Eurographics Workshop on 3D Object Retrieval (2008)
I. Pratikakis and T. Theoharis (Editors)

# A Neurofuzzy Approach to Active Learning based Annotation Propagation for 3D Object Databases

Michalis Lazaridis and Petros Daras

Informatics and Telematics Institute,
Centre for Research and Technology Hellas,
Thessaloniki, Greece,
GR 57001, PO BOX 60361

**Abstract**

*Most existing Content-based Information Retrieval (CBIR) systems using semantic annotation, either annotate all the objects in a database (full annotation) or a manually selected subset (partial annotation) in order to increase the system's performance. As databases become larger, the manual effort needed for full annotation becomes unaffordable. In this paper, a fully automatic framework for partial annotation and annotation propagation, applied to 3D content, is presented. A part of the available 3D objects is automatically selected for manually annotation, based on their "information content". For the non-annotated objects, the annotation is automatically propagated using a neurofuzzy model, which is trained during the manual annotation process and takes into account the information hidden into the already annotated objects. Experimental results show that the proposed method is effective, fast and robust to outliers. The framework can be seen as another step towards bridging the semantic gap between low-level geometric characteristics (content) and intuitive semantics (context).*

Categories and Subject Descriptors (according to ACM CCS): I.5.2 [Pattern Recognition]: Design Methodology

## 1. Introduction

During the last years significant progress has been achieved to all steps of the media chain, concerning the media creation, storage, transmission and general use of big quantities of multimedia data, such as sound, video, texts, drawings and 3D objects and scenes. Among these, 3D data is of particular interest because of the existence of new, improved digitization hardware tools, more efficient 3D reconstruction techniques and also their exploitation prospects in the scientific and industrial sectors, in fields as object recognition, scene analysis, bioinformatics, medicine, etc.

Also, the need for knowledge exploitation and reuse in modern multimedia database systems creates an imperative need for retrieval of objects semantically similar to some query objects that the user provides. Most of the methods that have been presented in the literature are focused on the improvement of the geometrical feature extraction process (descriptor extraction). Despite that fact, the awareness that no geometrical method can provide enough discriminative power has led to the research on other preprocessing techniques for classification and improved retrieval performance of multimedia content. The more powerful technique in this area is the use of semantic annotation. The annotation of database objects lies on the "attachment" of information on each object. This information can be constructional (e.g. designer information or date of scanning, in case of 3D models), structural (e.g. geometrical descriptor vector) or semantical (e.g. vector of probabilities, where each component expresses the probability that the object has a predetermined attribute). An annotation example could be the following: If the attributes of interest for the characterization of an object are [is it an animal? is it a mammal? does it have four legs?], then a probability vector with semantic information could be [1 1 0], which could partially describe a dolphin.

In traditional database systems, the user-annotator manually annotates all database objects, a work that requires significant labour. Database objects are presented to the annotator, one after the other, and s/he decides whether the object possesses or not a specific attribute. In modern database systems, where continuous renewal and management of contin-

uously increasing volume of information is crucial, manually annotation becomes non-functional. The process of annotation propagation focuses on this problem. The key question to better understand the nature of annotation propagation is: "how can we automatically expand annotations of certain already manually annotated objects to other objects that have the same or similar attributes without presenting them to the user for manual annotation?".

The scientific field of annotation propagation is an open research topic, since only a few of the proposed methods offer an acceptable and realistic solution to the problem. In the following paragraphs, annotation propagation methods that have been introduced by the scientific community are presented. Firstly, the methods that define object classes and propagate the probability for each object to belong to a concrete class, and secondly methods that propagate keywords, which are attached to the objects and describe their attributes.

Annotation Propagation and Relevance Feedback methods are usually used in combination. A unified frame for log-based feedback and classification is presented in [LJH06]. When feedback log data is unavailable, the log-based relevance feedback algorithm behaves exactly like a regular relevance feedback algorithm, which learns the correlation between low-level features and user's subjectivity. When feedback log data is available, the algorithm computes such a correlation using both the feedback log data and the online feedback from the users. In [HMK*02], a method is presented, which uses spectral methods to infer a semantic space from user's relevance feedback, so that the system will gradually improve its retrieval performance through accumulated user interactions. In addition to the long-term learning process, the traditional approach to query refinement using relevance feedback as a short-term learning process, is modeled. The proposed short and long-term learning frameworks have been integrated into an image retrieval system. In [ZC02], an active learning framework for content-based information retrieval is proposed. For each object in the database, a list of probabilities is maintained, each indicating the probability of this object having one of the attributes. Knowledge gain is defined to determine, among the objects that have not been annotated, which one the system is the most uncertain of. The system then presents it as the next sample to the annotator to which it is assigned attributes. During retrieval, the list of probabilities serves as an additional feature vector for calculating the semantic distance between two objects, or between the user query and an object in the database. In [ZS01], a classification-based, keyword propagation method is presented. The proposed framework consists of an image database that links images to semantic annotations, a similarity measure that integrates both semantic features and image features, and a machine learning algorithm to iteratively update the semantic network and to improve the system's performance over time. The semantic network is represented by a set of keywords having links to the images in the database. Weights are assigned to each individual link. The degree of relevance of the keywords to the associated images' semantic content is represented as the weight on each link. While the user is interacting with the system by providing feedbacks in a query session, a progressive learning process is activated to propagate the keyword annotation from the labeled images to un-labeled images so that more and more images are implicitly labeled by keywords. In [JM04], the use of the maximum entropy approach for the task of automatic image annotation is proposed. Maximum entropy allows one to predict the probability of a label in the test data, when labeled training data is available. The technique allows for the effective capturing of relationships between features. Features are computed over rectangular regions of the images, generated by partitioning the image into a grid. The regions are clustered across images. These clusters are called visterms (visual terms) to acknowledge that they are similar to terms in language. In [SS03], an attempt is made to propagate semantics of the annotations, by using WordNet, a lexicographic arrangement of words, and low-level features extracted from the images. The hierarchical organization of WordNet leads to the concept of implication/likelihood among words.

In the present paper a new framework for annotation and annotation propagation in 3D model databases is proposed, based on propagation of probabilities through neurofuzzy controllers, using a combination of low-level geometric and high-level semantic information. The system automatically selects the most informative training examples and presents them to the annotator for manual annotation. These examples serve as training data for the neurofuzzy controllers to learn the relationship between the low-level information and the attributes that an object has. The proposed framework is a complete and efficient solution for the automatic annotation of 3D databases.

The rest of the paper is organized as follows: In the second section, the concept of the proposed framework is introduced. Experimental results are given in the third section. In the fourth section, conclusions are drawn based on the results and finally, limitations of the concept and future work are presented in the fifth section.

## 2. Concept

The central idea behind the proposed framework is the training of the annotation system, using the information given from the user during the manual annotation, so that it can be later used for the automatic annotation. The system operates, therefore, in two distinct modes: The *training mode* and the *on-line mode*. Before describing the two modes, some information is given about the features (geometrical and semantic) used and also the operating principles of the neurofuzzy controller.

### 2.1. Features

Every 3D object $O_i, i \in \{1,...,N\}$, where N is the number of 3D objects in the database, is represented by two vectors: The first one is the feature (descriptor) vector, $G_i = \langle g_{i1},...,g_{iT}\rangle$, where T is the number of the features, which expresses the low-level geometrical features extracted from the object; the second one is the probability vector, $P_i = \langle p_{i1},...,p_{iK}\rangle$, where K is the number of all possible attributes in the database, consisting of the probabilities $p_{ik}$ for an object i to have a specific attribute $A_k, k \in \{1,...,K\}$. Every database object belongs to certain categories or has certain attributes. The attributes form a tree structure with multiple levels. Each node (leaf or internal) represents an attribute. The root node can be seen as a generic entity attribute that all objects have. Attributes that are closer to the root are more general so that when an object has a specific attribute it also has its parent attribute (inheritance). In the current implementation it is also assumed that an object can have maximum one attribute per attribute tree level. An example of a two-level attribute tree structure can be seen in Figure 1.

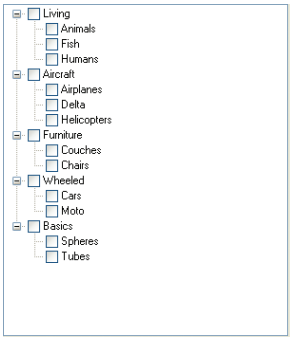


**Figure 1:** *Attribute tree structure with two levels*

#### 2.1.1. Low-Level Features and Geometrical Distance

3D-objects are pre-processed to obtain a so-called low-level feature vector which is a numerical vector representing an abstraction of the 3D object.

The geometrical distance between two objects $O_i$ and $O_j$ is defined as the Minkowski distance ($L_n$) between the descriptor vectors $G_i$ and $G_j$ of the objects, as follows:

$$d_g(i,j) = \sqrt[n]{\sum_{t=1}^{T} |g_{it} - g_{jt}|^n} \tag{1}$$

where $n \in \mathbb{N}_+, t \in \{1,...,T\}$.

The distances can be normalized and converted to similarities as follows:

$$similarity(i,j) = 1 - \frac{distance(i,j)}{distance_{max}} \tag{2}$$

where $distance_{max}$ is the maximum distance between two database objects.

#### 2.1.2. Semantic Features and Semantic Distance

The semantic features for each object i are the elements of the probability vector $P_i$.

The semantic distance between two objects $O_i$ and $O_j$ is defined as:

$$d_s(i,j) = \sum_{k=1}^{K} ws_k \left[p_{ik}(1-p_{jk}) + p_{jk}(1-p_{ik})\right] \tag{3}$$

where $ws_k$ is the semantic weight of attribute $A_k$, $k \in \{1,...,K\}$. The semantic weights of attributes depend on the level on which the specific attribute lies in the attribute tree and can be calculated as follows:

$$ws_k = \alpha^{l_k - 1}, \alpha \in [0,1] \subset \mathbb{R} \tag{4}$$

where $l_k \in \mathbb{N}^+$ is the attribute level.

The overall distance between two models can be calculated as the weighted sum of the geometrical and the semantic distance between the models:

$$d(i,j) = w_g \cdot d_g(i,j) + w_s \cdot d_s(i,j) \tag{5}$$

with $w_g + w_s = 1$.

### 2.2. Neurofuzzy Controllers

A neurofuzzy controller is a neural network based fuzzy system, capable of learning its own internal parameters through the use of training examples. Neurofuzzy systems combine the learning capabilities of neural networks with the transparency of Fuzzy systems ( [Jan93], [Cal99], [Kra03], [CP04], [KBLP07]). In Figure 2 the distinct steps of a Fuzzy

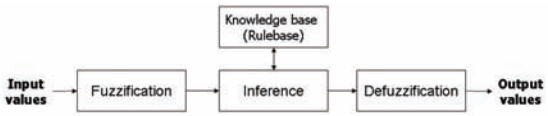


**Figure 2:** *Distinct steps of a Fuzzy Logic system*

Logic system are shown. In the fuzzification step, a qualitative description of the numerical input variables through linguistic variables takes place. Every linguistic variable's domain is covered by fuzzy sets, called linguistic terms. A membership function, defined for every linguistic term, estimates a membership value for every numerical input value. The knowledge base (rulebase) encodes the Expert knowledge over the system with fuzzy rules in the following form:

IF x is X AND y is Y THEN z is Z

where x and y are the input variables, z is the output variable, X, Y and Z are terms of the corresponding linguistic variables.

By inference based on the rulebase, a qualitative output value is calculated, in the form of a linguistic variable. Finally, during the defuzzification, the linguistic output variables are mapped to exact numerical values.

The fuzzy logic steps of a neurofuzzy controller are implemented as a multi-level structure of coupled neurons. Back-propagation of the error (difference between actual and target value) allows for the adjustment of the neural weights and the fuzzy logic parameters during the training mode. In order to achieve the latter the activation function of the neurons must be differentiable, thus various changes to the classic Fuzzy System are needed.

In Figure 3, a neural network for implementing a Fuzzy Logic system is shown. The network has a five layer structure. The neurons of the first layer represent the input variables and simply propagate the input values to the next layer. The second layer is the fuzzification layer. The third and fourth layers implement the rulebase and inference mechanism of the Fuzzy Logic System. The fifth layer represents the defuzzification. Therefore, regarding the i-th linguistic output variable:

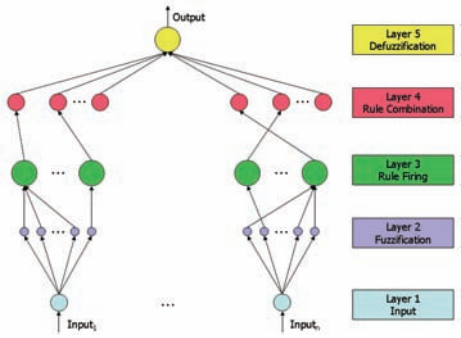


**Figure 3:** *Neurofuzzy model layers*

$$y_i = \frac{\sum_{j=1}^{n} m_{ij} x_j}{\sum_{j=1}^{n} x_j} \tag{6}$$

where $m_{ij}$ are the weights of this layer (abscissas of the singletons), which are adjusted during the training and $x_j$ are the outputs of the fourth layer.

The goal of training the network is to find a set of weights such that the error between the desired output and the actual output is minimized. The Delta rule ( [RHW86]) is based on the gradient descent algorithm. The error measurement is defined as:

$$E = \frac{1}{2}(t-o)^2 \tag{7}$$

where $t$ is the desired (target) output and $o$ the real output for the actual session. E is positive and becomes smaller as the performance of the network becomes higher. The gradient descent algorithm proposes the change of the weights $w_i$ by an amount $\Delta w_i$, proportional to the gradient of E at the present location:

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} \tag{8}$$

where $\eta$ is the learning rate. A trained neurofuzzy model is able to describe well the input-output mapping for any arbitrary non-linear function.

### 2.3. Training mode

In Figure 4, the components that are active during the training mode are shown. There are three main components

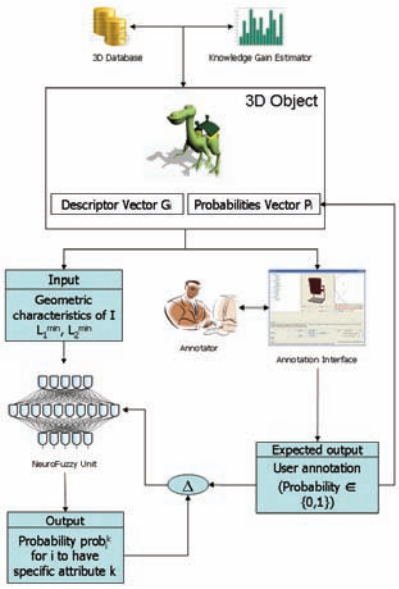


**Figure 4:** *Training mode*

which are active during the training mode:

- The NeuroFuzzy Unit
- The Knowledge Gain Estimator
- The Annotation Interface

In the next sections each component is explained in detail.

#### 2.3.1. The NeuroFuzzy Unit

The Neurofuzzy Unit serves as a classifier. A neurofuzzy controller is generated for each attribute of the attribute tree.

The objective of each neurofuzzy controller is to estimate the probability for an object to have a specific attribute by comparing its low-level features with those of other objects known to have the attribute.

Each neurofuzzy controller $NF_k$, dedicated to an attribute $k$, is a two input-one output system. The input variables are:

$L_1^{min}$ : The $L_1$ (Manhattan) distance of object i to its closest neighbor with the attribute $k$.

$L_2^{min}$ : The $L_2$ (Euclidean) distance of object i to its closest neighbor with the attribute $k$.

The output variable $prob_i^k$ represents the probability $p_{ik}$ of object i to have the attribute $k$.

For each newly annotated object i, all K neurofuzzy controllers are trained with the above mentioned distances from the corresponding sets of the already annotated objects as input. This implies that one annotated object per attribute must be available during system's initialization. For object $i \in \{1,...,N\}$ and a specific attribute $A_k$ with $k \in \{1,...,K\}$, the target output of the corresponding neurofuzzy controller is computed as:

$$t_{ik} = \begin{cases} 1.0 & \text{if object i has the attribute } A_k \\ 0.0 & \text{otherwise} \end{cases} \quad (9)$$

The success of the neurofuzzy controllers depends on the assumption that objects close to each other in the geometrical feature space are considered semantically similar up to a degree.

The use of other classification learning techniques is also possible. The advantages of the neurofuzzy approach are readability of the acquired knowledge and great performance, which was essential for the real time operation.

### 2.3.2. The Knowledge Gain Estimator

It is crucial for the proposed annotation system and especially for its learning ability, to choose the "right" training examples for manual annotation. The procedure of picking the objects that contain the most valuable information for training, is referred to as *Active Learning* ( [FSST93], [CMM*00]). The objective of such an action is to reduce the amount of data (and thus the required time) that is necessary for the system to be optimally trained, which is, in our case, the number of manually annotated objects. The criterion for selection of the examples is, therefore, the maximization of the knowledge that will be gained through the selection of an object for manual annotation. The knowledge gain ($KG(i)$) for an object i can be estimated as follows:

$$KG(i) = \frac{1}{\min\limits_{1 \le l \le L} \{ \max\limits_{level(k)=l} p_{ik} \}} \quad (10)$$

where $level(k)$ is the level of attribute $k$ in the attribute tree and $L$ the number of levels. That means that objects with a low dominant probability per attribute level are favoured.

### 2.3.3. The Annotation Interface

The Annotation Interface (Figure 5) provides annotation, annotation propagation and annotation validation capabilities. It consists of four main frames:

- The attribute tree area, where the attribute tree structure is shown, with all its attributes in a clickable form. The annotator can check/uncheck the corresponding attributes.
- The screenshot area, where a 2D screenshot of the example object is shown.
- The results area, where results and statistics concerning the annotation process are presented.
- The controls area, for initializing and controlling the system, as well as setting various system parameters.

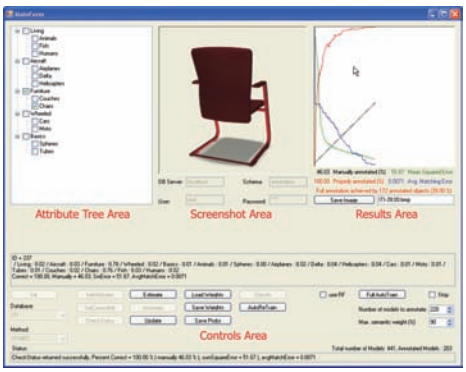


**Figure 5:** *Annotation Interface*

During an annotation session, the knowledge estimator selects a non-annotated object for manual annotation. A screenshot of the object is presented to the annotator along with the attribute tree structure, where s/he can select by checking the appropriate checkboxes which attributes the object has. For all checked attributes the corresponding value of the probability vector is set to 1.0. For all other attributes the probability is set to 0.0. The user then applies the annotation and the neurofuzzy unit recomputes all the probabilities of all other non-annotated objects, based on the updated annotated set. This procedure is repeated until all database objects are annotated or until the user breaks up.

## 2.4. On-line mode

As depicted in Figure 6, during the on-line mode, the trained Neurofuzzy Unit classifies the non-annotated database objects without external help. As in training mode, for every non-annotated object $i$, whose probability vector needs to be estimated, all K neurofuzzy controllers are involved, with the set of the already annotated objects as input. The object is considered to have a specific attribute if the corresponding probability is the highest among the probabilities of all other attributes of the same level (dominant attribute). The resulting attributes can then serve as suggested values if the annotator wishes to continue with the manual annotation, or they are stored directly in the database.

## 3. Experimental Results

The annotation propagation framework was tested on two 3D model databases:

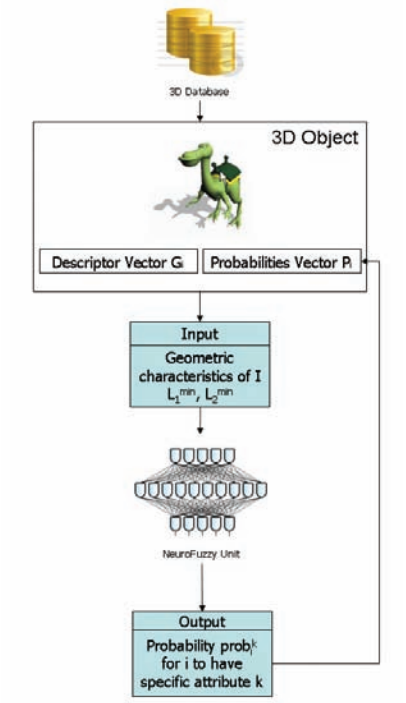


**Figure 6:** *On-line mode*

- The ITI database (http://www.iti.gr) with 441 models and 17 attributes
- The VICTORY database (http://www.victory-eu.org) with 164 models and 16 attributes

The geometrical descriptors used, were produced through the Spherical Trace Transform with Krawtchouk moments as initial functional [DPA*07]. The dimension of the geometrical descriptor space is 4320.

The descriptor vectors were normalized into the interval $[0,1]$ using the following scheme:

$$norm(\langle g_{i1},...,g_{iT}\rangle) = \left\langle \frac{g_{i1}}{\max\limits_{n\in\{1,...,N\}} g_{n1}},...,\frac{g_{iT}}{\max\limits_{n\in\{1,...,N\}} g_{nT}} \right\rangle \tag{11}$$

for object $i$, where $T$ is the number of the geometric features (4320) and $N$ the number of database models.

The geometrical distances were also normalized into the interval $[0,100]$.

In Table 1 the linguistic terms for both input linguistic variables are shown: The weights of the fifth layer were initialized with the value 0.5.

In order for the weights to reach rapidly the region around their optimal values and then just finetune in that region, the learning rate $\eta$ for the gradient descent algorithm was set

**Table 1:** *Linguistic terms of input variables*

| Variable | Term | Mean | Spread |
|---|---|---|---|
| $L_1/L_2$ | Very Close | 5.0 | 4.0 |
| $L_1/L_2$ | Close | 15.0 | 4.0 |
| $L_1/L_2$ | Medium | 25.0 | 4.0 |
| $L_1/L_2$ | Far | 35.0 | 4.0 |
| $L_1/L_2$ | Very Far | 45.0 | 4.0 |

variable, using the following scheme:

$$\eta = \max\{\eta_{max} - \frac{r_{annot}}{\eta_{div}}, \eta_{min}\} \tag{12}$$

where

$$r_{annot} = \frac{\text{number of annotated objects}}{\text{number of database objects}} \tag{13}$$

Experimentally, the following values were selected: $\eta_{max} = 0.2$, $\eta_{min} = 0.001$, $\eta_{div} = 5$.

Two metrics were used in order to observe the system's performance and state:

- the Mean Squared Error (MSE) and
- the Average Matching Error (AME).

The *Mean Squared Error* of an estimator is one of many ways to quantify the amount by which an estimator differs from the true value of the quantity being estimated:

$$MSE = \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K}\frac{1}{2}(\overline{p}_{nk} - p_{nk})^2 \tag{14}$$

where $N$ is the number of database objects, $K$ the number of attributes and $\overline{p}_{nk}$, $p_{nk}$ the desired and actual output probability for object $n$ having the attribute $k$, respectively.

The *Average Matching Error* is defined as follows: Every database object i is selected as a query in a Search & Retrieval session. For the set $R = \{r_1,...,r_J\}$ of the top J retrieved results a matching error is calculated using:

$$ME_i = \frac{1}{J}\sum_{j\in R} d_s(i,j) \tag{15}$$

where $d_s(i,j)$ is the semantic distance between the query i and the j-th retrieved result. The parameter J is fixed as the number of occurrences of the rarest attribute. The Average Matching Error is then calculated by:

$$AME = \frac{1}{N}\sum_{i=1}^{N} ME_i \tag{16}$$

Both metrics decrease as the performance increases.

During a Search & Retrieval session, the overall distance (Equation 5) between two models is needed. The semantic weight is proportional to the percentage of the annotated models. That means that on system start, searches are based

mainly on content. As more and more annotations occur, the system changes the balance towards semantics.

The following figures show the performance of the automatic annotation process on all test databases, compared to the manual annotation process. Figure 7 shows the results for the ITI database. The horizontal axis of the figure is the percentage of annotated objects. The vertical axis is the percentage of objects that are properly annotated. The black curve shows the performance of the manual annotation (identity function), while the red curve indicates the performance of the automatic annotation system. After the manual annotation of 40.82 % of the database, all models are properly annotated. Figure 8 shows the MSE (green curve) and AME (blue curve) error metrics for the ITI database, respectively. Figures 9 and 10 depict the performance and the error metrics for the VICTORY database. After the manual annotation of 45.12 % of the database, all models are properly annotated. The results show a similar system behaviour for both databases. As one can easily notice in the figures, the system starts operating after a number of objects has been annotated. This is due to the requirement, that minimum one object per attribute must be manually annotated, before the system can operate. After this, the system starts learning with great speed. After a number of characteristic models per attribute has been annotated, the system can easily recognize the majority of the models and the learning speed decreases. At this point, the outliers become more "interesting" for the system, thus they are selected for annotation.

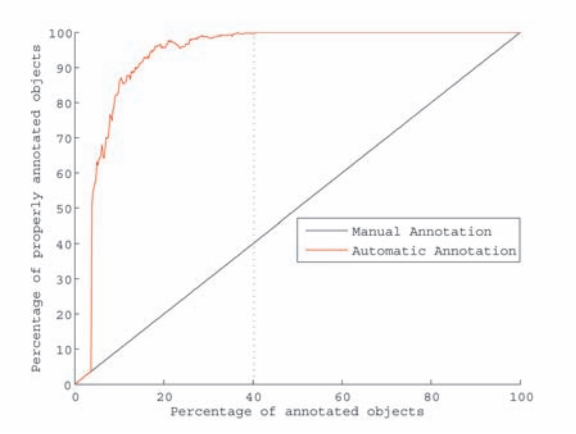


**Figure 7:** *Performance of the system for the ITI database*

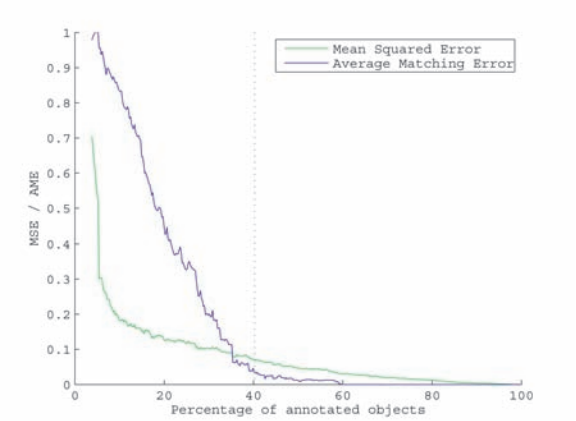


**Figure 8:** *Error metrics for the ITI database*

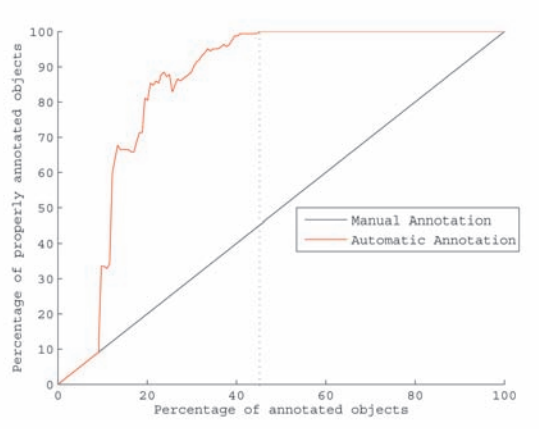


**Figure 9:** *Performance of the system for the VDB database*

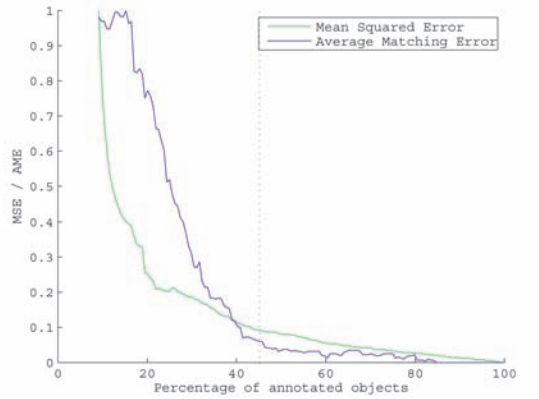


**Figure 10:** *Error metrics for the VDB Database*

## 4. Conclusions

In this paper a neurofuzzy approach for active learning-based annotation propagation was proposed. A neurofuzzy controller set was used to estimate the attributes of each database model using knowledge obtained from manual annotations of objects suggested by the system. The selec-

tion of the training examples was based on the information content of the models. Although the proposed framework induced a substantial acceleration of the annotation process, one cannot neglect the fact that the whole process still depends on the quality of the geometrical information. A workaround for this unwanted property would be the integration of other techniques, such as Relevance Feedback.

Another interesting perspective for the effectiveness of the current concept is scalability. From the experimental results it is obvious that in databases of a few hundred objects roughly 25 % of the objects have to be manually annotated so that annotation is effectively propagated to 90 % of the total objects in the database. To get a 100 % correct propagation, more than 40 % of the objects need to be annotated manually. For large databases this might still be prohibitive regarding manual annotation effort. The implementation of the concept in a distributed manner would be a significant help in this direction. While the 3D data set and the acquired knowledge can be kept in a central location, multiple annotators can work on this dataset over a distributed user interface.

## 5. Future Work

As mentioned above, in the current concept it is assumed that each object has maximum one attribute per attribute tree level. This limitation is not acceptable in a real annotation system, since an object can be described in many different contexts (e.g. a marble statue can be described as an artefact, a material, a time period, a human, etc.). Furthermore, the attribute tree is a static structure. No attribute creation/deletion/editing is currently supported. Another known limitation is the minimum size of the categories (minimum number of objects having a specific attribute) in order for each neurofuzzy controller to have enough input to be trained properly. This size depends on the diversity of the objects of each category. Finally, the integration of a Relevance Feedback scheme in the current concept would noticeably accelerate the automatic annotation process and also reduce the dependency on the geometrical information.

## 6. Acknowledgements

This work was supported by the EC projects ***VICTORY*** (http://www.victory-eu.org) and ***CATER*** (http://www.caterist.org/).

## References

[Cal99] CALLAN R.: *The Essence of Neural Networks*. Prentice Hall, London, England, 1999.

[CMM*00] COX I. J., MILLER M. L., MINKA T. P., PAPATHOMAS T. V., YIANILOS P. N.: The bayesian image retrieval system, pichunter: Theory, implementation, and psychophysical experiments. *IEEE TRANSACTIONS ON IMAGE PROCESSING Volume 9*, Issue 1 (Jan. 2000).

[CP04] CHAKRABORTY D., PAL N. R.: A neuro-fuzzy scheme for simultaneous feature selection and fuzzy rule-based classification. *IEEE TRANSACTIONS ON NEURAL NETWORKS Volume 15*, No. 1 (Jan. 2004).

[DPA*07] D.ZARPALAS, P.DARAS, A.AXENOPOULOS, D.TZOVARAS, M.G.STRINTZIS: 3d model search and retrieval using the spherical trace transform. *EURASIP Journal on Advances in Signal Processing* (Jan. 2007).

[FSST93] FREUND Y., SEUNG H. S., SHAMIR E., TISHBY N.: *Selective Sampling Using the Query by Committee Algorithm, Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 1993.

[HMK*02] HE X., MA W.-Y., KING O., LI M., ZHANG H.: Learning and inferring a semantic space from user's relevance feedback for image retrieval. *International Multimedia Conference Proceedings, 10th ACM international conference on Multimedia, Juan-les-Pins, France* (2002).

[Jan93] JANG J.-S. R.: Anfis: adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man and Cybernetics Volume 23* (1993), 665–685.

[JM04] JEON J., MANMATHA R.: Using maximum entropy for automatic image annotation. *Lecture notes in computer science* (2004).

[KBLP07] KODOGIANNIS V., BOULOUGOURA M., LYGOURAS J., PETROUNIAS I.: A neuro-fuzzy-based system for detecting abnormal patterns in wireless-capsule endoscopic images. *Neurocomputing, Selected papers from the International Conference on Intelligent Computing (ICIC 2005) Volume 70*, Issues 4-6 (Jan. 2007).

[Kra03] KRAISS K.-F.: *Man-Mashine Interaction II, Lecture Script*. Institute for Man-Mashine Interaction, RWTH Aachen, Germany, 2003.

[LJH06] LYU, JIN, HOI: A unified log-based relevance feedback scheme for image retrieval. *IEEE Transactions on Knowledge and Data Engineering Volume 18*, Issue 4 (Apr. 2006), 509–524.

[RHW86] RUMELHART D., HINTON G., WILLIAMS R.: *Learning internal representations by error propagation, Parallel Distributed Processing: Explorations in the Microstructures of Cognition*. MIT Press, Cambrigde, 1986.

[SS03] SHEVADE B., SUNDARAM H.: Vidya: An experiental annotation system. *Arts Media and Engineering Program, Arizona State University* (2003).

[ZC02] ZHANG C., CHEN T.: An active learning framework for content based information retrieval. *Technical Report, CMU-AMP-01-04* (2002).

[ZS01] ZHANG H., SU Z.: Improving cbir by semantic propagation and cross modality query expansion. *In Proceedings of the international workshop on MultiMedia Content-Based Indexing and Retrieval* (Sept. 2001).