy* (*region, perceptual-area, object part, object*). As the user provides examples from images or video, visual features are extracted and classifiers

[1] E-mail: {ajaimes, sfchang}@ee.columbia.edu WWW: http://www.ee.columbia.edu/{~ajaimes, ~sfchang}
[2] We make a distinction between a *classifier* (i.e., the function that performs classification) and the *algorithm* used to produce the classifier (e.g., a machine-learning algorithm).

are learned for each node of the *hierarchy*. In this paper, features and classifiers are selected automatically at each node, depending on their performance (using k-fold cross-validation) over the training set provided by the user. We show experimental results that compare performance of different features and classifiers (built using various machine-learning algorithms) at different nodes of the hierarchy. These results demonstrate the importance, feasibility, and usefulness of automatic feature/classifier selection for classification of visual information, the dependency that exists between the features used and the data, and the performance benefits of using multiple learning algorithms to build classifiers. We also introduce the concept of *Recurrent Visual Semantics*, and show how it helps identify domains in which learning techniques such as the one presented can be applied.

In most studies in the machine-learning community, the goal is to compare different algorithms. The criterion in those cases is often how well those algorithms perform on standard data sets (e.g., UC Irvine repository [20]), or in a particular domain. As pointed out in [28], such comparisons of algorithms may be misleading since the performance of the classifiers they produce depends strongly on the specific domain in which they are applied (e.g., features, training, etc.). For this reason, many researchers in the Machine Learning encourage the empirical evaluation of algorithms in their application domains (see comments in [19], and [15], among others). This further motivates the approach presented in this paper.

1.1. Previous work

The problem of selecting the right features is well known in content-based retrieval [24][25]. Most automatic classification/indexing approaches, however, have relied on using either domain specific features [6][31] or general features that have worked well in different domains (e.g., color histogram [29] is widely used). Additionally, in most cases, the *algorithms* themselves have been manually chosen by an expert [6]. In this respect, most previous approaches are *static* since the expert must manually change the classifiers and features chosen. Our approach is different from previous work since (1) the *features* used are selected automatically depending on the specific classifier being built, and (2) *classifiers*, built using multiple *algorithms*, are selected automatically according to how well they perform the desired task. Our framework is *dynamic* since the characteristics of the approach vary with the training data.

In FourEyes [18], models (e.g., features) are chosen by automatically selecting groupings (e.g., as produced manually or automatically using different features). Our approach is different since the features *and* classifiers are selected automatically. In [3], a scheme to automatically select color features is presented, but different classifiers are not used. Finally, in the *Visual Apprentice* we presented in [7], classifier selection is not performed and features are given weights based on a simpele variance measure.

Outline

In section 2 we introduce the concept of *Recurrent Visual Semantics*. In section 3 we give an overview of *The Visual Apprentice*, and present our approach. In section 4, we discuss issues in applying learning techniques in real-world content-based retrieval applications and experimental results.

2. RECURRENT VISUAL SEMANTICS

We define *Recurrent Visual Semantics* (*RVS*) 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 [8]. Examples of domains in which *Recurrent Visual Semantics* can be easily identified include news (e.g., in the conflict with Yugoslavia, images were repetitive: refugees, news conference, etc.) and sports. Baseball video, for example, is particularly interesting given the fairly standard way in which cameras are placed in the broadcast of professional games. The *RVS* is present in this domain because different visually similar elements (that have a common level of meaning) repeat. This repetition occurs at various levels: objects (e.g., players), scenes (e.g., batting scene below), shots (e.g., the camera motion after a homerun occurs), and shot sequences (e.g., a homerun shot sequence often includes the batting scene, a scene of the player running, etc.). Some examples are given in Fig. 1.

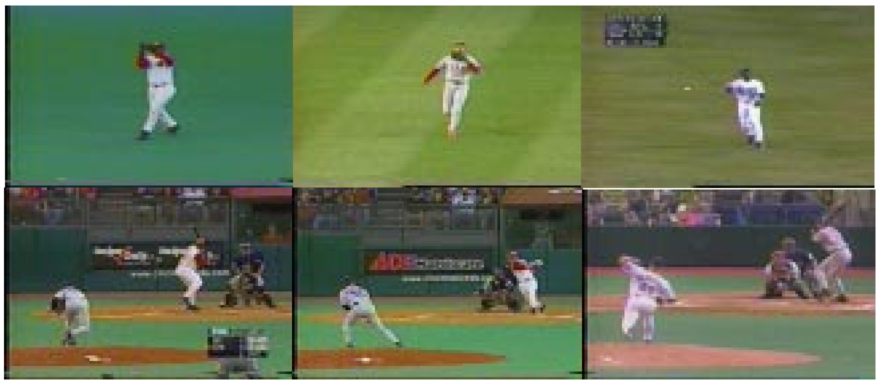

Fig. 1. Sample scenes to illustrate the existence of *Recurrent Visual Semantics* in professional Baseball broadcast.

The first step in deciding whether learning techniques are applicable (in the context of content-based retrieval), is to identify the existence of *RVS*. In other words, a domain is chosen, and the elements (objects, shots, scenes, etc.) that repeat are identified. We can refer to these as the *basic repetitive elements* in the domain in question. Note that these elements can be similar at different levels (e.g., in terms of syntactic or semantic attributes [10]). The main concern, however is with *visually similar* elements.

The existence of *RVS* motivates our approach of using learning techniques in content-based retrieval. Once *RVS* is identified, *Visual Object Detectors* can be learned. Recursiveness and domain constraints facilitate training and the future data inputs to the detectors, thus decreasing the possibility of errors. Next we briefly describe the *Visual Apprentice* framework.

3. LEARNING VISUAL OBJECT DETECTORS

3.1. The Visual Apprentice

In the *Visual Apprentice* presented in [7], users can build automatic *Visual Object Detectors* according to their interests. Learning is based on an *object-definition hierarchy* consisting of the following levels (Fig. 2): (1) region; (2) perceptual; (3) object-part; (4) object, and (5) scene.

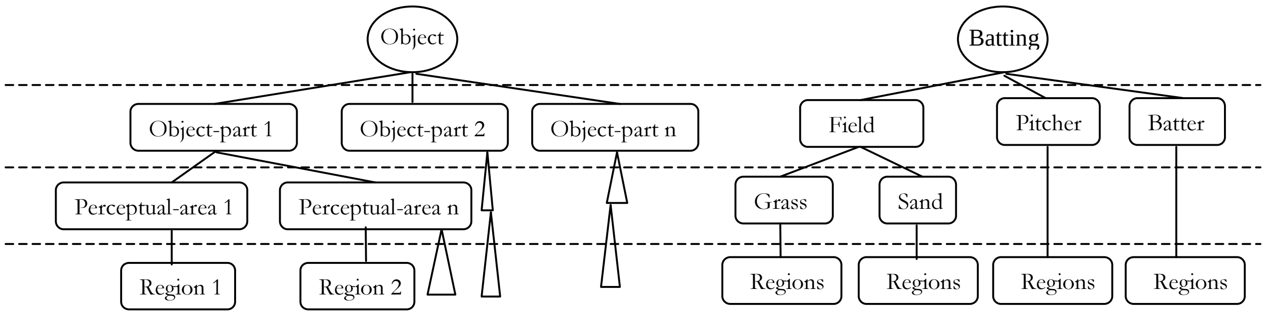

Fig. 2. General *Object Definition Hierarchy* and *hierarchy* for the image in figure 3. Every node in the tree has a conceptual meaning (e.g., pitcher), but also corresponds to a set of connected pixels in the image. This example shows how a scene can be modeled using the *hierarchy*.

To construct a *Visual Object Detector*, the user begins by selecting the training images/videos and building an *object-definition hierarchy* according to his interests. As described in [7], our recognition strategy is based on automatic segmentation of training and new images (Fig. 3)- during training, each image/video is automatically segmented and example regions are manually labeled by the user according to the *hierarchy* he defined. The labeled regions from all of the training examples are then used to compute the training set: features (color, texture, shape, spatial relationships, etc.) for each node of the *hierarchy* are automatically extracted and stored in a database. This data is then used for training by different learning *algorithms* that yield a set of *classifiers* at each node of the hierarchy (e.g., grass-regions, sand, etc.).

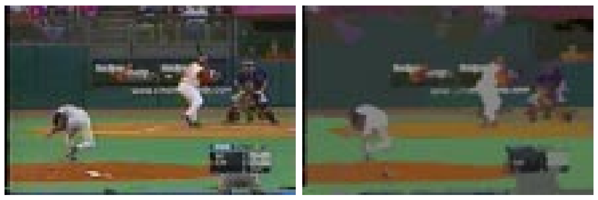

Fig. 3. Original and automatically segmented Baseball image.

The *Visual Object Detector* performs automatic classification by first applying automatic segmentation, and then combining classifiers and grouping strategies at the levels of Fig. 2: *regions* are classified first and combined to obtain *perceptual-areas* which are used by *object-part* classifiers. *Object-parts*, in turn, are combined and passed to *object* classifiers.

The object definition hierarchy of the Visual Apprentice framework is similar to the definition of Composite Visual Objects presented in [5]. In our framework, however, multiple classifiers are learned automatically. As in [5], our approach can automatically generate MPEG-7 descriptors [2].

3.2. Automatic Selection of Features and Classifiers

We use the framework of *The Visual Apprentice:* for the training set at each node of the *hierarchy* (Fig. 2), a *superset* of features is extracted. Then, different learning algorithms are trained, for each node of the hierarchy, and the best sub-set of features/classifiers is chosen automatically for each hierarchy node.

An overview of the general approach is given in Fig. 4. Features are selected first, and then used by multiple learning algorithms to build classifiers. This corresponds to the *filter* approach to feature selection. In the *wrapper* model (section 3.2.3), the performance of feature subsets is measured by classifiers. The feature selection process, then, depends on the learning algorithm being used. For this particular implementation of our framework, we chose the wrapper model [13] because in many cases different features can yield different results with different learning algorithms.

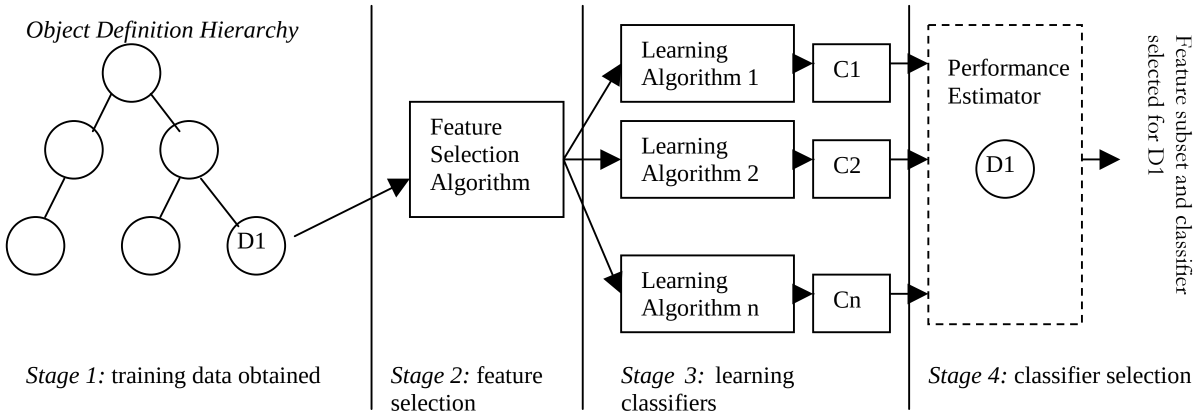
Object Definition Hierarchy
D1
Feature Selection Algorithm
Learning Algorithm 1
Learning Algorithm 2
Learning Algorithm n
C1
C2
Cn
Performance Estimator
D1
Feature subset and classifier selected for D1
Stage 1: training data obtained
Stage 2: feature selection
Stage 3: learning classifiers
Stage 4: classifier selection

Fig. 4. In our model, features and classifiers are chosen automatically for each node of the object definition hierarchy. D1 represents the training set for a leaf node, and Ci the classifiers built by each algorithm. Each classifier uses its own features and is generated using a particular algorithm. Performance of classifiers is compared and the best classifier, for each node, is selected.

In the following, sections we briefly describe the features and learning algorithms in the *Visual Apprentice*, and the feature and classifier selection methodology.

3.2.1. Features

As mentioned earlier, the problem of feature selection is well known in content-based retrieval. In most cases, researchers use domain specific features or features that have worked well in different domains. In contrast, we use a very large set of features and automatically choose the best subset for the specific task (e.g., classifier at each node of the hierarchy). We call this the *superset* of features.

As part of our growing *superset*, we have expanded the *Visual Apprentice* to include 43 features[3] in five groups: area and location, color, shape, texture, and motion (see the Appendix for a more detailed description).

3.2.2. Learning Algorithms

In this paper, we expand the work presented in [7] by using different learning algorithms. Again, ideally a large *superset* of learning algorithms would be available. In particular, we use the following algorithms [15]: ID3, Naïve-Bayes, IB, MC4 (see the Appendix for descriptions).

3.2.3 Feature Selection

The feature selection problem can be characterized as follows [11]: given a set of features *A* (e.g., the *superset* described above) 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. The criterion for any feature set *S* can be defined in terms of a function *C*(*S*), which gives high values for better feature sets and lower values for worse feature sets. One possibility is to define a criterion function as $(1-P_e)$, where P_e is the probability of error- this measure is *dependent on the learning algorithm used* and the *training* and *test data sets*. What we wish to find can be stated as:

$$C(B) = \max_{D \subseteq A} C(D)$$

Feature subset selection, then, can be considered a search problem for which several strategies have been proposed [11]- in many of them, the goal is to find the optimal feature sub-set without exhaustively searching all possibilities. In this paper, rather than focusing on a particular selection strategy, the goal is to show the benefits of automating the feature selection process.

Since the measure P_e is dependent on the *learning algorithm* and *data* used, we use the wrapper model described in [13]. In that model (fig. 5), feature selection is performed with respect to a particular classifier and data set.

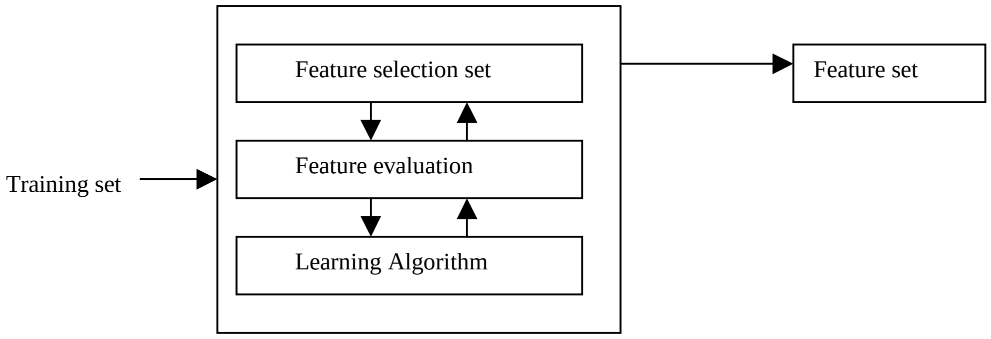

Fig. 5. The wrapper model for feature selection in [13]. The classifiers themselves are used to measure performance, so the features selected depend specifically on the *data* and *classifier* being used.

In the wrapper model, a search for the best features is performed, using a given learning algorithm, by building different classifiers using subsets of the original feature set. Once the best features have been selected, a classifier is learned using the same learning algorithm. The result of this stage (Fig. 4), then, is a classifier for the learning algorithm used. Since we use multiple learning algorithms, multiple classifiers are produced. In the experiments of section 4., we used best-first forward search to find the best features. This process does not guarantee the optimal feature subset, but the search is not exhaustive.

[3] We also use features that deal with spatial relationships between perceptual-areas and between object parts. We do not include those in the current discussion.

3.2.4. Classifier Selection

In most studies in the machine-learning community, the goal is to compare different algorithms. The criterion in those cases is often the performance of the algorithms on standard data sets (e.g., UC Irvine repository [20]), or in a particular domain. Instead of trying to find the best algorithms for classifying visual information, our goals are centered in determining the best *classifiers*[4] for specific tasks (e.g., nodes of the *object hierarchy*).

Several methods have been proposed for comparing classifiers [4][19]. In our approach, classifiers are selected according to how well they perform on the training data provided by the user: for each node of the *object definition hierarchy*, we have a training set S of n feature vectors $\boldsymbol{v_i}$. Specifically, $S=\{\boldsymbol{v_1},\ldots,\boldsymbol{v_n}\}$ where $\boldsymbol{v_i}$ contains the feature values (e.g., for the *superset* described above or the features that are selected for that specific classifier), and class label l (provided by the user during training) that correspond to sample i (e.g., $\boldsymbol{v_1}=\{l$ = pitcher, *color* L = 3.2, etc.$\}$). The set S is used by different *learning algorithms* to build *classifiers*.

In order to select the best classifier[5], first we measure each classifier's performance using *k-fold cross-validation* [14]: the set S 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. The accuracy on each fold can be determined in different ways, usually 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 (i.e., each algorithm produces a *classifier* and that *classifier's* accuracy estimate is obtained

The best classifier is chosen according to its performance estimate given by the cross-validation accuracy: for each node, the classifier with the highest score is selected. The process occurs for every node of the *hierarchy* defined by the user. Another option for selecting classifiers, are the measures of precision and recall. We did not use those measures for selection, but a will be seen in section 4, recall is very important in our framework.

4. EXPERIMENTAL RESULTS

In this section, first we describe our experimental setup and relevant testing issues in content-based retrieval, and then we discuss experimental results that show the benefits of feature/classifier selection.

4.1. Training and Test Set Issues

In the Machine Learning community, common data repositories (e.g., [20]) are often used to test algorithms. In content-based retrieval, however, different researchers use distinct sets of images/videos to test their approaches. In this section, we describe the data we used for our experiments, with the purpose of highlighting the following two issues: (1) application of content-based techniques in real-world scenarios; and (2) selection of training/testing data sets in a particular domain. We first describe the data and then how it was selected.

4.1.1. Baseball Video

Before selecting the data for our experiments, we obtained 30 hours of professional Baseball video from television broadcast, spanning 5 channels, 10 stadiums, and 10 teams. During the process of acquiring the data, we noticed several factors (Table 1) that influence variations in the quality and visual appearance of the scenes. For humans, most of these factors (e.g., lighting, grass color and texture variations, etc.) would have no impact on the ability to recognize different scenes. For a content-based classification system, however, these changes can often mean significant feature value variations. Most of these factors, however, are ignored in most of the experiments reported in content-based retrieval, mainly because in most cases the data used for training/testing comes from a single "collection" (e.g., Corel database, a particular news source, etc.).

[4] A *classifier* is a function that, given an input example, assigns that example to one of k classes. A *learning algorithm* is a function that, given a set of examples and their classes, constructs a classifier.
[5] Note that we refer to classifier comparison since the selection is based on how the classifiers perform on the training data.

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 grass vs. artificial grass, 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

Table 1. Some factors in real-world content-based retrieval applications, particularly in professional Baseball broadcasts.

Some of the factors listed on Table 1 can vary significantly within a single game, but others remain constant. Weather, for example, can be very different in two segments of the same game, while lighting might remain constant if the game is played at night. Differences across games depend on the stadium in which the game is played (patterns on the natural grass or artificial grass; lighting in outdoor, indoor, or semi-covered stadium fields, etc.), and team uniforms (a team usually has several uniforms), among others. It was surprising to find, for instance, that sometimes the variations in the quality of the signal were high within a game (mostly due to human error or noise).

All of these factors show the difficulty of applying content-based techniques in real-world scenarios.

4.2. Experiments

We performed experiments using Baseball video, and news images to detect skies and handshakes.

4.2.1. Baseball Video

As mentioned above, the initial data set consisted of 30 hours of baseball video, recorded on Super VHS format from cable television. The selection of training/test sets posed another challenge since there were so many factors involved in the variation of the data. The following possibilities were considered: (1) building of detectors for specific teams, stadiums, or times-of-day (e.g., day detector, night detector), and (2) building general detectors applicable under the different variations discussed in section 4.1.

As an initial step, using the concept of *Recurrent Visual Semantics* of section 2, we identified several scene classes that would be useful in Baseball video and that are suitable for our framework. We then selected the batting scene (Fig. 3) as the object class[6] for which to build an automatic classifier using the *Visual Apprentice*. This class was chosen because it marks a semantically meaningful event in Baseball.

We chose to train the system on one team to obtain classifiers that can be applicable under all other variations (e.g., different time-of-day, weather, stadiums, etc.). For training and testing, we digitized 12 innings from 6 different New York Mets games (see variations in Table 2). The video was digitized in MPEG1 format at 1.5 MBps (30 frames/sec.). For automatic scene cut detection we tested the compressed-domain method from [17], but did not obtain satisfactory results. Scene cut detection in this domain poses another challenge due to quick camera movements and the existence of large homogeneous regions (making motion estimation more difficult). All scenes were manually segmented, and each shot was forced to a length of 30 frames: after examining the data, we found that an average pitch lasts approximately 20 frames, therefore the use of more than 30 frames per shot is not necessary for this particular object class. The final training/test set consisted of 376 shots.

[6] As discussed earlier, an *object-definition hierarchy* defines an object class, and such a definition may correspond to a scene. See Fig. 2 for the *definition hierarchy* of the class chosen.

Teams	Natural Grass	Artificial Grass	Day Innings	Night Innings	Channels
7	6	6	6	6	4

Table 2. Variations in the training/test set.

To summarize, 12 innings were digitized in MPEG1 format. Scenes cuts where manually set, and 30 frames for each shot were obtained (uncompressed). These frames were then segmented automatically for training and testing.

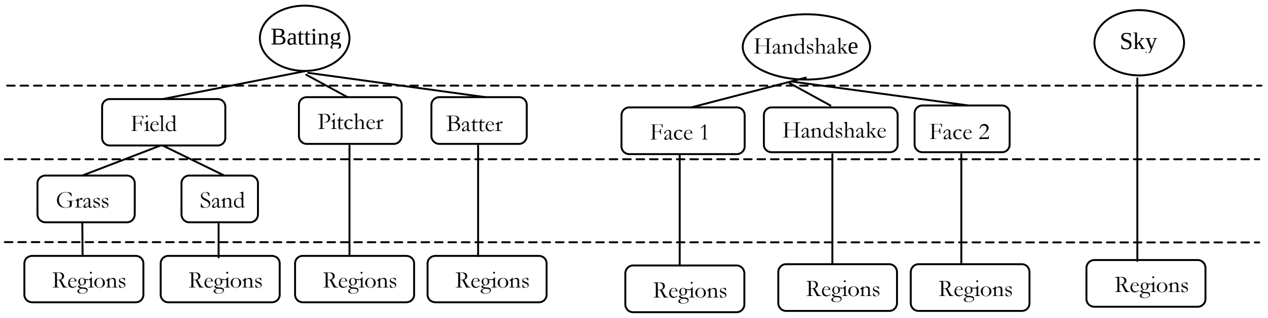

Fig. 6. Object definition hiearchies for the classes used in testing. In the actual hierarchy for the batting example, we included three perceptual areas: mound, top grass, and bottom grass.

4.2.2. Results

To show detailed results of feature/classifier selection, we focus on region level classifiers for the Baseball example. Then, we show final classification results for the Batting scene, handshakes, and skies.

In total, 376 shots where extracted from the 12 innings of Baseball video and the length of each shot was 30 frames. Out of the 376 shots, 125 where shots of the batting scene of Fig. 3. Out of the 125, 60 shots were employed by the user to train the system, based on an *object definition hierarchy* similar to the one depicted in Fig. 6. In the actual training though, the field object-part was divided into three perceptual areas: mound, top grass, and bottom grass.

Feature and classifier selection were performed using only the training set utilized by the user, so testing of the approach was performed on the remaining 316 shots. The following region-level classes were utilized for the tests: *pitcher, top grass (grass near the batter), mound (where the pitcher stands), bottom grass(near the pitcher), and batter.*

Since each classifier is built independently, the training instances at each node only have two labels: the label corresponding to the node (e.g., pitcher) and the negative examples (e.g., these are generated using the closed-world assumption: all pitcher regions are labeled pitcher, other regions are negative examples of pitcher).

Table 1 shows the best features selected for each algorithm (using the wrapper model described above) and the cross-validation error rate of the respective classifier (built using that algorithm with those features). In addition to showing the cross-validation performance (CV), we show the error rate for each classifier on the test set (consisting of the remaining 65 batting shots), and the recall for each class. In our particular framework, recall is very important since many false alarms are dismissed at higher levels of the hierarchy. If the system finds false pitcher regions, for example, the likelihood that they will form a false pitcher (i.e., a node at a higher level of the hierarchy) is very small.

The training set of 60 shots originally contained 3,395 regions, including 494 that correspond to each of the classes in the hierarchy (there were approximately 100 examples for each of the labels). Since each classifier is trained independently, in that training set, only 100 unknown regions where included- in other words, the training set consisted of 494 regions with the labels of table one, and 100 regions with the label unknown, for a total of 594 regions in the training set. It is important to note that the original distribution of the data plays an important role in the application of some learning algorithms. In this case, however, each classifier is concerned only with examples in two classes: unknown and the label that it seeks to classify (e.g., the pitcher classifier receives as input positive pitcher regions and unknown regions, which may correspond to regions of other labels). The *training set* was used to build classifiers and 10-fold cross-validation was done to determine the their performance.

	IB 1	**IB 3**	**IB 5**	**ID3**	**Naïve-Bayes**	**MC4**
Pitcher	[0, 1, 2, 4, 11, 12] *CV:* 0.43% *T:* 22.75% *R:* 116	[1, 2, 4, 7, 12, 15, 17, 37] *CV:* 0.43% *T:* 23.63% *R:* 118	[1, 2, 7, 12, 20, 37] *CV:* 0.58% *T:* 23.14% *R:* 115	1, 2, 7, 9, 11, 12, 19, 20, 37] *CV:* 1.30% *T:* 22.36% *R:* 115	[1, 2, 3, 6, 12, 18, 21, 30, 37] *CV:* 1.29% *T:* 17.60% *R:* 112	[1, 2, 10, 11, 15, 16, 17, 19, 20, 22, 23, 24, 25, 26, 27, 28, 30, 35, 38] *CV:* 0.45% *T:* 27.03% *R:* 95
Top Grass	[0, 2, 7, 8, 9, 21] *CV:* 0.14% *T:* 14.86% *R:* 142	[2, 5, 8, 9, 10, 11, 19, 35, 38] *CV:* 0.43% *T:* 16.25% *R:* 145	[0, 2, 7, 8, 9, 21,22] *CV:* 0.14% *T:* 15.06% *R:* 145	[1, 2, 6, 8, 11, 12, 38] *CV:* 1.01% *T:* 16.23% *R:* 128	[0, 2, 3, 4, 7, 9, 10, 11, 12, 13, 25, 32, 34] *CV:* 1.00% *T:* 13.36% *R:* 124	[1, 2, 6, 8, 9, 11, 12, 38] *CV:* 1.01% *T:* 16.31% *R:* 129
Bottom Grass	[0, 2, 4, 5, 7, 8, 13, 15, 20, 38] *CV:* 0.72% *T:* 15.45% *R:* 138	[2, 5, 13, 38] *CV:* 1.29% *T:* 15.51% *R:* 123	[2, 5, 6, 7, 8, 14, 20] *CV:* 0.57% *T:* 14.43% *R:* 143	[0, 2, 4, 5, 15, 16, 17, 38] *CV:* 1.86% *T:* 15.41% *R:* 124	[2, 5, 7, 8, 9, 10, 11, 12, 29, 35] *CV:* 1.00% *T:* 13.28% *R:* 119	[2, 4, 5, 38] *CV:* 1.72% *T:* 20.55% *R:* 126
Mound	[0, 2, 8] *CV:* 0.00% *T:* 15.51% *R:* 97	[2, 16] *CV:* 0.14% *T:* 19.04% *R:* 73	[2, 5, 6, 7, 8, 14, 20] *CV:* 0.57% *T:* 14.43% *R:* 143	[0, 2, 4, 5, 15, 16, 17, 38] *CV:* 1.86% *T:* 15.41% *R:* 124	[2, 5, 7, 8, 9, 10, 11, 12, 29, 35] *CV:* 1.00% *T:* 13.28% *R:* 119	[2, 4, 5, 38] *CV:* 1.72% *T:* 20.55% *R:* 126
Batter	[4, 7, 8, 10, 35, 37, 38] *CV:* 0.71% *T:* 19.86% *R:* 9	[0, 1, 16, 24, 25, 30, 38] *CV:* 0.72% *T:* 19.20% *R:* 31	[0, 1, 2, 20, 26, 27, 38] *CV:* 0.57% *T:* 19.36% *R:* 33	[2, 5, 20, 22, 35] *CV:* 0.86% *T:* 22.56% *R:* 10	[0, 2, 5, 8, 34] *CV:* 0.86% *T:* 21.99% *R:* 24	[0, 9, 33, 35, 38] *CV:* 1.58% *T:* 32.87% *R:* 27

Table 3. Different classifiers with their corresponding features and performance.

In Table 3, the number in brackets correspond to the features selected from the following set: (0) area, (1) center x, (2) center y, (3) orientation, (4) major axis length, (5) minor axis length, (6) major axis angle, (7) L, (8) U, (9) V, (10) dominant L, (11) U, (12) V, (13) perimeter, (14) form factor, (15) roundness, (16) aspect ratio, (17) compactness, (18) extent, (19) mean Maximum difference, (20) Mean MTV, (21) mean Horizontal, (22) Vertical, (23) Diagonal, and (24) anti-diagonal Local Directed Standard Deviation, (25) motion duration, (26) maximum horizontal, (27) vertical displacement, (28) minimum horizontal, (29) vertical displacement, (30) absolute horizontal, (31) vertical displacement, (32) trajectory length, (33) displacement length, (34) motion angle, (35) average speed in x, (36) y, (37) average acceleration in x, (38) y. The percentage values preceded by CV indicate the error rate (using 10-fold cross-validation) for the classifier built using the particular algorithm and features shown. That measure was taken over the training set of 60 shots. The Test and Recall values correspond to the performance of the particular algorithm on the reaming test set of 65 shots of the batting scene. Note that due to some implementation issues, some of the features listed in the appendix were not used in the experiments.

The table shows the variations of recall that occur when using different features and classifiers. Observing the results of mound classifiers, for example, we note significant differences between IB1 and IB5- if IB1 were used, many important mound regions would be missed. Many of the false alarms given by IB5, on the other hand, are often discarded by the higher level classifier (e.g., mound perceptual area). This highlights the advantage of having different classifiers (constructed using different learning algorithms) at each node- different classifiers perform differently depending on the data. It also shows the advantages of dynamically selecting features/classifiers. In addition, experiments like this one can provide some insight regarding which features and classifiers are most useful in the visual information domain. These results show the benefit of having various features and classifiers.

In each of the experiments performed, the test sets used were independent of the training sets. For the baseball batting scene classifier, 316 shots were used for testing. The test set included 65 batting scene shots, and, as above, an independent training set of 60 shots was used. We obtained 92% accuracy (64% recall, and 100% precision). This is not surprising, since detection of a batting scene implies detection of each of the components- all of the components of the object hierarchy must be present (if they were always provided during training)- it is unlikely to encounter false positives for all of them within a single shot. It is also important to note that a change in the hierarchy (e.g., remove the mound- higher accuracy?) implies changes in performance.

4.2.2. Handshakes and skies

For the handshake tests, we used 80 training images, and an independent test set of 733 news images. Out of the 733 images, 85 were handshakes. We obtained 94% accuracy (74% recall, 70% precision) in a set of 89 images automatically placed in the handshake class. Sky detection was performed on a set of 1,300 images that contained 128 skies. We obtained 94% accuracy (50% recall, 87% precision), in a set of 134 images retrieved. In future work, we plan to integrate handshake and sky detectors with the indoor/outdoor classification scheme in [23].

5. CONCLUSIONS AND FUTURE WORK

We have presented a new approach, based on performance, in which visual features and classifiers are selected automatically when building *Visual Object Detectors*. We use the framework of the *Visual Apprentice*, in which a user defines visual object models according to his interests via a multiple-level *object-definition hierarchy* (*region, perceptual-area, object part, object*). As the user provides examples from images or video, visual features are extracted and classifiers are learned for each node of the *hierarchy*. In this paper, automatic selection of features and classifiers is performed at each node, depending on their performance (using k-fold cross-validation) over the training set. We have also presented the concept of *Recurrent Visual Semantics (RVS)* and shown experimental results that support the benefits of having several learning algorithms and the automatic selection of features/classifiers.

Our approach to feature/classifier selection follows recent trends in experimental design, in which more general results are sought at the cost of additional computation. The motivation for our approach is two-fold: (1) performance of classifiers is strongly linked to training data, and (2) rapid growth in computer power facilitates the techniques adopted. In our dynamic framework, the additional cost of computation occurs during the training phase. Computational complexity is also increased, with the benefits of a more general framework that my often produce better performance than simpler (or manually constructed) approaches.

The work presented in this paper demonstrates the feasibility of having dynamic frameworks for content-based retrieval. Using learning techniques and different selection strategies, we are able to easily construct detectors that, constrained by *RVS*, can be successfully applied in complex real-world scenarios. More importantly, such detectors are *dynamic* and tailored to the specific needs of users who train them.

Our current work includes performing more experiments to build an *object definition hierarchy library*, and expanding our framework to include other information such as camera motion.

REFERENCES

[1] D. Aha, editor, *Lazy Learning*, Kluwer Academic Publishers, The Netherlands, 1997.

[2] A. B. Benitez, S. Paek, S.-F. Chang, A. Huang, A. Puri, C.-S. Li, J. R. Smith, L. D. Bergman, C. Judice, "Object-Based Multimedia Description Schemes and Applications for MPEG-7", To appear in *Image Communications Journal*, Invited Paper on a Special Issue on MPEG-7, 1999.

[3] M. Das and E.M. Riseman, “Feature Selection for Robust Color Image Retrieval”, *DARPA IUW,* New Orleans, LA, 1997.

[4] T.G. Dietterich, “Proper Statistical Tests for Comparing Supervised Classification Learning Algorithms”, *Technical Report*, Department of Computer Science, Oregon State University, 1996.

[5] G. Durand, C. Thienot, and P. Faudemay, "Extraction of Composite Visual Objects from Audiovisual Materials", SPIE Conference on Multimedia and Archiving Systems IV, Boston, MA, 1999.

[6] D. Forsyth and M. Fleck, “Body Plans”, *Computer Vision and Pattern Recognition 97,* San Juan, Puerto Rico, 1997.

[7] A. Jaimes and S.-F. Chang, "Model-based classification of visual information for content-based retrieval", *Storage and Retrieval for Image and Video Databases VII*, IS&T/SPIE, San Jose, CA, January 1999.

[8] A. Jaimes and S.-F. Chang, "Learning Visual Object Filters and Agents for on-line Media", *ADVENT Project Technical Report*, June, 1999.

[9] 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), Taipei, Taiwan, January 8-11, 2000.

[10] A. Jaimes and S.-F. Chang, "A Conceptual Framework for Indexing Visual Information at Multiple Levels", Internet Imaging 2000, IS&T/SPIE. San Jose, CA, January 2000.

[11] A. Jain and D. Zongker, "Feature Selection: Evaluation, Application, and Small Sample Performance", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, No. 2, February 1997.

[12] G. H. John, R. Kohavi, and K. Pfleger, "Irrelevant Features and the Subset Selection Problem", *11th International Conference on Machine Learning*, pp. 121-129, 1994.

[13] R. Kohavi, "Feature Subset Selection Using the Wrapper Model: Overfitting and Dynamic Search Space Topology", *First International Conference on Knowledge Discovery and Data Mining*, pp. 192-197, 1995.

[14] R. Kohavi, "A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection," *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann Publishers, Inc., pp. 1137-1143, 1995.

[15] R. Kohavi, G. John, R. Long, D. Manley, and K. Pfleger, "MLC++: A Machine Learning Library in C++", *Tools with Artificial Intelligence 94*, 1994.

[16] D. Koller andM. Sahami, "Toward Optimal Feature Selection", *13th International Conference on Machine Learning*, Bary, Italy, July 1996.

[17] J. Meng and S.-F. Chang, "Tools for Compressed-Domain Video Indexing and Editing,"*SPIE Conference on Storage and Retrieval for Image and Video Database*, Vol. 2670, San Jose, Feb. 1996.

[18] T. Minka, "An Image Database Browser that Learns from User Interaction", *M.I.T. Media Laboratory Technical Report* No. 365, Cambridge, Massachusetts, 1996.

[19] T. Mitchell, *Machine Learning*, McGraw-Hill, New York, 1997.

[20] 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.

[21] A.Y. Ng, "On Feature Selection: Learning with Exponentially many Irrelevant Features as Training Examples", *International Conference on Machine Learning*, 1998.

[22] J. Novovicová, P. Pudil, and J. Kittler, "Divergence Based Feature Selection for Multimodal Class Densities", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 2, February 1996.

[23] S. Paek, C. L. Sable, V. Hatzivassiloglou, A. Jaimes, B. H. Schiffman, S.-F. Chang, and K. McKeown, "Integration of Visual and Text-Based Approaches for the Content Labeling and Classification of Photographs", *ACM SIGIR 99, Workshop on Multimedia Indexing and Retrieval*, Berkeley, August 15-19, 1999

[24] R.W. Picard, "Computer Learning of Subjectivity," *ACM Computing surveys*, Vol. 27, No. 4, pp, 621-623, December 1995.

[25] 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.

[26] T. R. Reed and J.M. Hans Du Buf, "A Review of Recent Texture Segmentation and Feature Extraction Techniques", *CVGIP: Image Understanding*, Vol. 57, No. 3, May 1993.

[27] J. Russ, *The Image Processing Handbook*, 3rd edition, CRC Press, Boca Raton, FL, 1999.

[28] S. L. Salzberg, "On Comparing Classifiers: A Critique of Current Research and Methods", *Data Mining and Knowledge Discovery*, 1, pp. 1-12, 1999.

[29] J. R. Smith and S.-F. Chang. "VisualSEEk: a fully automated content-based image query system.," *ACM Multimedia '96*, November, 1996.

[30] D. L. Swets and J. J. Weng, "Efficient Content-Based Image Retrieval using Automatic Feature Selection", *International Conference on Computer Vision*, Coral Gables, Florida, Nov. 1995.

[31] A. Vailaya, M. Figueiredo, A. Jain, and H.J. Zhang, "Content-Based Hierarchical Classification of Vacation Images", *IEEE Multimedia Systems*, Florence, Italy, June 1999.

APPENDIX

Features

Features are computed at each node of the object definition hierarchy. Regions, perceptual areas, object-parts, or objects are represented by feature vectors. We place the features from our current implementation into five different groups.

- *Area and location:* area, bounding box center (x, and y), orientation, major axis length, major axis angle, minor axis lenght.
- *Color*: average L, U, and V, dominant L, U, and V (LUV quantized to 166 colors [29]).
- *Shape*: perimeter, form factor, roundness, bounding box aspect ratio, compactness, extent (see [27]).
- *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 [26][31]).
- *Motion*: motion trajectory, maximum/minimum horizontal and vertical displacement, absolute horizontal/vertical displacement, trajectory length, displacement distance, average motion angle, average horizontal/vertical speed/acceleration motion length.

Learning Algorithms

For the experiments presented in this paper, we used the tools in [15].

- *Instance Based* [1]: instead of creating general, explicit descriptions of the target function to be learned, these algorithms simply store the training examples. In particular, given a new instance, its classification decision is determined by the following measure:

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^{n} \left(a_r(x_i) - a_r(x_j)\right)^2}$$

Where $a_r(x)$ denotes the value of the rth attribute of instance x, and $d(x_i, x_j)$ the distance between two instances.

$$f(x_q) \leftarrow \arg\max_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$$

Where $x_1 .... x_k$ are the *k* instances from the training set that are nearest to x_q (the instance to be classified), *f* is a target function of the form $f : \Re \rightarrow V$, V being the finite set $\{v_1 .,,, v_s\}$ (e.g., the set of classes) and finally, $\delta(a,b) = 1$ if *a=b* and $\delta(a,b) = 0$ otherwise (in the case of discrete case) . In the continuous case, $\delta(a,b)$ is a function of *d(x,y).*

- *ID3* [19]: this algorithm constructs decision trees by ordering classification rules, based on the attribute values present in the training data. The algorithm begins with an empty tree, and uses a statistical test to determine which feature alone is best for classifying the training examples. A descendant of that node is then created for each possible value of that attribute and the training examples are sorted to the appropriate descendant node. The process is repeated for the training examples of each node. We use Information Gain to determine how well a given attribute separates the training examples according to their target classification:

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\otimes} \log_2 p_{\otimes}$$

where S is a collection of positive and negative examples of a target concept, $p_{\oplus}$ is the proportion of positive examples in S and $p_{\otimes}$ is the proportion of negative examples in S. Information Gain is defined as follows:

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where Values(A) is the set of all possible values for attribute A, and S_v is the subset of S for which attribute A has value *v*.

Decision tree algorithms, among other learning algoritms, perform some form of feature selection. Even in those cases, however, our approach to select features (even if the algorithms also have such fucntionality) is beneficial. The justification and benefits in performance of selecting features *before* using those algorithms, for example, is given in [21][16]. Particular examples of beneficial feature selection for ID3 are given in [12] .

- *Naïve-Bayes:* based on the Bayes theorem as follows:

$$Vmap = \arg\max_{v_j \in V} \frac{P(a_1, a_2, ...a_n \mid v_j)P(v_j)}{P(a_1, a_2, ..., a_n)}$$

Where *Vmap* is the most probable target value for an instance given the attribute values $(a_1, a_2, ..., a_n)$ that describe the instance, and where again v_j is a target value (e.g., a class) from a set V as described above.

The Naïve-Bayes classifier makes the simplifying assumption that the attribute values are conditionally independent given the target value:

$$Out = a\,rg \max_{v_j \in V} P(v_j) \prod_i P(a_i \mid v_i)$$

Where *Out* denotes the target value output by the Naïve Bayes classifier.

The various $P(v_j)$ and $P(a_i \mid v_j)$ are estimated based on their frequencies over the training data. It is important to note that even though in practice the independence assumption rarely holds, this algorithm has produced results comparable to those of other approaches.

- MC4: This algorithm is similar to C4.5 (a commercial version of ID3), which includes prunning.