REFERENCES

[1] D. Aha, Editor. *Lazy Learning*. Kluwer Academic Publishers, The Netherlands, 1997.
[2] W. Cohen, "Learning Trees and Rules with Set-valued Features", *Thirteenth National Conference on Artificial Intelligence, AAAI 1996*, Portland, Oregon, August 1996.
[3] B. Gunsel, A.M. Ferman, and A.M. Tekalp, "Temporal video segmentation using unsupervised clustering and semantic object tracking", *Journal of Electronic Imaging* 7(3), 592-604, July 1998 SPIE IS&T.
[4] A. Amir, and M. Lindenbaum, "A generic grouping algorithm and its quantitative analysis", *IEEE Transactions on Pattern Recognition and Machine Intelligence*, Vol. 20 No. 2, February 1998.
[5] 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.
[6] J.M. Keller, M.R. Gray and J.A. Givens Jr., "A fuzzy k-nearest neighbor algorithm", *IEEE Transactions on systems man, cybernetics.* Vol SMC-15 No. 4, pp 580-585, July/August 1985.
[7] C. Carson, S. Belongie, H. Greenspan, and J. Malik, "Region-Based Image Querying", *CVPR 1997 Workshop on Content-Based Access of Image and Video Libraries*.
[8] S.-F. Chang, J.R. Smith, M. Beigi and A. Benitez, "Visual Information Retrieval from Large Distributed On-line Repositories", *Communications of the ACM*, December, 1997. ACM.
[9] S.-F. Chang, B. Chen and H. Sundaram, "Semantic Visual Templates: Linking Visual Features to Semantics", *ICIP 1998, Workshop on Content Based Video Search and Retrieval*, Chicago IL, Oct 10-12 1998.
[10] D.A. Forsyth and M. Fleck, "Body Plans", *Proceedings Computer Vision and Pattern Recognition - CVPR* 97, San Juan, Puerto Rico, June 1997.
[11] C. Frankel, M.J. Swain and V. Athitsos, "WebSeer: An Image Search Engine for the World Wide Web", University of Chicago Tech. Report TR-96-14, July 31, 1996.
[12] W.E.L. Grimson, *Object Recognition by Computer: The Role of Geometric Constraints.* MIT Press: Cambridge, 1990.
[13] A. Jaimes and S.-F. Chang, "Syntax, Semantics and Visual Information: A Conceptual Representation for Indexing", ADVENT Project Technical Report No. 1999-01, Columbia University, January, 1999.
[14] A. Jaimes, and S.-F. Chang, "The Visual Apprentice: A Multiple-level Learning and Classification Framework", ADVENT Project Technical Report No. 1999-02, January, 1999.
[15] P. Lipson, "Context and Configuration Based Scene Classification", PhD dissertation. MIT Electrical and Computer Science Department, September 1996.
[16] D.G. Lowe, *Perceptual Organization and Visual Recognition.* Kluwer Academic: Boston, 1985.
[17] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, New York, 1992.
[18] T. Minka, and R. Picard, "An Image Database Browser that Learns From User Interaction", Technical Report 365, MIT Media Laboratory and Modeling Group, 1996.
[19] T. Mitchell, *Machine Learning*. McGraw-Hill, New York, 1997.
[20] S. Paek, A.B. Benitez and S.-F. Chang, "Self-Describing Schemes for Interoperable MPEG-7 Content Descriptions", *Symposium on Electronic Imaging: Visual Communications and Image Processing*, 1999, IS&T/SPIE99. San Jose, CA, Jan. 1999.
[21] J.R. Smith, and S.-F. Chang, "Multi-stage Classification of Images from Features and Related Text", *Proc. Fourth DELOS workshop*, Pisa, Italy, August, 1997.
[22] M. Szummer and R.W. Picard, "Indoor-Outdoor Image Classification", *IEEE International Workshop on Content-based Access of Image and Video Databases*, in conjunction with ICCV'98. Bombay, India.
[23] T.F. Syeda-Mahmood, "Data and Model-Driven Selection Using Color Regions", *International Journal of Computer Vision*, 21(1/2), 9-36 (1997).
[24] A. Vailaya, A. Jain and H.J. Zhang, "On Image Classification: City vs. Landscape", *IEEE Workshop on Content-Based Access of Image and Video Libraries*, June 21, 1998, Santa Barbara, California.
[25] D. Zhong and S.-F. Chang, "Video Object Model and Segmentation for Content-Based Video Indexing", *IEEE Intern. Conf. on Circuits and Systems*, June, 1997, Hong Kong. (special session on Networked Multimedia Technology &Application).
[26] D. Zhong and S.-F. Chang, "AMOS: An Active System for Mpeg-4 Video Object Segmentation", *1998 International Conference on Image Processing*, October 4-7, 1998, Chicago, Illinois, USA
[27] A. W. Moore and M. S. Lee, "Efficient Algorithms for Minimizing Cross Validation Error", *Proceedings of the 11th International Conference on Machine Learning*, Morgan Kaufmann, 1994

.

Table 1: Classification performance results

Image Set	Image Set	Recall	Precision
Ships	A	94 %	70 %
Roses	A	94 %	75 %
Elephants	A	91 %	77 %
Cocktails	A	95 %	36 %
Skater	B	100 %	62 %
Faces	C	70 %	89 %

Image set A: images of ships, roses, elephants, and cocktails.
Training set: 30 of each class for a total of 120.
Testing set: Remaining 70 of each class, for a total of 280.

Image set B: the skater sequence contains 120 frames.
Training set: first 30 frames of the skater sequence.
Testing set: remaining 90 frames from skater sequence and the 400 images from set A, for a total of 490 images.

Image set C: news images from an internet newsgroup.
Training set: 70 news images that contain 86 faces in total (only faces used for training).
Testing set: 181 images with one face, 137 with two faces and 60 with no faces for a total of 378 images.

Internal *object-part* structural information was not used in these tests, but detailed results are provided in [14].
The results above suggest that the framework presented here is suitable for performing *visual classification* when the elements in the class present a strong consistency in terms of regions (e.g., elephants). In cases where that consistency is limited to a very small subset of the conceptual definition of a class (e.g., skater), we say that our approach is suitable for *detection- The Visual Apprentice* learns from the examples it is provided, therefore, the definition of skater in the system is limited to skaters with very similar visual characteristics to the one in the examples.

In other classes, such as cocktails, for instance, there is a wide variation in terms of the color, size and location of the regions. Our approach is not suitable in such cases.

6. CONCLUSIONS AND FUTURE WORK

We have presented a different approach to content-based retrieval in which distinct users can utilize an *object definition hierarchy* to define their own *visual classes.* Classes are defined in *The Visual Apprentice*, an implementation of our approach, which learns classifiers that can be used to automatically organize the contents of a database. Our novel framework for classification of visual information incorporates multiple fuzzy classifiers and learning techniques according to the levels of the *object definition hierarchy*: *(1) region, (2) perceptual, (3) object-part, (4) object* and *(5) scene.* The use of generic and task-specific classifiers is possible, and our approach is flexible since each user can have his own *visual classifiers*.

Future research directions include placing a feedback loop in the *The Visual Apprentice* so that learning is performed incrementally, allowing the user to place additional restrictions on the visual classes. This will require more sophisticated interface tools. In addition, we are working on integrating *visual classification* results with information provided by other media accompanying the visual information. In particular, we are working on using *The Visual Apprentice* to build classifiers for news images/video making use of text information. Motion information is also being included in our model and we plan to integrate the work from [26] to assist labeling of video objects. Our system forms part of Columbia's Mpeg-7 testbed [20].

4.2.4 Objects and scenes

The process described above for learning structural relationships is also used at this level, with the difference that each element in the *ARGs* corresponds to an *object-part* instead of a *perceptual-area.* Scenes, as structured sets of objects, can be given the same treatment.

5. APPLICATIONS AND PERFORMANCE

5.1 Applications

In this section we briefly discuss the visual domains in which our framework is suitable and possible applications of our techniques.

Strong visual consistency across images/videos of the same class can take on many forms: consistent motion, overall color, texture, etc. Our approach is based on regions, therefore, it is suitable for cases in which similar regions are obtained for elements in the same *visual class* (e.g., some animals, people, nature, etc.). Classes may consist of objects or scenes- figure 9 shows examples of how scene structure may be defined as a visual class in our framework.

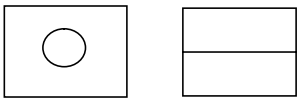

Figure 9: Scene structure can be defined in terms of regions. In the first case, for instance, the visual class can be defined as one having a flat background, independently of the regions that appear in the center.

The cases in which our framework can be applied to *classification* depend on the definition of the specific class. In general, *visual classes* can be built when elements of the class have similar regions- when that is not the case, we can refer to the process as *detection*. The skater of figure 1 is a good example for a *detection* application. A television viewer watching a program in which the skater appears, may want to build a classifier that will automatically filter other programs being transmitted that contain the same skater. This type of application can be very useful in such scenario and particularly in news domains because often a story is repeated by different sources (e.g. Dolly the sheep).

5.2 Performance

A good performance analysis of the implementation of our framework requires testing at each individual level of the hierarchy as well as for each individual classifier. This analysis is presented in detail in [14]. In this paper, however, since the focus is on the presentation of our framework, we only present a few cases in order to demonstrate the feasibility of this approach.

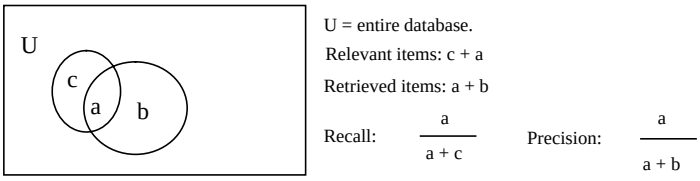

Figure 10: Graphical representation of performance measures used.

tuple (a_i, m_i), a_i is a *perceptual-area* and m_i its membership value for that class. The goal at this step is to find combinations of these that may correspond to *object-parts*.

An *object-part*, by definition, may or may not contain more than one *perceptual-area*. When only one *perceptual-area* is present, it is equivalent to the corresponding *object-part*, thus *perceptual-area* classification alone yields the result for the *object-part* (as is the case with the face *object-part* of the skater). In cases where more than one *perceptual-area* is present, spatial relationships between those areas must be considered. If they are ignored, two *object-parts* having similar *perceptual-areas*, but very distinct spatial layout would be incorrectly placed in the same class.

To represent the structural relationships between *perceptual-areas* within *object-parts*, we first build *Attributed Relational Graphs (ARG)* [5] based on the spatial relationships of figure 7. In an *ARG*, each node represents an *object* and an arc between two nodes represents a relationship between them. Instead of using a relational representation (e.g., *ARG*) directly, we convert each *ARG* to an attribute-value representation: objects in the *ARG* are ordered and a feature vector is generated, as explained in figure 8.

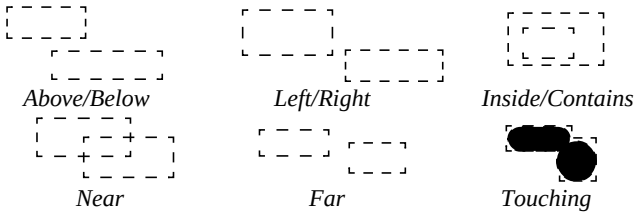

Figure 7: Structural relationships used in *The Visual Apprentice.* Bounding boxes are used, as seen above.

Attribute-value representations for structural relationships have been used in some learning systems [19]. One of the advantages of such transformation is that existing learning techniques that use feature vector representations can be applied. This avoids the difficult search in the space of more complex representations (e.g., Horn clauses). One of the disadvantages, however, is that the number of attributes can grow very quickly. We avoid this problem by using *ARGs* after *perceptual-areas* have been found- this is equivalent to having labeled objects in the nodes of the *ARGs* and a very small set of object attribute features (e.g., as opposed to *region* feature vectors). Another advantage of having labeled nodes (provided by our multi-stage classification framework) is that we do not deal with matching of graphs (*ARGs*) that contain unlabeled objects, which is a hard combinatorial problem.

A problem with the feature vector representation is that the number of nodes in the *ARGs* (in our case *perceptual-areas*) may vary between images/videos. Three possibilities to deal with this may be considered: (1) use set-valued attributes as in [2], (2) use wild card (unknown) attribute values when nodes are not present, and (3) build several representations for a given ARG [5]- i.e., if it contains n nodes, build separate feature vectors for n, n-1, ... 2 nodes. For figure 8, for example, feature vectors could be generated corresponding to the *ARGs* for (A,B,C,D), (A,B,C), (A,B), etc.

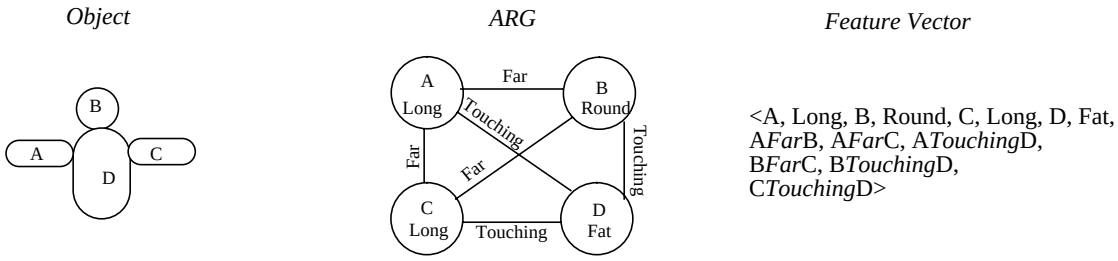

Figure 8: *Attributed Relational Graph (ARG)* and corresponding *feature vector* representation. In this simple case, every node has only two attributes: label and shape. In the feature vector, nodes and their attributes are listed first, and then their corresponding relationships.

Feature vectors obtained from the transformation are used by a decision tree [19] to learn the relationships for the given class. We use the closed-world assumption to generate negative examples for the *ARG* feature vectors. During classification, *ARGs* are generated for all possible sets of *perceptual-areas* found by the previous stage. Those feature vectors are then classified by the decision tree associated with the corresponding *object-part* class.

Using the *regions* in R_{op}, *groups* that may belong to *perceptual-areas* must be found by applying the corresponding *perceptual-area* classifier. The first step is to use a *region* classification algorithm analogous to the one in the previous section (with the difference that the classification function will produce a binary result), to obtain a new set of regions, $R_{opg} \subseteq R_{op}$ where each region in R_{opg} is likely to form part of the *perceptual-area* in question. Considering the skater of figure 1 again: *regions* that are likely to belong to the skater body will be in the set R_{op}. This set may include white, and blue *regions*, so the first step in finding a blue body *perceptual-area* will be selecting the blue *regions* from the set R_{op} and placing those in the set R_{opg}.

Since a *perceptual-area* by definition is an area of connected *pixels* in the image, we must find *groups* (i.e., adjoining regions) in the set R_{opg}. We apply a grouping function g (as defined in 4.1) to R_{opg}, to obtain a set $PG = \{g_1, g_2, .., g_n\}$ where every $g_i \in PG$ is a *group* of adjoining *regions* (e.g. *group* of blue *regions*). The goal then becomes to find the most likely *perceptual-area* candidates from the set PG. Since each element g_i, however, comprises a set of *regions* it is possible for a subgroup of g_i to be a better candidate for the *perceptual-area* class than g_i itself.

In this sense, we must perform a search over the space of possible *groups* of regions from the set *PG* to find the best possible ones (i.e., those that are more likely to be *perceptual-areas*). This can be treated as a classical search problem in Artificial Intelligence [12], and therefore, we can use heuristic techniques to reduce the search space. One of such techniques is Evolution Algorithms [17]. We treat each element g_i in the set PG as an individual in a population P (e.g., P=PG). 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 (figure 6):

1. Initialize population (P = R_{opg} set of *strong regions* that are candidates for the current *perceptual-area*).
2. Preliminary cross-over (PG = *groups* of adjoining *regions* from P).
3. Secondary cross-over (PS = *groups* formed by individuals from PG and adjoining weak *regions* from P), only population PS is kept.
4. Fitness function (*perceptual-area* classifier) used to evaluate fit individuals in current population.
5. Selected individuals are mutated.
6. Go to step 4.

As illustrated in figure 6, 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 as a *perceptual-area* candidate: at every generation step, each *group* is mutated, thus generating a new individual. A new individual's fitness in the population is measured by a *perceptual-area* classifier, thus the features for the new individual must be computed. Current features for *perceptual-areas* are the same as for *regions* and *perceptual-area* classifiers are lazy classifiers (similar to those in section 4.2.1).

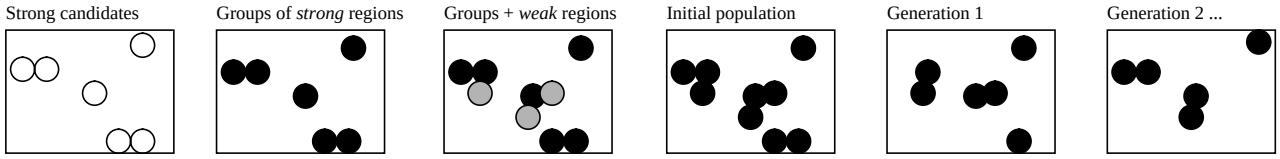

Figure 6: Evolution process for a set of regions. Grouping *weak* regions with existing groups of *strong* regions avoids forming groups of isolated weak regions. At every generation, individuals are mutated to form new ones and new features are computed. A fitness function is used to determine whether an individual survives into the next generation.

4.2.3 Object-part

The result of the previous step is a set P_g of *perceptual-area* candidates $P_g = \{(a_1, m_1), (a_2, m_2), ..., (a_n, m_n)\}$ where in each

The input to each lazy-classifier is an instance (feature vector) and the output a fuzzy-membership value for that instance. The current *region* feature vectors contain the following attributes [14]: color (average Hue, Saturation, Value), location (x, y center location, orientation), size (area), shape (aspect ratio, formfactor, roundness, compactness, extent) and texture (coarseness, contrast). Since performance of most learning algorithms is strongly affected by irrelevant features, for each lazy-classifier, the Branch and Bound algorithm is used for feature selection. Each classifier consists of the following:

- *Concept description:* set of instances that belong to the class (i.e., positive examples).
  CD = $\{t_1, t_2, ..., t_n\}$ where t_i is a training instance (e.g., *region feature vector*)

- *Similarity function:* computes similarity between the new instance (feature vector) and instances in the concept description (K-Nearest Neighbor, Weighted Euclidean Distance). Between an input instance *x* and a training instance *y*:

$$sim(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2 \times \omega_i^2}$$

  Where ω_i is a constant that weights the role of the i^{th} attribute in the similarity function, and y_i is the value of feature *i* of element *y* and *n* is the total number of features.

  The feature weight ω_i for each feature is obtained from the variance of that feature, measured independently of other features over the training set:

$$\omega_i = 1 - \sum_{j=1}^{m} (y_j - \mu)^2$$

  Where y_j is the value of feature *i* of the j^{th} element in the training set and μ is the mean value for that feature over the training set (which contains *m* elements).

- *Classification function:* uses the similarity function result to compute a fuzzy-membership value for the input, using the parameters h_t (hard boundary threshold) and s_t (soft boundary threshold), where $h_t < s_t$.

  if $sim_k(x) > s_t$ then $m_x = 0$, $s_x = 0$
  else if $sim_k(x) > h_t$ then $s_x = 0$, $m_x = sim_k(x)$
  else $s_x = 1$, $m_x = sim_k(x)$

  Where $sim_k(x)$ is the result of computing the similarity value of the input *x* with its *k* nearest neighbors in the training set; m_x is the membership value assigned to the input *x* and $s_x = 1$ if *x* is considered a *strong* member of the class and 0 if it is considered a *weak* member.

In summary, the similarity function *sim(x,y)* produces a value for each input instance, which represents how similar it is to the training examples. The classification function, which provides the output of the classifier, uses this information to determine the input instance's membership value in the class, thus producing a fuzzy-set when the classifier is applied to a set of instances. In addition, the classifier determines whether the input instance is a *strong* member or a *weak* one. This is useful when performing grouping, as will be seen in the next section.

Classification at this level produces for each *region* in the incoming image a value that determines the *region's* degree of membership in the current *region* class.

4.2.2 Perceptual grouping, search and evolution programs

The previous step 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 and s_i=0 if it is a *weak* member and 1 if it is a *strong* member. *Regions* from the original image whose classification produces a membership value of zero are not included in the set R_{op} (as defined in the classification function above).

the next section we describe the classification process in more detail and the specific classification approach used at each level in the current implementation of *The Visual Apprentice*.

4.2 Classification process

The first step in the classification of a new image/video is its automatic segmentation. This is done using the segmentation parameters provided by the user during the class definition and training phase.

Classification follows the bottom up order of the *object definition hierarchy*. In classes where more than one classifier is present at the same level, those classifiers are ordered from best to worst according to their performance, therefore setting the *Focus of Attention* for classification at each level. This is useful since in order for an element to pass a classification test, all of its components (i.e., descendants in the *hierarchy*) must exist, and it may be unnecessary to use a classifier at a given level if other classifiers at the same level have failed (e.g., a skater must contain two *object-parts;* if no face is found it is not necessary to search for a body *object-part*). Performance is computed by calculating error estimates for each classifier independently using leave-one-out error cross-validation [27] on the training data.

Once the input image/video has been segmented, a *region* level classifier is applied, according to the specific *focus of attention* for the given class. This yields a fuzzy-set of candidate *regions* that belong to a particular *object-part*. A *perceptual-area* classifier is then applied to that set of *regions: groups* of adjoining regions are found and passed to the evolution program, which finds the best combinations. The focus of attention is set again (at the *perceptual-area* level) and the process is repeated.

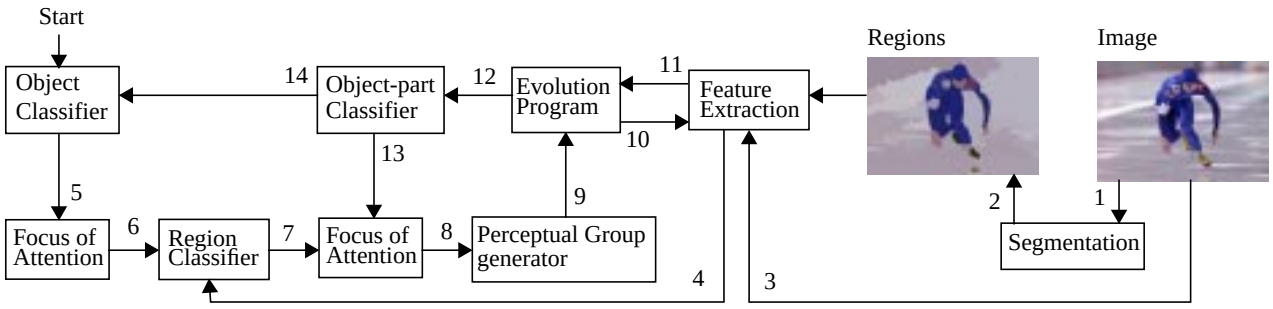

Figure 5: Overview of the classification process. Steps 1-4 are performed once and yield the features for the input image. First (steps 5,6), a region classifier corresponding to a an object-part is selected. Regions are classified (7) and a choice to find perceptual areas is made (8; e.g., blue). The evolution program extracts new features for the generated groups and performs a loop to find best possible groups (10, 11 repeat). Once those groups are found, if the object-part contains more perceptual areas, the process repeats at step 13. If not, the object part classifier yields a result to the object classifier. Again, the process repeats from step 5 for a new object-part if one was defined.

In the next sections we describe classification and grouping at each level in the order that it is performed: *(1) region, (2) perceptual, (3) object-part, (4) object* and *(5) scene.*

4.2.1 Region

Region classification serves as an initial selection process similar to the one in [23]. Current *region* classifiers are similar to IB1, a lazy learning algorithm [1] which is a variation of K-Nearest-Neighbor classification.

Lazy learning algorithms perform classification by combining their stored data (i.e., the training set) when an input occurs. This makes it possible to have dynamic models that can be changed by the user over time since the examples provided are not discarded after training (e.g., for the skater of figure 1, the user may, at a later time, change the features of the classifier so that it accommodates red skaters). In addition to providing this flexibility, Lazy-learners[1] have significant advantages over other learning algorithms when the target function is very complex [19].

1. Also referred to as memory-based or instance-based learners.

(e.g., *groups* of *regions*: prospective body *regions* are *grouped* and the best *groups* found by the corresponding *perceptual-area* classifier).

Using a binary classifier for a class *j* implies absolute certainty about the membership of the elements in the set *j*. In that case, when applying a grouping function *g* to the set *j*, we can have a guarantee that the best possible *groups* from *j* are in *S*. The use of fuzzy-classifiers, on the other hand, allows the grouping function *g* to select elements from *j* according to their degrees of membership in class *j* (rather than blindly) and improves the possibility for grouping functions to find the best *groups* in *j*.

Classification and grouping are performed in order according to the structure of our *object definition hierarchy* (figure 3). First, individual *regions* are classified 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*. Each fuzzy-classifier acts like a "black box", permitting the incorporation of specialized classifiers (e.g., face *object-part* classifier below replaced by a face recognition module) or generic classifiers (e.g., each classifier as a specific instance of a decision tree learning algorithm). It is important to note that the structure of figure 3 is not identical to the *object definition hierarchy* of figure 2; we have chosen to have only one *region* classifier as a descendant to each *object-part* classifier.

Classifiers interact across levels, but not within levels: a classifier C_{jl} for class *j* and level *l* is a function of the classifiers it uses from level *l-1*, but not of another classifier from level *l* (in figure 3, no horizontal connections exist between classifiers, thus face and body *object-part* classifiers, for instance, do not exchange information).

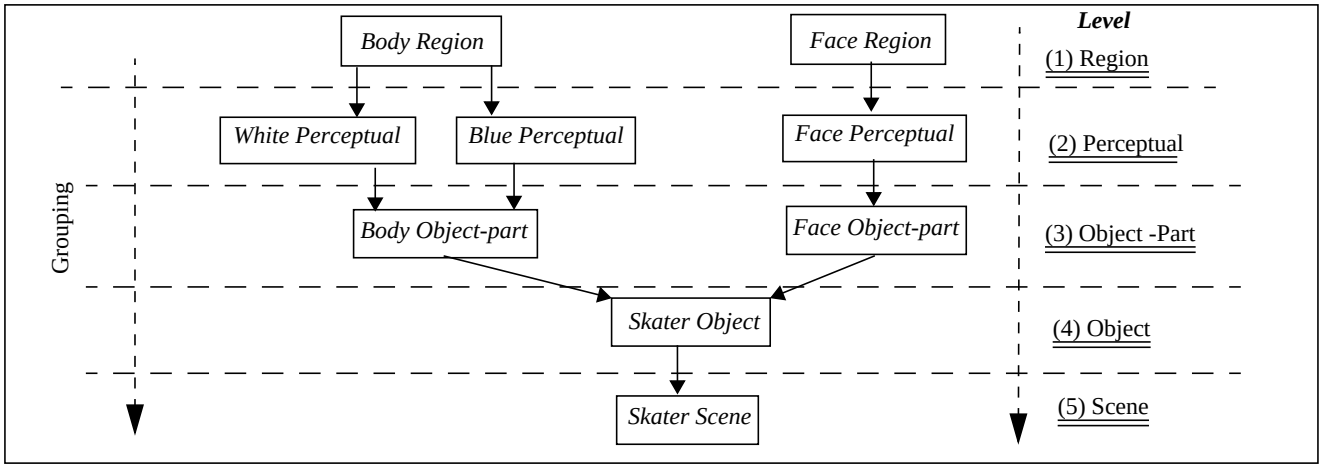

Figure 3: Multiple level classifiers make up models. Each classifier acts as a "black box" passing its results to the next level, thus allowing the incorporation of different learning and grouping strategies. This diagram corresponds to the skater of figure 1 as defined by observer B in section 3.1. Each box represents a classifier and arrows indicate the propagation of classification information.

As defined earlier, given a universe *U* and a set $j \in U$, a classifier $c_j(i)$ is a characteristic function for the set *j (*e.g., c_j is an equivalent way of defining *j*). Taking this into consideration, we can visualize the interaction between classifiers at different levels in terms of sets. Figure 4 is useful in interpreting the flow of information and in particular when reading section 4.2.

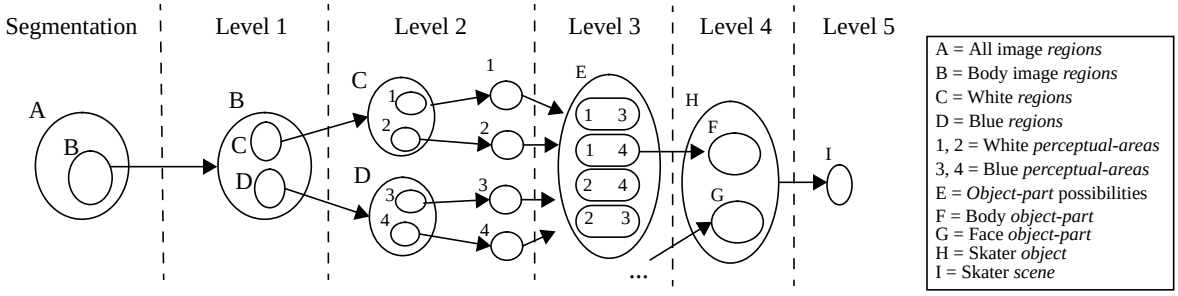

Figure 4: Classification and grouping can be seen in terms of sets. At each step, elements that pass a classifier are selected from a set and in other cases new sets are formed by obtaining *groups* of elements at a particular level.

In

an element of level *i-1*, unless they are equal *(e.g.* an *object* cannot be part of a *perceptual-area*; a face can be equal to a single perceptual area); (3) elements of the same level are disjoint (e.g. intersection of two elements of the same level = null; two *object-parts* cannot intersect); (4) *regions* do not contain subsets (i.e. *regions* are the basic units and cannot be separated); (5) No. elements level i <= No. elements at level i-1.

During training, the user performs the following tasks: (1) selection of the best segmentation[1] for the class; (2) creation of a *class-definition hierarchy* by defining the labels to be used from level 2 to level 5; (3) labeling of *regions* in each training image/video frame according to the *hierarchy*. As a result of user interaction, we obtain the following sets for a defined class *j*:

- Conceptual *object definition hierarchy* (e.g. figure 2): H_j.
- Set of segmentation parameters: $SEG_j = \{par_1, par_2, ..., par_n\}$
- Labeled Element Set: $LES_j = \{\{(e_{11}, l_{11}), (e_{12}, l_{12}), ..., (e_{1n}, l_{1n})\}, ..., \{(e_{k1}, l_{k1}), (e_{k2}, l_{k2}), ..., (e_{kp}, l_{kp})\}, ..., \{(e_{m1}, l_{m1}), ..., (e_{mq}, l_{mq})\}$ 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 it (e.g., (op_{31}, l_{31}) = (body *object-part*, body label)). Label level distinctions emphasize that labels must be different at different levels of the *hierarchy*.

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). For each element in the label 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 LES_j, we obtain a training set of positive examples for each class *j*:

- $TS_j = \{\{(fv_{11}, l_{11}), (fv_{12}, l_{12}), ..., (fv_{1n}, l_{1n})\}, ..., \{(fv_{k1}, l_{k1}), (fv_{k2}, l_{k2}), ..., (fv_{kp}, l_{kp})\}, ..., \{(fv_{m1}, l_{m1}), ..., fv_{mq}, l_{mq})\}$ 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 it (e.g., (op_{31}, l_{31}) = (body *object-part* feature vector, body label)).

Attributes for the *feature vectors* may vary across levels of the *hierarchy*. Structural relationships, for instance, are considered only between *perceptual-areas*, between *object-parts*, and between *objects. Feature vectors* at each level are discussed in the next section.

4. LEARNING, CLASSIFICATION AND GROUPING

Classification and grouping is performed at the five levels of the *hierarchy* defined above: *(1) region, (2) perceptual, (3) object-part, (4) object* and *(5) scene.* In this section, we describe the general characteristics of our classifiers and how they interact to perform grouping. We also describe in detail the classification process at every level of our framework.

4.1 Fuzzy-classification and grouping at multiple levels

Given a universe *U* of elements (e.g., *U* comprises nodes at a given level of the *object definition hierarchy*), 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 [6] 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).*

In addition to fuzzy-classification functions, we define a grouping function $g: U \rightarrow S$ where *S* is a set of disjoint subsets of *U*. We call every set s_i of *S* a *group*, for which any pair of elements in s_i are connected to each other, meaning that there is a spatial path between those two elements. In simpler terms, the function *g* returns *groups* of adjoining areas of the image

1. The best segmentation for a given class minimizes areas of internal regions that fall outside the boundaries of the objects of interest, while minimizing the number of regions inside those objects. Parameters depend on the segmentation algorithm in use.

(5) *Scene:* structured[1] set of *objects*.
(4) Object: structured set of adjoining *object-parts*.
(3) Object-part: structured set of *perceptual-areas*.
(2) *Perceptual-area:* set of contiguous and perceptually homogeneous *regions.*
(1) *Region:* set of connected [2]*pixels.*

Figure 2 illustrates a generic *object definition hierarchy* and a specific example for the skater image in figure 1, which we describe next.

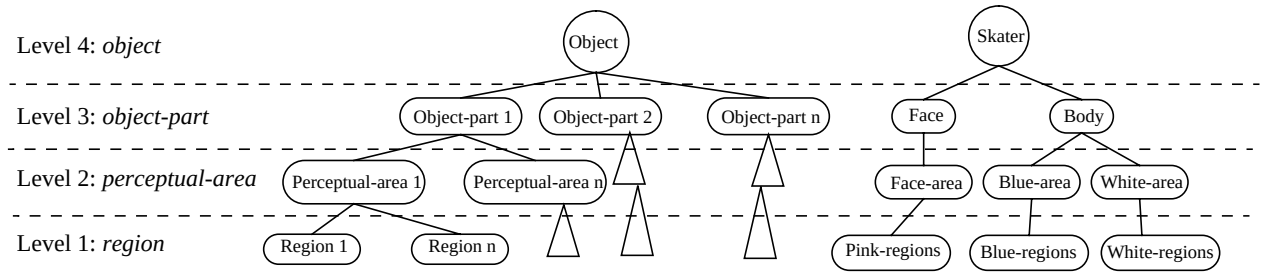

Figure 2: General *object definition hierarchy* and *hierarchy* for the skater of figure 1. Every node in the tree has a conceptual meaning (e.g., face), but also corresponds to a set of connected pixels in the image. As such, each set of pixels contains all of the descendants that correspond to its node (e.g., body contains blue and white areas, etc.). Sets at the same level are disjoint.

Observing figure 1 we can build the following *object definition hierarchy:*

(5) Scene: one *object* is present (a skater).
(4) *Object* skater*:* the division of an *object* into parts is subjective. Let us consider three observers and their divisions:
Observer A: *object* should not be divided into parts.
Observer B: *object-parts* are face and body.
Observer C: *object-parts* are face, body, and skates.

We continue our definition only for observer B:

(3) *Object-part* face: only one *perceptual-area* (entire face).
Object-part body: body of the skater can be divided into two main *perceptual-areas* (blue body-area and white arm-area)
(2) *Perceptual-area* blue body: set of blue *regions*.
Perceptual-area white arm: set of white *regions*.
(1) *Region* blue: set of connected blue *pixels*.
Region white: set of connected white *pixels.*

As the previous example suggests, in general, the way in which *definition hierarchies* are built is subjective and may vary between users. Our framework accommodates this subjectivity: users can build different models based on their individual interests. There are, however, some restrictions on the way a hierarchy can be built. These are discussed next.

Every node *e* in our *object-definition-hierarchy* has a conceptual interpretation (e.g., face). In addition, each node *e* corresponds to a set of connected pixels in the image/video. We can state the restrictions for building valid *definition-hierarchies* as follows: (1) an element of level *i* ($i \neq 5$) is a subset of only one element 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) an element of level *i* cannot be a subset of

1. The word structured is used only to emphasize the importance of spatial relationships between elements in the particular set. Spatial locations of elements may be part of their attributes.
2. Regions, which are at the lowest level of the hierarchy, constitute the basic units in our framework and can be extracted using any segmentation algorithm based on low-level features such as color, texture, or motion. We currently use the implementation from [25] for image/video segmentation.

2. GENERAL APPROACH

Given the importance of objects and their parts in classification of visual information [13], we base our framework on an *object-definition hierarchy* consisting of the following levels: (1) *region*; (2) *perceptual*; (3) *object-part*; (4) *object* and (5) *scene*.

First, the user defines the *visual class*. This is done by creating labels for *objects* and their *parts* (e.g., the *scene* class skater in figure 1 contains one *object*, with *object-parts* face, body and skates). Then, the user provides training examples for the class: for each image/video example, *regions* are labeled according to the class definition. The labeled regions from all of the training examples are used to compute the training set: features (color, texture, shape, spatial relationships, etc.) for each level are automatically extracted and stored in a database. This database is used for training by multiple learning algorithms to yield a set of fuzzy-classifiers at different levels (e.g., *region, object-part*).

Automatic classification is performed by combining the learned classifiers and using grouping strategies at the levels listed above. *Regions* are classified first and combined to obtain *perceptual-areas* (i.e. areas that are visually homogeneous; e.g., blue skater-body area of figure 1) which are used by *object-part* classifiers. *Object-parts*, in turn, are combined and passed to *object* classifiers.

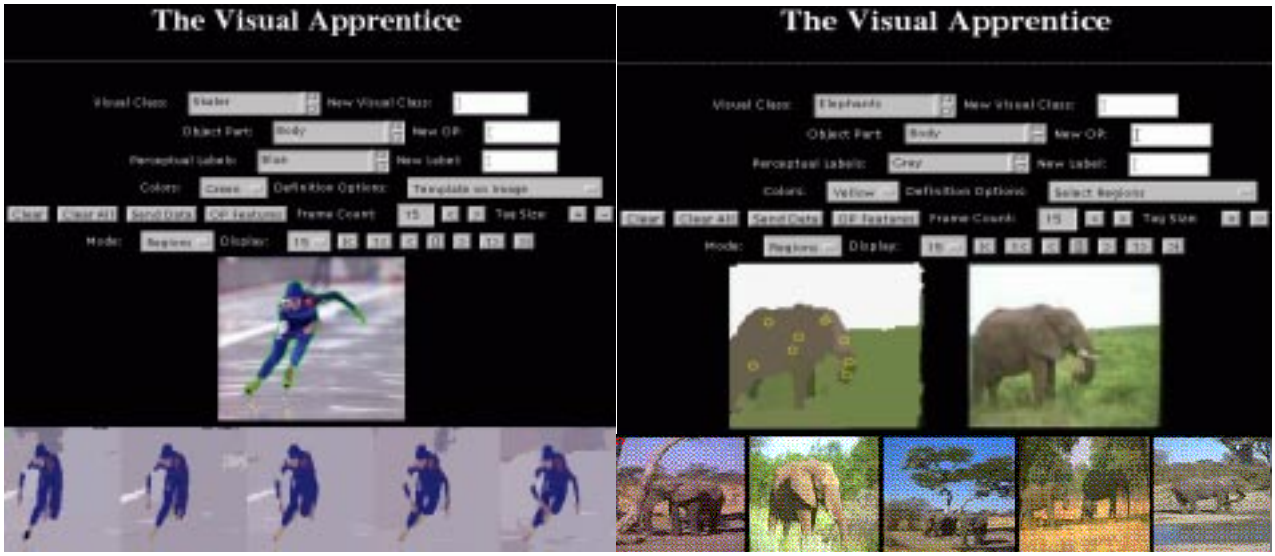

Figure 1: During training, the user defines a class and labels regions in the training set according to the definition. On the left is a *visual class* for which the user is interested in *detection* of a skater with similar visual features. On the right, the user's interest is in building a *classifier* for elephants (there is strong *visual* consistency in the elephants class).

In the next section, we describe the components present in a class definition and the results of training.

3. CLASS DEFINITIONS AND TRAINING

3.1 Grouping and perceptual organization

Studies in cognition and human vision have shown that during visual recognition, humans perform grouping of features at different levels [16][4]. 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 which we perceive categorically (i.e., as a single visual *group*). The best example of categorical perception is color [23]: when observing an *object* and asked how many colors it contains, we "group" different shades of the same color. An elephant's ear, for instance, constitutes one *perceptual-area* for an average observer- there is no interest in differentiating between distinct shades of gray. We base our framework on classification and grouping at the following levels: (1) *region*; (2) *perceptual*; (3) *object-part*; (4) *object* and (5) *scene*. We define an *object definition hierarchy* as follows:

appearance). In this scenario classification can be ambiguous, unless the user and the system share the same meaning for a given class (e.g., class "stars")- the user's definition of similarity must match the system's definition. In practice, this is difficult to achieve in a general application, given that different users have different criteria for similarity and possibly different meanings associated with each class.

Existing approaches in a priori classification can be mainly divided into two groups: those that use specialized algorithms [10][11][24][22] for automatic classification and those that rely on manual classification. In both cases, the lack of flexibility is evident: creation of new specialized classifiers must be programed by an expert and manual based classification is expensive. Another limitation of most of these approaches is that all users are presented with the same classifications and the initial classes are chosen subjectively by the designers of the specialized algorithms or pre-defined classes. This is particularly problematic with images/video since there are practically unlimited ways of indexing visual information and classification can be very subjective, varying across users or for a single user over time [13][18].

In this paper, we present a different approach towards content-based retrieval and a novel framework for classification[1] of visual information. Users define *visual classes* based on objects and classifiers are learned automatically as each user provides examples. Our approach is flexible since: (1) each user can have his own set of classifiers (system class definitions and similarity criteria match those provided by the user); (2) classifiers can be changed over time and (3) once a user's class has been learned, database contents can be classified automatically.

Our novel framework for classification of visual information is based on a hierarchical decomposition of the image with the following levels: (1) *region*; (2) *perceptual*; (3) *object-part*; (4) *object* and (5) *scene.* We present *The Visual Apprentice*, an implementation of this framework for still images and video that uses a combination of lazy-learning, decision trees and evolution programs for fuzzy-classification and grouping.

In the framework presented, we are concerned with the problem of *visual* rather than *conceptual* classification. This distinction is of great importance since, as mentioned above, a semantic class carries information at different levels. Our interest is in the *visual* appearance of *objects* and in particular, in letting users define classes in those terms. It is clear that in some cases (e.g., elephants) the amount of *visual* variation for objects in the same class is smaller than in others (i.e. house). We use this criterion to differentiate between *classification* and *detection*.

1.1 Related work

This approach to content-based retrieval differs significantly from previous techniques (QBIC, Photobook, WebSEEk, Virage) [8] that do not attempt content classification based on visual features, and from others that use specialized algorithms (e.g., [11]'s face detection). It also differs from the work in [9] that is based on the formulation of sketch queries and from other approaches in which images are classified using global low-level features [22][24].

Our framework for classification of visual information differs from other approaches based on regions [21][15][7] that perform classification based on global region configuration, and from others [18] that neither use spatial relationships nor allow definition of complex objects [3]. Our approach is different from [10] in the following: 1. users construct their own models (rather than having a fixed set). 2. No assumption is made about elements of the model (regions have arbitrary shapes, cylinder-like "primitives" not assumed). 3. Generic region extraction rather than only specialized (like a skin filter). 4. Fuzzy classification paradigm rather than binary. In addition, our approach is different from other model-based techniques such as [23] that use single features for region extraction and from others that use 3D object models.

1.2. Outline

In section 2, we give a general overview. In section 3, we present an *object definition hierarchy* and describe the training phase. Section 4 describes the classifiers used and their interaction in detail. In section 5, applications of our framework and performance are discussed. Conclusions follow in section 6.

1. In some cases, we will use the word *detection*. *Classification* assigns an object to a category, whereas *detection* determines the presence of an object: *detection* will be used to refer to objects that have a rather "unique" visual appearance or that belong to classes in which there is great variation (e.g., house).

Model-Based Classification of Visual Information for Content-Based Retrieval

Alejandro Jaimes and Shih-Fu Chang[1]

Image and Advanced TV Laboratory, Department of Electrical Engineering
Columbia University, 1312 SW Mudd Building Mailcode 4712 Box F8
New York, N.Y. 10027

ABSTRACT

Most existing approaches to content-based retrieval rely on query by example or user sketch based on low-level features; these are not suitable for *semantic* (object level) distinctions. In other approaches, information is classified according to a pre-defined set of classes and classification is either performed manually or using class-specific algorithms. Most of these systems lack flexibility: the user does not have the ability to define or change the classes, and new classification schemes require implementation of new class-specific algorithms and/or the input of an expert. In this paper, we present a different approach to content-based retrieval and a novel framework for classification of visual information in which (1) users define their own *visual classes* and classifiers are learned automatically; and (2) multiple fuzzy-classifiers and machine learning techniques are combined for automatic classification at multiple levels (*region, perceptual, object-part, object* and *scene*). We present *The Visual Apprentice*, an implementation of our framework for still images and video that uses a combination of lazy-learning, decision trees, and evolution programs for classification and grouping. Our system is flexible in that models can be changed by users over time, different types of classifiers are combined, and user-model definitions can be applied to *object* and *scene* structure classification. Special emphasis is placed on the difference between *semantic* and *visual classes*, and between *classification* and *detection*. Examples and results are presented to demonstrate the applicability of our approach to perform *visual classification* and *detection*.

Keywords: content-based retrieval, image classification, video classification, model-based classification, detection, machine learning.

1. INTRODUCTION

Digital image databases have been growing in size and popularity. Recently, there have been many efforts to support searching and browsing based on the visual content of images/videos [8]. Existing approaches to content-based retrieval of visual information can be divided into two groups: those in which users perform queries by example or user sketch, and those in which a priori classification of the information is performed. Some systems include both, allowing query by sketch or similarity within categories.

Similarity and query by sketch techniques are based on low-level features (color, texture, etc.), thus concentrating on the form of the visual information (*syntax:* color, texture, etc.) rather than on the content (*semantics:* objects). Users, however, are generally interested first in semantics and then in syntax- the query "show me all the images that contain green apples" is more likely than the query "show me all the green areas that are apples". The focus is first on the objects depicted and then on their specific form. Low-level features are unsuitable at the semantic level: meaningful queries by sketch are difficult to formulate, and similarity queries do not express semantic level distinctions.

A priori classification, on the other hand, can provide a good general organization of the contents of the database in terms of semantics. For instance, the pre-defined scene class "man" carries general information at different levels [13]: *conceptual* (e.g., definition of man in the dictionary), *physical* (size, weight, texture) and *visual* (hair color, clothing), among others. Some of this information is explicit (e.g., man vs. woman), but a lot of it is implicit or undefined (e.g., actual physical

1. E-mail: {ajaimes, sfchang}@ee.columbia.edu WWW: http://www.ee.columbia.edu/{~ajaimes, ~sfchang}