

# Authentic Facial Expression Analysis

Nicu Sebe<sup>1</sup>, Michael S. Lew<sup>2</sup>, Ira Cohen<sup>3</sup>, Yafei Sun<sup>4</sup>, Theo Gevers<sup>1</sup>, Thomas S. Huang<sup>5</sup>

<sup>1</sup>Faculty of Science, University of Amsterdam, The Netherlands

{nicu, gevers}@science.uva.nl

<sup>2</sup>Leiden Institute of Advanced Computer Science, Leiden University, The Netherlands

mlew@liacs.nl

<sup>3</sup>HP Labs, USA

icohen@exch.hpl.hp.com

<sup>4</sup>School of Computer Science and Engineering, Sichuan University, China

sunyafei@cs.scu.edu.cn

<sup>5</sup>Beckman Institute for Advanced Science and Technology, University of Illinois at Urbana-Champaign, IL, USA

huang@ifp.uiuc.edu

## Abstract

*It is argued that for the computer to be able to interact with humans, it needs to have the communication skills of humans. One of these skills is the ability to understand the emotional state of the person. The most expressive way humans display emotions is through facial expressions. In most facial expression systems and databases, the emotion data was collected by asking the subjects to perform a series of facial expressions. However, these directed or deliberate facial action tasks typically differ in appearance and timing from the authentic facial expressions induced through events in the normal environment of the subject. In this paper, we present our effort in creating an authentic facial expression database based on spontaneous emotions derived from the environment. Furthermore, we test and compare a wide range of classifiers from the machine learning literature that can be used for facial expression classification.*

## 1 Introduction

Extensive studies of human facial expressions performed by Ekman [11, 12] gave evidence to support universality in facial expressions. According to these studies, the “universal facial expressions” are those representing happiness, sadness, anger, fear, surprise, and disgust. To code facial expressions, Ekman and Friesen [12] developed the Facial Action Coding System (FACS) in which the movements on the face are described by a set of action units (AUs) which have some related muscular basis. Ekman’s work inspired many researchers to analyze facial expressions by means of image and video processing. By tracking facial features and measuring the amount of facial movement, they attempt to categorize different facial expressions. Recent work on facial expression analysis and recognition [2, 4, 9, 22, 28] has used these “basic expressions” or a subset of them. The two recent surveys in the area [13, 23] provide an in depth review of many of the research done in recent years. All the methods developed are similar in that they first extract some features from the images or video, then

these features are used as inputs into a classification system, and the outcome is one of the preselected emotion categories. They differ mainly in the features extracted and in the classifiers used to distinguish between the different emotions.

Construction and labeling of a good database of facial expressions requires expertise, time, and training of subjects. Only a few such databases are available, such as the Cohn-Kanade [17] and JAFFE [20] databases. Most (or perhaps all) of these existing facial expression data have been collected by asking the subjects to perform a series of expressions. The main problem with this approach is that these deliberate facial action tasks typically differ in appearance and timing from the authentic facial expressions induced through events in the normal environment of the subject. Kanade et al. [17] consider the distinction between the deliberate and spontaneous/authentic facial actions and show that deliberate facial behavior is mediated by separate motor pathways than spontaneous facial behaviors. As a consequence, for a representative test for detecting human emotions in spontaneous settings, we need a test set which captures facial expressions in spontaneous settings. To be specific, one of our main contributions in this work is to create a test set in which the facial emotions correspond to the true emotional state of the person at that moment, not an artificial facial expression which does not match the emotional state. As far as we are aware, this is the first attempt to create an authentic emotion database. More details are given in Section 2.

We have developed a real time facial expression recognition system. This system presented briefly in Section 3 uses a model based non-rigid face tracking algorithm to extract motion features that serve as input to a classifier used for recognizing the different facial expressions. We were also interested in testing different classifiers from the machine learning literature that can be used for facial expression analysis. We present an extensive evaluation of 24 classifiers using our authentic emotion database and the Cohn-Kanade database (Section 4). We have concluding remarks in Section 5.

## 2 Authentic Expression Database

In many applications of human computer interaction, it is important to be able to detect the emotional state of the person in a natural situation. However, as any photographer can attest, getting a real smile can be challenging. Asking someone to smile often does not create the same picture as an authentic smile. The fundamental reason of course is that the subject often does not feel happy so his smile is artificial and in many subtle ways quite different than a genuine smile.

Our goal for the authentic expression database was to create ground truth where the facial expressions would correspond to the current emotional state of the subject. We consulted several members of the psychology department who recommended that the test be constrained as to minimize bias. First, the subjects could not know that they were being tested for their emotional state. Knowing that one is in a scientific test can invalidate or bias the results by influencing the emotional state. Second, we would need to interview each subject after the test to find out their true emotional state for each expression. Third, we were warned that even having a researcher in the same room with the subject could bias the results.



Figure 1: A snap shot of our facial expression recognition system. On the right side is a wireframe model overlaid on a face being tracked. On the left side the correct expression, Angry, is detected (the bars show the relative probability of Angry compared to the other expressions). The example is from the authentic database.

We decided to create a video kiosk with a hidden camera which would display segments from recent movie trailers. This method had the main advantages that it would naturally attract people to watch it and we could potentially elicit emotions through different genres of video footage - i.e. horror films for shock, comedy for joy, etc. From over 60 people who used the video kiosk, we were able to get the agreement of 28 students within the computer science department for the database. After each subject had seen the video trailers, they were interviewed to find out their emotional state corresponding to the hidden camera video footage. We also secured agreement for the motion data from their video footage

to be distributed to the scientific community which is one of the primary goals for this database.

In this kind of experiment, we can only capture the expressions corresponding to the naturally occurring emotions. This means that our range of emotions for the database was constrained to the ones genuinely felt by the subjects. For this database, the emotions found were either (1) Neutral; (2) Joy; (3) Surprise, or (4) Disgust. From having created the database, some items of note based purely on our experiences: (1) It is very difficult to get a wide range of emotions for all of the subjects. Having all of the subjects experience genuine sadness (or fear) for example is difficult. (2) The facial expressions corresponding to the internal emotions is often misleading. Some of the subjects appeared to be sad when they were actually happy. (3) Students are usually open to having the data extracted from the video used for test sets. The older faculty members were generally not agreeable to being part of the database.

## 3 Facial Expression Recognition System

In this section we briefly present our real time facial expression recognition system. The system is composed of a face tracking algorithm which outputs a vector of motion features of certain regions of the face. The features are used as inputs to a classifier. A snap shot of the system with the face tracking and the recognition result is shown in Figure 1.

### 3.1 Face Tracking and Feature Extraction

The face tracking we use is based on a system developed by Tao and Huang [26] called the Piecewise Bézier Volume Deformation (PBVD) tracker. This face tracker uses a model-based approach where an explicit 3D wireframe model of the face is constructed. In the first frame of the image sequence, landmark facial features such as the eye and mouth corners are selected interactively. A face model consisting of 16 surface patches embedded in Bézier volumes is then warped to fit the selected facial features. The surface patches defined this way are guaranteed to be continuous and smooth. The shape of the mesh can be changed by changing the locations of the control points in the Bézier volume.

Once the model is constructed and fitted, head motion and local deformations of the facial features such as the eyebrows, eyelids, and mouth can be tracked. First the 2D image motions are measured using template matching between frames at different resolutions. Image templates from the previous frame and from the very first frame are both used for more robust tracking. The measured 2D image motions are modeled as projections of the true 3D motions onto the image plane. From the 2D motions of many points on the mesh, the 3D motion can be estimated by solving an overdetermined system of equations of the projective motions in the least squared sense.

The recovered motions are represented in terms of magnitudes of some predefined motion of various facial features. Each feature motion corresponds to a simple deformation on

the face, defined in terms of the Bézier volume control parameters. We refer to these motions vectors as Motion-Units (MU's). Note that they are similar but not equivalent to Ekman's AU's and are numeric in nature, representing not only the activation of a facial region, but also the direction and intensity of the motion. The MU's used in the face tracker are shown in Figure 2. The MU's are used as the basic features for the classifiers described in the next section.

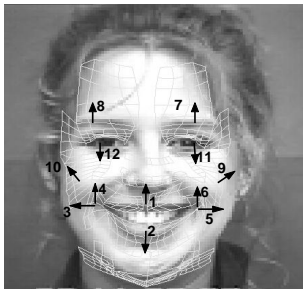


Figure 2: The 12 facial motion measurements

## 3.2 Classifiers

Several classifiers from the machine learning literature were considered in our system. We give a brief description for each of the classifiers and ask the reader to get more details from the original references. We also investigated the use of voting algorithms to improve the classification results.

### 3.2.1 Generative Bayesian Networks classifiers

Bayesian networks can represent joint distributions in an intuitive and efficient way; as such, Bayesian networks are naturally suited to classification. We can use a Bayesian network to compute the posterior probability of a set of *labels* given the observable *features*, and then we classify the features with the most probable label.

A Bayesian network is composed of a directed acyclic graph in which every node is associated with a variable  $X_i$  and with a conditional distribution  $p(X_i|\Pi_i)$ , where  $\Pi_i$  denotes the parents of  $X_i$  in the graph. The directed acyclic graph is the *structure*, and the distributions  $p(X_i|\Pi_i)$  represent the *parameters* of the network. A Bayesian network classifier is a *generative* classifier when the class variable is an ancestor (e.g., parent) of some or all features. We consider three examples of generative Bayesian Networks:

**NB** is the Naive-Bayes classifier [10] which makes the assumption that all features are conditionally independent given the class label. Although this assumption is typically violated in practice, NB have been used successfully in many classification problems. One of the reasons of the success of NB is attributed to the small number of parameters needed to be learnt. Better results may be achieved by discretizing the continuous input features. The **Nbd** classifier provides this preprocessing step by chaining a discretization filter to the NB classifier.

**TAN** is the Tree-Augmented Naive-Bayes classifier [15].

This classifier attempts to find a structure that captures the dependencies among the input features. Of course, finding all the dependencies is an NP-complete problem so TAN restricts to a simple model: the class variable is the parent of all the features and each feature has at most one other feature as a parent, such that the resultant graph of the features forms a tree. Using the algorithm of Friedman et al. [15], the most likely TAN classifier can be estimated efficiently.

**SSS** is the stochastic structure search algorithm [7]. This classifier goes beyond the simplifying assumptions of NB and TAN and searches for the correct Bayesian network structure focusing on classification. The idea is to use a strategy that can efficiently search through the whole space of possible structures and to extract the ones that give the best classification results.

### 3.2.2 The Decision Tree Inducers

The purpose of the decision tree inducers is to create from a given data set an efficient description of a classifier by means of a decision tree. The decision tree represents a data structure which efficiently organizes descriptors. The purpose of the tree is to store an ordered series of descriptors. As one travels through the tree, he is asked questions and the answers determine which further questions will be asked. At the end of the path is a classification. When viewed as a black box the decision tree represents a function of parameters (or descriptors) leading to a certain value of the classifier. We consider the following decision tree algorithms and use their *MCC++* implementation [18]:

**ID3** is a very basic decision tree algorithm with no pruning based on [24].

**C4.5** is an extension of ID3 that accounts for unavailable values, continuous attribute value ranges, and pruning of decision trees [25].

**MC4** is similar to C4.5 [25] with the exception that unknowns are regarded as a separate value. The algorithm grows the decision tree following the standard methodology of choosing the best attribute according to the evaluation criterion. After the tree is grown, a pruning phase replaces subtrees with leaves using the same pruning algorithm that C4.5 uses.

**OC1** is the Oblique decision tree algorithm by Murthy [21]. It combines deterministic hill-climbing with two forms of randomization to find a good oblique split (in the form of a hyperplane) at each node of a decision tree.

### 3.2.3 Other inducers

**SVM** Support vector machines [27] were developed based on the Structural Risk Minimization principle from statistical learning theory. They are one of the most popular classifiers and can be applied to regression, classification, and density estimation problems. For the SVM kernel we used Vapnik's polynomial [27] of order 5.

**kNN** is the instance-based learning algorithm (nearest-neighbor) by Aha [1]. This is a good, robust algorithm, but slow when there are many attributes.

**PEBLS** is the Parallel Exemplar-Based Learning System by Cost and Salzberg [8]. This is a nearest-neighbor learning system designed for applications where the instances have symbolic feature values.

**CN2** is the direct rule induction algorithm by Clark and Niblett [6]. This algorithm inductively learns a set of propositional if...then... rules from a set of training examples. To do this, it performs a general-to-specific beam search through rule-space for the “best” rule, removes training examples covered by that rule, then repeats until no more “good” rules can be found.

**Winnov** is the multiplicative algorithm described in [19].

**Perceptron** is the simple algorithm described in [16]. Both Perceptron and Winnov are classifiers that build linear discriminators and they are only capable of handling continuous attributes with no-unknowns and two-class problem. For our multi-class problem we implemented several classifiers, each classifying one class against the rest of the classes and in the end we averaged the results.

### 3.2.4 Voting algorithms

Methods for voting classification, such as Bagging and Boosting (AdaBoost) have been shown to be very successful in improving the accuracy of certain classifiers for artificial and real-world datasets [3]. A voting algorithm takes an inducer and a training set as input and runs the inducer multiple times by changing the distribution of training set instances. The generated classifiers are then combined to create a final classifier that is used to classify the test set.

The **bagging** algorithm (**B**ootstrap **a**ggregating) by Breiman [5] votes classifiers generated by different bootstrap samples (replicates). A bootstrap sample is generated by uniformly sampling  $m$  instances from the training set with replacement.  $T$  bootstrap samples  $B_1, B_2, \dots, B_T$  are generated and a classifier  $C_i$  is built from each bootstrap sample  $B_i$ . A final classifier  $C^*$  is built from  $C_1, C_2, \dots, C_T$  whose output is the class predicted most often by its sub-classifiers, with ties broken arbitrarily. Bagging works best on unstable inducers (e.g., decision trees), that is, inducers that suffer from high variance because of small perturbations in the data. However, bagging may slightly degrade performance of stable algorithms (e.g. kNN) because effectively smaller training sets are used for training each classifier.

Like bagging, **AdaBoost** (**A**daptive **B**oosting) [14] generates a set of classifiers and votes them. The AdaBoost however, generates classifiers sequentially, while bagging can generate them in parallel. AdaBoost also changes the weights of the training instances provided as input to each inducer based on classifiers that were previously built. The goal is to force the inducer to minimize the expected error over different input distributions. Given an integer  $T$  specifying the

number of trials,  $T$  weighted training sets  $S_1, S_2, \dots, S_T$  are generated in sequence and  $T$  classifiers  $C_1, C_2, \dots, C_T$  are built. A final classifier  $C^*$  is formed using a weighted voting scheme: the weight of each classifier depends upon its performance on the training set used to build it.

## 4 Facial Expression Recognition Experiments

In our experiments we use the authentic database described in Section 2 and the Cohn-Kanade AU code facial expression database [17].

The Cohn-Kanade database [17] consists of expression sequences of subjects, starting from a Neutral expression and ending in the peak of the facial expression. There are 104 subjects in the database. However, because for some of the subjects, not all of the six facial expressions sequences were available to us, we used only a subset of 53 subjects, for which at least four of the sequences were available.

For both databases we only have a small number of frames per person for each expression which makes insufficient data to perform person dependent tests. We measure the classification error of each frame, where each frame in the video sequence was manually labeled to one of the expressions. This manual labeling can introduce some ‘noise’ in our classification because the boundary between Neutral and the expression of a sequence is not necessarily optimal, and frames near this boundary might cause confusion between the expression and the Neutral. A different labeling scheme is to label only some of the frames that are around the peak of the expression leaving many frames in between unlabeled. We did not take this approach because a real-time classification system would not have this information available to it.

When performing the error estimation we used  $n$ -fold cross-validation ( $n=10$  in our experiments) in which the dataset was randomly split into  $n$  mutually exclusive subsets (the folds) of approximately equal size. The inducer is trained and tested  $n$  times; each time tested on a fold and trained on the dataset minus the fold. The cross-validation estimate of error is the average of the estimated errors from the  $n$  folds. To show the statistical significance of our results we also present the 95% confidence intervals for the classification errors.

We show the results for all the classifiers in Table 1. Note that the results for the authentic database outperform the ones for the Cohn-Kanade database. One reason for this is that we have a simpler classification problem: only 4 classes are available. Surprisingly, the best classification results are obtained with the kNN classifier ( $k=3$  in our experiments). This classifier is a distance-based classifier and does not assume any model. It seems that facial expression recognition is not a simple classification problem and all the models tried (e.g., NB, TAN, or SSS) were not able to entirely capture the complex decision boundary that separates the different expressions. This argumentation may also explain the surprisingly poor behavior of the SVM.

Classifiers	Datasets	
	Authentic	Cohn-Kanade
NB	8.46 ± 0.93%	24.40 ± 0.85%
NB bagging	8.35 ± 0.92%	24.33 ± 0.82%
NB boosting	8.25 ± 0.97%	24.45 ± 0.82%
NBd	8.46 ± 0.93%	24.40 ± 0.85%
NBd bagging	9.26 ± 1.15%	21.08 ± 0.49%
NBd boosting	8.65 ± 1.03%	18.95 ± 0.53%
TAN	6.46 ± 0.34%	13.20 ± 0.27%
SSS	5.89 ± 0.67%	11.40 ± 0.65%
ID3	9.76 ± 1.00%	16.70 ± 0.53%
ID3 bagging	7.45 ± 0.66%	11.82 ± 0.48%
ID3 boosting	6.96 ± 1.00%	10.70 ± 0.53%
C4.5	8.45 ± 0.91%	16.10 ± 0.69%
MC4	8.45 ± 0.94%	16.32 ± 0.53%
MC4 bagging	7.35 ± 0.76%	12.08 ± 0.32%
MC4 boosting	5.84 ± 0.78%	8.28 ± 0.43%
OC1	9.05 ± 1.10%	16.87 ± 0.29%
SVM	13.23 ± 0.93%	24.58 ± 0.76%
kNN	4.43 ± 0.97%	6.96 ± 0.40%
kNN bagging	4.53 ± 0.97%	6.94 ± 0.40%
kNN boosting	4.43 ± 0.97%	6.96 ± 0.40%
PEBLS	6.05 ± 1.09%	15.29 ± 0.27%
CN2	9.26 ± 0.82%	27.54 ± 0.27%
Winnow	12.07 ± 1.87%	15.69 ± 2.06%
Perceptron	7.75 ± 1.41%	12.50 ± 1.54%

Table 1: Classification errors for facial expression recognition together with their 95% confidence intervals.

kNN may give the best classification results but it has its own disadvantages: it is computationally slow and needs to keep all the instances in the memory. The main advantage of the model-based classifiers is their ability to incorporate unlabeled data [7]. This is very important since labeling data for emotion recognition is very expensive and requires expertise, time, and training of subjects. However, collecting unlabeled data is not as difficult. Therefore, it is beneficial to be able to use classifiers that are learnt with a combination of some labeled data and a large amount of unlabeled data.

Another important aspect to notice is that the voting algorithms improve the classification results of the decision trees algorithms but do not significantly improve the results of the more stable algorithms such as NB and kNN.

We were also interested to investigate how the classification error behaves when more and more training instances are available. The corresponding learning curves are presented in Figure 3. As expected kNN improves significantly as more data are used for training.

To analyze the confusion between the different emotions, we present the average confusion matrices of two classifiers in Tables 2 and 3. Note that most of the expressions are detected with high accuracy and the confusion is larger with the Neutral class. One reason why Surprise is detected with only 81% for the authentic database is that in general this emotion is blended with happiness. This is why in this case the confu-

sion with Happy is much larger than that with the other emotions.

Emotion	Neutral	Happy	Surprise	Disgust
Neutral	<u>94.20</u>	3.96	0.60	1.24
Happy	6.34	<u>93.14</u>	0.00	0.52
Surprise	2.43	16.43	<u>81.14</u>	0.00
Disgust	0.45	0.45	0.00	<u>99.10</u>

Table 2: Average confusion matrix using the MC4 boosting classifier (authentic database)

## 5 Conclusion

In this work we presented our efforts in creating an authentic facial expression database based on spontaneous emotions. We created a video kiosk with a hidden camera which displayed segments of movies and allowed filming of several subjects that showed spontaneous emotions. One of our main contributions in this work was to create a database in which the facial expressions correspond to the true emotional state of the subjects. As far as we are aware this is the first attempt to create such a database and our intention is to make it available to the scientific community.

Furthermore, we tested and compared a wide range of classifiers from the machine learning community including Bayesian Networks, decision trees, SVM, kNN, etc. We also considered the use of voting classification schemes such as bagging and boosting to improve the classification results of the classifiers. We demonstrated the classifiers for facial expression recognition using our authentic database and the Cohn-Kanade database. Finally, we integrated the classifiers and a face tracking system to build a real time facial expression recognition system.

## References

- [1] D.W. Aha. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies*, 36(1):267–287, 1992.
- [2] M.S. Bartlett, I. Littlewort, G. Fasel, and J.R. Movellan. Real time face detection and expression recognition: Development and application to human-computer interaction. In *CVPR Workshop on Computer Vision and Pattern Recognition for Human-Computer Interaction*, 2003.
- [3] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–142, 1999.
- [4] F. Bourel, C. Chibelushi, and A. Low. Robust facial expression recognition using a state-based model of spatially-localised facial dynamic. In *Int. Conference on Automatic Face and Gesture Recognition*, pages 113–118, 2002.
- [5] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.
- [6] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

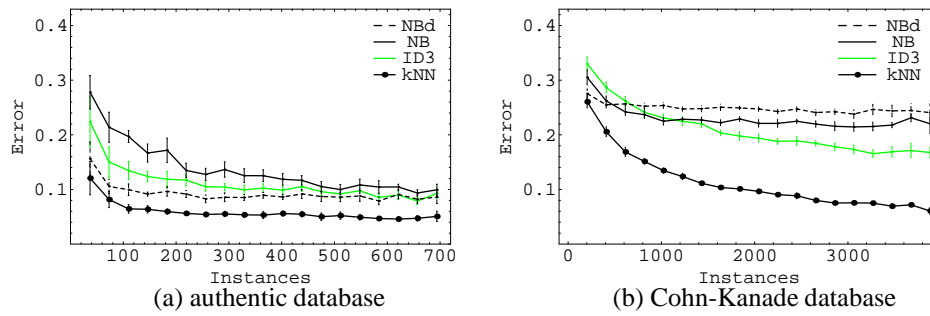


Figure 3: The learning curve for different classifiers. The vertical bars represent the 95% confidence intervals.

Emotion	Neutral	Happy	Surprise	Anger	Disgust	Fear	Sad
Neutral	<u>94.81</u>	1.23	0.55	0.61	0.80	0.67	1.33
Happy	2.5	<u>95.69</u>	0.17	0.00	0.17	1.3	0.17
Surprise	5.38	2.05	<u>88.71</u>	0.00	0.25	1.79	0.37
Anger	5.54	0.46	0.00	<u>91.22</u>	1.84	0.00	0.94
Disgust	8.63	0.00	0.00	4.31	<u>85.61</u>	0.71	0.74
Fear	3.63	1.21	0.25	0.00	0.24	<u>94.67</u>	0.00
Sad	7.18	0.00	0.02	0.06	0.00	0.00	<u>91.99</u>

Table 3: Average confusion matrix using the kNN classifier (Cohn-Kanade database)

- [7] I. Cohen, F. Cozman, N. Sebe, M. Cirello, and T. Huang. Semi-supervised learning of classifiers: Theory, algorithms for bayesian network classifiers and applications to human-computer interaction. *IEEE Trans. PAMI*, to appear in 2004.
- [8] S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10(1):57–78, 1993.
- [9] G. Donato, M.S. Bartlett, J.C. Hager, P. Ekman, and T.J. Sejnowski. Classifying facial actions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(10):974–989, 1999.
- [10] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.
- [11] P. Ekman. Strong evidence for universals in facial expressions: A reply to Russell’s mistaken critique. *Psychological Bulletin*, 115(2):268–287, 1994.
- [12] P. Ekman and W.V. Friesen. *Facial Action Coding System: Investigator’s Guide*. Consulting Psychologists Press, 1978.
- [13] B. Fasel and J. Luetttin. Automatic facial expression analysis: A survey. *Pattern Recognition*, 36:259–275, 2003.
- [14] Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.
- [15] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2):131–163, 1997.
- [16] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computation*. Addison Wesley, 1991.
- [17] T. Kanade, J. Cohn, and Y. Tian. Comprehensive database for facial expression analysis. In *Int. Conference on Automatic Face and Gesture Recognition*, pages 46–53, 2000.
- [18] Ron Kohavi, Dan Sommerfield, and James Dougherty. Data mining using *M<sub>LC</sub>++*: A machine learning library in C++. *International Journal on Artificial Intelligence Tools*, 6(4):537–566, 1997.
- [19] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 10(1):57–78, 1993.
- [20] M. Lyons, A. Akamatsu, M. Kamachi, and J. Gyoba. Coding facial expressions with Gabor wavelets. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 200–205, 1998.
- [21] S.K. Murthy, S. Kasif, and S. Salzberg. A system for the induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–33, 1994.
- [22] N. Oliver, A. Pentland, and F. Bérard. LAFTER: A real-time face and lips tracker with facial expression recognition. *Pattern Recognition*, 33:1369–1382, 2000.
- [23] M. Pantic and L.J.M. Rothkrantz. Automatic analysis of facial expressions: The state of the art. *IEEE Trans. on PAMI*, 22(12):1424–1445, 2000.
- [24] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.
- [25] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufman, 1993.
- [26] H. Tao and T.S. Huang. Connected vibrations: A modal analysis approach to non-rigid motion tracking. In *CVPR*, pages 735–740, 1998.
- [27] V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.
- [28] Y. Zhang and Q. Ji. Facial expression understanding in image sequences using dynamic and active visual information fusion. In *ICCV*, pages 113–118, 2003.