UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

Machine learning techniques for face analysis

Valenti, R.; Sebe, N.; Gevers, T.; Cohen, I.

DOI
10.1007/978-3-540-75171-7_7
Publication date
2008

Published in
Machine learning techniques for multimedia: Case studies on organization and retrieval

Link to publication

Citation for published version (APA):
Valenti, R., Sebe, N., Gevers, T., & Cohen, I. (2008). Machine learning techniques for face analysis. In M. Cord, & P. Cunningham (Eds.), *Machine learning techniques for multimedia: Case studies on organization and retrieval* (pp. 159-187). (Cognitive Technologies). Springer. https://doi.org/10.1007/978-3-540-75171-7_7

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)

Download date:14 Nov 2026

Machine Learning Techniques for Face Analysis

Roberto Valenti[1], Nicu Sebe[1], Theo Gevers[1], and Ira Cohen[2]

[1] Faculty of Science, University of Amsterdam, The Netherlands
{rvalenti,nicu,gevers@science.uva.nl}
[2] HP Labs, USA
{iracohen@hp.com}

In recent years there has been a growing interest in improving all aspects of the interaction between humans and computers with the clear goal of achieving a natural interaction, similar to the way human-human interaction takes place. The most expressive way humans display emotions is through facial expressions. Humans detect and interpret faces and facial expressions in a scene with little or no effort. Still, development of an automated system that accomplishes this task is rather difficult. There are several related problems: detection of an image segment as a face, extraction of the facial expression information, and classification of the expression (e.g.,in emotion categories).A system that performs these operations accurately and in real time would be a major step forward in achieving a human-like interaction between the man and machine. In this chapter, we present several machine learning algorithms applied to face analysis and stress the importance of learning the structure of Bayesian network classifiers when they are applied to face and facial expression analysis.

1 Introduction

Information systems are ubiquitous in all human endeavors including scientific, medical, military, transportation, and consumer. Individual users use them for learning, searching for information (including data mining), doing research (including visual computing), and authoring. Multiple users (groups of users, and groups of groups of users) use them for communication and collaboration. And either single or multiple users use them for entertainment. An information system consists of two components: Computer (data/knowledge base, and information processing engine), and humans. It is the intelligent interaction between the two that we are addressing in this chapter.

Automatic face analysis has attracted increasing interest in the research community mainly due to its many useful applications. A system involving such an analysis assumes that the face can be accurately detected and tracked, the facial features can be precisely identified, and that the facial expressions, if any, can be precisely

classified and interpreted. For doing this, in the following, we present in detail the three essential components of our automatic system for human-computer interaction: face detection, facial feature detection, and facial emotion recognition. This chapter presents our real time facial expression recognition system [10] which uses a facial features detector and a model based non-rigid face tracking algorithm to extract motion features that serve as input to a Bayesian network classifier used for recognizing the different facial expressions. Parts of this system has been developed in collaboration with our colleagues from the Beckman Institute, University of Illinois at Urbana-Champaign, USA. We present here the components of the system and give reference to the publications that contain extensive details on the individual components [9, 40].

2 Background

2.1 Face Detection

Images containing face are essential to intelligent vision-based human-computer interaction. The rapidly expanding research in face processing is based on the premise that information about user's identity, state, and intend can be extracted from images and that computers can react accordingly, e.g., by observing a person's facial expression. Given an arbitrary image, the goal of face detection is to automatically locate a human face in an image or video, if it is present. Face detection in a general setting is a challenging problem for various reasons. The first set of reasons are inherent: there are many types of faces, with different colors, texture, sizes, etc. In addition, the face is a non-rigid object which can change its appearance. The second set of reasons are environmental: changing lighting, rotations, translations, and scales of the faces in natural images.

To solve the problem of face detection, two main approaches can be taken. The first is a model based approach, where a description of what is a human face is used for detection. The second is an appearance based approach, where we learn what faces are directly from their appearance in images. In this work, we focus on the latter.

There have been numerous appearance based approaches. We list a few from recent years and refer to the reviews of Yang et al. [46] and Hjelmas and Low [23] for further details. Rowley et al. [37] used Neural networks to detect faces in images by training from a corpus of face and non-face images. Colmenarez and Huang [11] used maximum entropic discrimination between faces and non-faces to perform maximum likelihood classification, which was used for a real time face tracking system. Yang et al. [47] used SNoW based classifiers to learn the face and non-face discrimination boundary on natural face images. Wang et al. [44] learned a minimum spanning weighted tree for learning pairwise dependencies graphs of facial pixels, followed by a discriminant projection to reduce complexity. Viola and Jones [43] used boosting and a cascade of classifiers for face detection.

Very relevant to our work is the research of Schneiderman [38] who learns a sparse structure of statistical dependecies for several object classes including faces. While analyzing such dependencies can reveal useful information, we go beyond the scope of Schneiderman's work and present a framework that not only learns the structure of a face but also allows the use of unlabeled data in classification.

Face detection provides interesting challenges to the underlying pattern classification and learning techniques. When a raw or filtered image is considered as input to a pattern classifier, the dimension of the space is extremely large (i.e., the number of pixels in normalized training images). The classes of face and non-face images are decidedly characterized by multimodal distribution functions and effective decision boundaries are likely to be non-linear in the image space. To be effective, the classifiers must be able to extrapolate from a modest number of training samples.

2.2 Facial Feature Detection

Various approaches to facial feature detection exist in the literature. Although many of the methods have been shown to achieve good results, they mainly focus on finding the location of some facial features (e.g., eyes and mouth corners) in restricted environments (e.g., constant lighting, simple background, etc.). Since we want to obtain a complex and accurate system of feature annotation, these methods are not suitable for us.

In recent years deformable model-based approaches for image interpretation have been proven very successful, especially in images containing objects with large variability such as faces. These approaches are more appropriate for our specific case since they make use of a template (e.g., the shape of an object). Among the early deformable template models is the Active Contour Model by Kass et al. [26] in which a correlation structure between shape markers is used to constrain local changes. Cootes et al. [14] proposed a generalized extension, namely Active Shape Models (ASM), where deformation variability is learned using a training set. Active Appearance Models (AAM) were later proposed in [12] and they are closely related to the simultaneous formulation of Active Blobs [39] and Morphable Models [24]. AAM can be seen as an extension of ASM which includes the appearance information of an object.

While active appearance models have been shown to be very successful, they suffer from important drawbacks such as background handling and initialization. Previous work tried to solve the latter by using an object detector to provide an acceptable model initialization. In Section 5.2, we bring this concept one step further and we reduce the existing AAM problems by considering the initialization information as a part of the active appearance model.

2.3 Emotion Recognition Research

Ekman and Friesen [17] developed the Facial Action Coding System (FACS) to code facial expressions where movements on the face are described by a set of action units (AUs). Each AU has some related muscular basis. This system of coding facial

expressions is done manually by following a set of prescribed rules. The inputs are still images of facial expressions, often at the peak of the expression. This process is very time-consuming.

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 has used these "basic expressions" or a subset of them. The two recent surveys in the area [35, 19] provide an in depth review of many of the research done in automatic facial expression recognition in recent years.

The work in computer-assisted quantification of facial expressions did not start until the 1990s. Black and Yacoob [2] used local parameterized models of image motion to recover non-rigid motion. Once recovered, these parameters were used as inputs to a rule-based classifier to recognize the six basic facial expressions. Essa and Pentland [18] used an optical flow region-based method to recognize expressions. Oliver et al. [32] used lower face tracking to extract mouth shape features and used them as inputs to an HMM based facial expression recognition system (recognizing neutral, happy, sad, and an open mouth). Chen [5] used a suite of static classifiers to recognize facial expressions, reporting on both person-dependent and person-independent results. Cohen et al. [10] describe classification schemes for facial expression recognition in two types of settings: dynamic and static classification. In the static setting, the authors learn the structure of Bayesian networks classifiers using as input 12 motion units given by a face tracking system for each frame in a video. For the dynamic setting, they used a multi-level HMM classifier that combines the temporal information and allows not only to perform the classification of a video segment to the corresponding facial expression, as in the previous works on HMM based classifiers, but also to automatically segment an arbitrary long sequence to the different expression segments without resorting to heuristic methods of segmentation.

These methods are similar in that they first extract some features from the images, 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 from the video images and in the classifiers used to distinguish between the different emotions.

3 Learning Classifiers for Human-Computer Interaction

Many pattern recognition and human-computer interaction applications require the design of classifiers. Classification is the task of systematic arrangement in groups or categories according to some set of observations, e.g., classifying images to those containing human faces and those that do not or classifying individual pixels as being skin or non-skin. Classification is a natural part of daily human activity and is performed on a routine basis. One of the tasks in machine learning has been to give the computer the ability to perform classification in different problems. In machine

classification, a classifier is constructed which takes as input a set of observations (such as images in the face detection problem) and outputs a prediction of the class *label* (e.g., face or no face). The mechanism which performs this operation is the *classifier*.

We are interested in probabilistic classifiers, in which the observations and class are treated as random variables, and a classification rule is derived using probabilistic arguments (e.g., if the probability of an image being a face given that we observed two eyes, nose, and mouth in the image is higher than some threshold, classify the image as a face). We consider two aspects. First, most of the research mentioned in the previous section tried to classify each observable independent from each the others. We want to take a different approach: can we learn the dependencies (the structure) between the observables (e.g., the pixels in an image patch)? Can we use this structure for classification? To achieve this we use Bayesian Networks. Bayesian Networks can represent joint distributions in an intuitive and efficient way; as such, Bayesian Networks are naturally suited for classification. Second, we are interested in using a framework that allows for the usage of labeled and unlabeled data (also called semi-supervised learning). The motivation for semi-supervised learning stems from the fact that labeled data are typically much harder to obtain compared to unlabeled data. For example, in facial expression recognition it is easy to collect videos of people displaying emotions, but it is very tedious and difficult to label the video to the corresponding expressions. Bayesian Networks are very well suited for this task: they can be learned with labeled and unlabeled data using maximum likelihood estimation.

Is there value to unlabeled data in supervised learning of classifiers? This fundamental question has been increasingly discussed in recent years, with a general optimistic view that unlabeled data hold great value. Due to an increasing number of applications and algorithms that successfully use unlabeled data [31, 41, 1] and magnified by theoretical issues over the value of unlabeled data in certain cases [4, 33], semi-supervised learning is seen optimistically as a learning paradigm that can relieve the practitioner from the need to collect many expensive labeled training data. However, several disparate empirical evidences in the literature suggest that there are situations in which the addition of unlabeled data to a pool of labeled data, causes degradation of the classifier's performance [31, 41, 1], in contrast to improvement of performance when adding more labeled data. Intrigued by these discrepancies, we performed extensive experiments, reported in [9]. Our experiments suggested that performance degradation can occur when the assumed classifier's model is incorrect. Such situations are quite common, as one rarely knows whether the assumed model is an accurate description of the underlying true data generating distribution. More details are given below (for the sake of consistency we keep the same notations as the one introduced in [9]).

The goal is to classify an incoming vector of observables $\mathbf{X}$. Each instantiation of $\mathbf{X}$ is a *sample*. There exists a *class variable* C; the values of C are the *classes*. Let $P(C, \mathbf{X})$ be the *true* joint distribution of the class and features from which any a sample of some (or all) of the variables from the set $\{C, \mathbf{X}\}$ is drawn, and let $p(C, \mathbf{X})$

be the density distribution associated with it. We want to build *classifiers* that receive a sample $\mathbf{x}$ and output either one of the values of C.

Probabilities of $(C, \mathbf{X})$ are estimated from data and then are fed into the optimal classification rule. Also, a parametrical model $p(C, \mathbf{X}|\theta)$ is adopted. An estimate of θ is denoted by $\hat{\theta}$ and we denote throughout by $\hat{\theta}^*$ the assimptotic value of $\hat{\theta}$. If the distribution $p(C, \mathbf{X})$ belongs to the family $p(C, \mathbf{X}|\theta)$, we say the "model is correct"; otherwise, we say the "model is incorrect". We use "estimation bias" loosely to mean the expected difference between $p(C, \mathbf{X})$ and the estimated $p\left(C, \mathbf{X}|\hat{\theta}\right)$.

The analysis presented in [9] and summarized here is based on the work of White [45] on the properties of maximum likelihood estimators without assuming model correctness. White [45] showed that under suitable regularity conditions, maximum likelihood estimators converge to a parameter set θ^* that minimizes the Kullback-Leibler (KL) distance between the assumed family of distributions, $p(Y|\theta)$, and the true distribution, $p(Y)$. White [45] also shows that the estimator is asymptotically Normal, i.e., $\sqrt{N}(\hat{\theta}_N - \theta^*) \sim \mathcal{N}(0, C_Y(\theta))$ as N (the number of samples) goes to infinity. $C_Y(\theta)$ is a covariance matrix equal to $A_Y(\theta)^{-1}B_Y(\theta)A_Y(\theta)^{-1}$, evaluated at θ^*, where $A_Y(\theta)$ and $B_Y(\theta)$ are matrices whose (i, j)'th element ($i, j = 1, ..., d$, where d is the number of parameters) is given by:

$$A_Y(\theta) = E\left[\partial^2 \log p(Y|\theta) / \partial\theta_i\theta_j\right],$$
$$B_Y(\theta) = E[(\partial \log p(Y|\theta) / \partial\theta_i)(\partial \log p(Y|\theta) / \partial\theta_j)].$$

Using these definitions, in [9] the following theorem was introduced:

Theorem 1. *Consider supervised learning where samples are randomly labeled with probability λ. Adopt the regularity conditions in Theorems 3.1, 3.2, 3.3 from [45], with Y replaced by $(C, \mathbf{X})$ and by $\mathbf{X}$, and also assume identifiability for the marginal distributions of $\mathbf{X}$. Then the value of θ^*, the limiting value of maximum likelihood estimates, is:*

$$\arg\max_{\theta} \left(\lambda E[\log p(C, \mathbf{X}|\theta)] + (1 - \lambda)E[\log p(\mathbf{X}|\theta)]\right), \tag{1}$$

where the expectations are with respect to $p(C, \mathbf{X})$. Additionally, $\sqrt{N}(\hat{\theta}_N - \theta^*) \sim \mathcal{N}(0, C_\lambda(\theta))$ as $N \to \infty$, where $C_\lambda(\theta)$ is given by:

$$\begin{aligned} C_\lambda(\theta) &= A_\lambda(\theta)^{-1}B_\lambda(\theta)A_\lambda(\theta)^{-1} \text{ with,} \\ A_\lambda(\theta) &= \left(\lambda A_{(C,\mathbf{X})}(\theta) + (1 - \lambda)A_{\mathbf{X}}(\theta)\right) \text{ and} \\ B_\lambda(\theta) &= \left(\lambda B_{(C,\mathbf{X})}(\theta) + (1 - \lambda)B_{\mathbf{X}}(\theta)\right), \end{aligned} \tag{2}$$

evaluated at θ^*. □

For a proof of this theorem we direct the interested reader to [9]. Here we restrict only to a few observations. Expression (1) indicates that semi-supervised learning can be viewed asymptotically as a "convex" combination of supervised and unsupervised learning. As such, the objective function for semi-supervised learning is a

combination of the objective function for supervised learning ($E[\log p(C, \mathbf{X}|\theta)]$) and the objective function for unsupervised learning ($E[\log p(\mathbf{X}|\theta)]$).

Denote by θ^*_λ the value of θ that maximizes Expression (1) for a given λ. Then, θ^*_1 is the asymptotic estimate of θ for *supervised* learning, denoted by θ^*_l. Likewise, θ^*_0 is the asymptotic estimate of θ for *unsupervised* learning, denoted by θ^*_u.

The asymptotic covariance matrix is positive definite as $B_Y(\theta)$ is positive definite, $A_Y(\theta)$ is symmetric for any Y, and

$$\theta A(\theta)^{-1} B_Y(\theta) A(\theta)^{-1} \theta^T = w(\theta) B_Y(\theta) w(\theta)^T > 0,$$

where $w(\theta) = \theta A_Y(\theta)^{-1}$. We see that asymptotically, an increase in N, the number of labeled and unlabeled samples, will lead to a reduction in the variance of $\hat{\theta}$. Such a guarantee can perhaps be the basis for the optimistic view that unlabeled data should always be used to improve classification accuracy. In [9] it was shown that this observation holds when the model is correct, and that when the model is incorrect this observation might not always hold.

3.1 Model Is Correct

Suppose first that the family of distributions $P(C, \mathbf{X}|\theta)$ contains the distribution $P(C, \mathbf{X})$; that is, $P(C, \mathbf{X}|\theta_\top) = P(C, \mathbf{X})$ for some $\theta_\top$. Under this condition, the maximum likelihood estimator is consistent, thus, $\theta^*_l = \theta^*_u = \theta_\top$ given identifiability. Thus, $\theta^*_\lambda = \theta_\top$ for any $0 \leq \lambda \leq 1$.

Additionally, using White's results [45], $A(\theta^*_\lambda) = -B(\theta^*_\lambda) = \mathbf{I}(\theta^*_\lambda)$, where $\mathbf{I}()$ denotes the Fisher information matrix. Thus, the Fisher information matrix can be written as:

$$\mathbf{I}(\theta) = \lambda \mathbf{I}_l(\theta) + (1 - \lambda)\mathbf{I}_u(\theta), \tag{3}$$

which matches the derivations made by Zhang and Oles [48]. The significance of Expression (3) is that it allows the use of the Cramer-Rao lower bound (CRLB) on the covariance of a consistent estimator:

$$\text{Cov}(\hat{\theta}_N) \geq \frac{1}{N}(\mathbf{I}(\theta))^{-1} \tag{4}$$

where N is the number of data (both labeled and unlabeled) and $\text{Cov}(\hat{\theta}_N)$ is the estimator's covariance matrix with N samples.

Consider the Taylor expansion of the classification error around $\theta_\top$, as suggested by Shahshahani and Landgrebe [41], linking the decrease in variance associated with unlabeled data to a decrease in classification error, and assume the existence of necessary derivatives:

$$\mathbf{e}(\hat{\theta}) \approx \mathbf{e}_{\text{B}} + \left.\frac{\partial \mathbf{e}(\theta)}{\partial \theta}\right|_{\theta_\top} \left(\hat{\theta} - \theta_\top\right) + \frac{1}{2}\text{tr}\left(\left.\frac{\partial^2 \mathbf{e}(\theta)}{\partial \theta^2}\right|_{\theta_\top} \left(\hat{\theta} - \theta_\top\right)\left(\hat{\theta} - \theta_\top\right)^T\right). \tag{5}$$

Take expected values on both sides. Asymptotically the expected value of the second term in the expansion is zero, as maximum likelihood estimators are asymptotically unbiased when the model is correct. Shahshahani and Landgrebe [41] thus argue that

$$E\Big[\mathbf{e}(\hat{\theta})\Big] \approx \mathbf{e}_{\text{B}} + (1/2)\text{tr}\left((\partial^2\mathbf{e}(\theta)/\partial\theta^2)|_{\theta_\top}\text{Cov}(\hat{\theta})\right)$$

where $\mathbf{e}_{\text{B}} = \mathbf{e}(\theta_\top)$ is the Bayes error rate. They also show that if $\text{Cov}(\theta') \geq \text{Cov}(\theta'')$ for some θ' and θ'', then the second term in the approximation is larger for θ' than for θ''. Because $\mathbf{I}_u(\theta)$ is always positive definite, $\mathbf{I}_l(\theta) \leq \mathbf{I}(\theta)$. Thus, using the Cramer-Rao lower bound (Expression (4)) the covariance with labeled and unlabeled data is smaller than the covariance with just labeled data, leading to the conclusion that *unlabeled data must cause a reduction in classification error when the model is correct*. It should be noted that this argument holds as the number of records goes to infinity, and is an approximation for finite values.

3.2 Model Is Incorrect

A more realistic scenario desribed in detail in [9] is when the distribution $P(C, \mathbf{X})$ does not belong to the family of distributions $P(C, \mathbf{X}|\theta)$. In view of Theorem 1, it is clear that unlabeled data can have the deleterious effect observed occasionally in the literature. Suppose that $\theta_u^* \neq \theta_l^*$ and that $\mathbf{e}(\theta_u^*) > \mathbf{e}(\theta_l^*)$ (for the difficulties in estimating $\mathbf{e}(\theta_u^*)$ and a solution for this please see [9]). If a large number of labeled samples is observed, the classification error is approximated by $\mathbf{e}(\theta_l^*)$. If we then have more samples, most of which unlabeled, we eventually reach a point where the classification error approaches $\mathbf{e}(\theta_u^*)$. So, the net result is that we started with classification error close to $\mathbf{e}(\theta_l^*)$, and by adding a large number of unlabeled samples, classification performance degraded (see again [9] for more details). The basic fact here is that estimation and classification bias are affected differently by different values of λ. Hence, a necessary condition for this kind of performance degradation is that $\mathbf{e}(\theta_u^*) \neq \mathbf{e}(\theta_l^*)$; a sufficient condition is that $\mathbf{e}(\theta_u^*) > \mathbf{e}(\theta_l^*)$.

The focus on asymptotics is adequate as we want to eliminate phenomena that can vary from dataset to dataset. If $\mathbf{e}(\theta_l^*)$ is smaller than $\mathbf{e}(\theta_u^*)$, then a large enough labeled dataset can be dwarfed by a much larger unlabeled dataset — the classification error using the whole dataset can be larger than the classification error using the labeled data only.

3.3 Discussion

Despite the shortcomings of semi-supervised learning presented in the previous sections, we do not discourage its use. Understanding the causes of performance degradation with unlabeled data motivates the exploration of new methods attempting to use positively the available unlabeled data. Incorrect modeling assumptions in Bayesian networks culminate mainly as discrepancies in the graph structure, signifying incorrect independence assumptions among variables. To eliminate the increased bias caused by the addition of unlabeled data we can try simple solutions,

such as model switching (Section 4.2) or attempt to learn better structures. We describe likelihood based structure learning methods (Section 4.3) and a possible alternative: classification driven structure learning (Section 4.4). In cases where relatively mild changes in structure still suffer from performance degradation from unlabeled data, there are different approaches that can be taken: discard the unlabeled data, give them a different weight (Section 4.5), or use the alternative of actively labeling some of the unlabeled data (Section 4.6).

To summarize, the main conclusions that can be derived from our analysis are:

- Labeled and unlabeled data contribute to a reduction in variance in semi-supervised learning under maximum likelihood estimation. *This is true regardless of whether the model is correct or not*.
- If the model is correct, the maximum likelihood estimator is unbiased and both labeled and unlabeled data contribute to a reduction in classification error by reducing variance.
- If the model is incorrect, there may be different asymptotic estimation biases for different values of λ (the ratio between the number of labeled and unlabeled data). Asymptotic classification error may also be different for different values of λ. An increase in the number of unlabeled samples may lead to a larger bias from the true distribution and a larger classification error.

In the next section, we discuss several possible solutions for the problem of performance degradation in the framework of Bayesian network classifiers.

4 Learning the Structure of Bayesian Network Classifiers

The conclusion of the previous section indicates the importance of obtaining the correct structure when using unlabeled data in learning a classifier. If the correct structure is obtained, unlabeled data improve the classifier; otherwise, unlabeled data can actually degrade performance. Somewhat surprisingly, the option of searching for better structures was not proposed by researchers that previously witnessed the performance degradation. Apparently, performance degradation was attributed to unpredictable, stochastic disturbances in modeling assumptions, and not to mistakes in the underlying structure – something that can be detected and fixed.

4.1 Bayesian Networks

Bayesian Networks [36] are tools for modeling and classification. A Bayesian Network (BN) 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 Π_i denotes the parents of X_i in the graph. The joint probability distribution is factored to the collection of conditional probability distributions of each node in the graph as:

$$p(X_1, ..., X_n) = \prod_{i=1}^{n} p(X_i|\Pi_i)\,. \tag{6}$$

The directed acyclic graph is the *structure*, and the distributions $p(X_i|\Pi_i)$ represent the *parameters* of the network. We say that the assumed structure for a network, S', is *correct* when it is possible to find a distribution, $p(C, \mathbf{X}|S')$, that matches the distribution that generates data, $p(C, \mathbf{X})$; otherwise, the structure is *incorrect*. In the above notations, $\mathbf{X}$ is an incoming vector of features. The classifier receives a record $\mathbf{x}$ and generates a label $\hat{c}(\mathbf{x})$. An optimal classification rule can be obtained from the exact distribution $p(C, \mathbf{X})$ which represents the a-posteriori probability of the class given the features.

Maximum likelihood estimation is one of the main methods to learn the parameters of the network. When there are missing data in training set, the Expectation Maximization (EM) algorithm [15] can be used to maximize the likelihood.

As a direct consequence of the analysis in Section 3, a Bayesian network that has the correct structure and the correct parameters is also optimal for classification because the a-posteriori distribution of the class variable is accurately represented (see [9] for a detailed analysis on this issue). As pointed out in [9] and [8] to solve the problem of performance degradation in BNs, there is a need to carefull analyze the structure of the BN classifier used in the classification.

4.2 Switching between Simple Models

One attempt to overcome the performance degradation from unlabeled data could be to switch models as soon as degradation is detected. Suppose that we learn a classifier with labeled data only and we observe a degradation in performance when the classifier is learned with labeled and unlabeled data. We can switch to a more complex structure at that point. An interesting idea is to start with a Naive Bayes classifier in which the features are assumed independent given the class. If performance degrades with unlabeled data, switch to a different type of Bayesian Network classifier, namely the Tree-Augmented Naive Bayes classifier (TAN) [21].

In the TAN classifier structure the class node has no parents and each feature has the class node and at most one other feature as parents, such that the result is a tree structure for the features. Learning the most likely TAN structure has an efficient and exact solution [21] using a modified Chow-Liu algorithm [7]. Learning the TAN classifiers when there are unlabeled data requires a modification of the original algorithm to what we named the EM-TAN algorithm [10].

If the correct structure can be represented using a TAN structure, this approach will indeed work. However, even the TAN structure is only a small set of all possible structures. Moreover, as the examples in the experimental section show, switching from NB to TAN does not guarantee that the performance degradation will not occur.

Very relevant is the research of Baluja [1]. The author uses labeled and unlabeled data in a probabilistic classifier framework to detect the orientation of a face. In his results, he obtained excellent classification results, but there were cases where unlabeled data degraded performance. As a consequence, he decided to switch from a Naive Bayes approach to more complex models. Following this intuitive direction, we explain Baluja's observations and provide a solution to the problem: structure learning.

4.3 Beyond Simple Models

A different approach to overcome performance degradation is to learn the structure of the Bayesian network without restrictions other than the generative one[3]. There are a number of such algorithms in the literature (among them [20, 3, 6]). Nearly all structure learning algorithms use the 'likelihood based' approach. The goal is to find structures that best fit the data (with perhaps a prior distribution over different structures). Since more complicated structures have higher likelihood scores, penalizing terms are added to avoid overfiting to the data, e.g, the minimum description length (MDL) term. The difficulty of structure search is the size of the space of possible structures. With finite amounts of data, algorithms that search through the space of structures maximizing the likelihood, can lead to poor classifiers because the a-posteriori probability of the class variable could have a small effect on the score [21]. Therefore, a network with a higher score is not necessarily a better classifier. Friedman et al. [21] suggest changing the scoring function to focus only on the posterior probability of the class variable, but show that it is not computationally feasible.

The drawbacks of likelihood based structure learning algorithms could be magnified when learning with unlabeled data; the posterior probability of the class has a smaller effect during the search, while the marginal of the features would dominate. Therefore, we decided to take a different approach presented in the next section.

4.4 Classification Driven Stochastic Structure Search

As pointed out in [8] one ellegant solution is to find the structure that minimizes the probability of classification error directly. To do so the classification driven stochastic search algorithm (SSS) was proposed in [9]. The basic idea of this approach is that, since one is interested in finding a structure that performs well as a classifier, it is natural to design an algorithm that use classification error as the guide for structure learning. For completness we summarize the main observation here and we direct the interested reader to [8] for a complete analysis.

One important observation is that unlabeled data can indicate incorrect structure through degradation of classification performance. Additionally, we also saw previously that classification performance improves with the correct structure. As a consequence, a structure with higher classification accuracy over another indicates an improvement towards finding the optimal classifier.

To learn structure using classification error, it is necessary to adopt a strategy for efficiently searching through the space of all structures while avoiding local maxima. As there is no simple closed-form expression that relates structure with classification error, it is difficult to design a gradient descent algorithm or a similar iterative method which would be in any case prone to find local minima due to the size of the search space.

In [8] the following measure was proposed to be maximized:

[3] A Bayesian network classifier is a *generative* classifier when the class variable is an ancestor (e.g., parent) of some (or all) features.

Definition 1. *The* inverse error measure *for structure S' is*

$$inv_e(S') = \frac{\frac{1}{p_{S'}(\hat{c}(X)\neq C)}}{\sum_S \frac{1}{p_S(\hat{c}(\mathbf{X})\neq C)}}, \tag{7}$$

where the summation is over the space of possible structures and $p_S(\hat{c}(\mathbf{X}) \neq C)$ is the probability of error of the best classifier learned with structure S.

Metropolis-Hastings sampling [30] can be used to generate samples from the inverse error measure, without the need to compute it for all possible structures. For constructing the Metropolis-Hastings sampling, a neighborhood of a structure is defined as the set of directed acyclic graphs to which we can transit in the next step. Transition is done using a predefined set of possible changes to the structure; at each transition a change consists of a single edge addition, removal, or reversal. In [8] the acceptance probability of a candidate structure, S_{new}, to replace a previous structure, S_t is defined as follows:

$$\min\left(1, \left(\frac{inv_e(S^{new})}{inv_e(S^t)}\right)^{1/T} \frac{q(S^t|S^{new})}{q(S^{new}|S^t)}\right) = \min\left(1, \left(\frac{p^t_{error}}{p^{new}_{error}}\right)^{1/T} \frac{N_t}{N_{new}}\right) \tag{8}$$

where $q(S'|S)$ is the transition probability from S to S' and N_t and N_{new} are the sizes of the neighborhoods of S_t and S_{new}, respectively; this choice corresponds to equal probability of transition to each member in the neighborhood of a structure. This choice of neighborhood and transition probability creates a Markov chain which is aperiodic and irreducible, thus satisfying the Markov chain Monte Carlo (MCMC) conditions [27].

The parameter T is used as a temperature factor in the acceptance probability. As such, T close to 1 would allow acceptance of more structures with higher probability of error than previous structures. T close to 0 mostly allows acceptance of structures that improve probability of error. A fixed T amounts to changing the distribution being sampled by the MCMC, while a decreasing T is a simulated annealing run, aimed at finding the maximum of the inverse error measures. The rate of decrease of the temperature determines the rate of convergence. Asymptotically in the number of data, a logarithmic decrease of T guarantees convergence to a global maximum with probability that tends to one [22].

The SSS algorithm, with a logarithmic cooling schedule T, can find a structure that is close to minimum probability of error. The estimate of the classification error of a given structure is obtained by using the labeled training data. Therefore, to avoid overfitting, a multiplicative penalty term is required. This penalty term, derived from the Vapnik-Chervonenkis (VC) bound on the empirical classification error, penalizes complex classifiers thus keeping the balance between bias and variance (for more details we refer the reader to [9]).

4.5 Should Unlabeled Be Weighed Differently?

An interesting strategy, suggested by Nigam et al. [31] is to change the weight of the unlabeled data (reducing their effect on the likelihood). The basic idea in Nigam et al's estimators is to produce a modified log-likelihood that is of the form:

$$\lambda' L_l(\theta) + (1 - \lambda') L_u(\theta) \tag{9}$$

where $L_l(\theta)$ and $L_u(\theta)$ are the likelihoods of the labeled and unlabeled data, respectively. For a sequence of λ', maximize the modified log-likelihood functions to obtain $\hat{\theta}_{\lambda'}$ ($\hat{\theta}$ denotes an estimate of θ), and choose the best one with respect to cross-validation or testing. This estimator is simply modifying the ratio of labeled to unlabeled samples for any fixed λ'. Note that this estimator can only make sense under the assumption that the model is incorrect. Otherwise, both terms in Expression (9) lead to unbiased estimators of θ.

Our experiments in [8] suggest that there is then no reason to impose different weights on the data, and much less reason to search for the best weight, when the differences are solely in the rate of reduction of variance. Presumably, there are a few labeled samples available and a large number of unlabeled samples; why should we increase the importance of the labeled samples, giving more weight to a term that will contribute more heavily to the variance?

4.6 Active Learning

All the methods presented above consider a "passive" use of unlabeled data. A different approach is known as active learning, in which an oracle is queried as to the label of some of the unlabeled data. Such an approach increases the size of the labeled data set, reduces the classifier's variance, and thus reduces the classification error. There are different ways to choose which unlabeled data to query. The straightforward approach is to choose a sample randomly. This approach ensures that the data distribution $p(C, \mathbf{X})$ is unchanged, a desirable property when estimating generative classifiers. However, the random sample approach typically requires many more samples to achieve the same performance as methods that choose to label data close to the decision boundary. We note that, for generative classifiers, the latter approach changes the data distribution therefore leading to estimation bias. Nevertheless, McCallum and Nigam [29] used active learning with generative models with success. They proposed to first actively query some of the labeled data followed by estimation of the model's parameters with the remainder of the unlabeled data.

We performed extensive experiments in [8]. Here we present only the main conclusions. With correctly specified generative models and a large pool of unlabeled data, "passive" use of the unlabeled data is typically sufficient to achieve good performance. Active learning can help reduce the chances of numerical errors (improve EM starting point, for example), and help in the estimation of classification error. With incorrectly specified generative models, active learning is very profitable in quickly reducing the error, while adding the remainder of unlabeled data might not be desirable.

4.7 Summary

The idea of structure search is particularly promising when unlabeled data are present. It seems that simple heuristic methods, such as the solution proposed by Nigam et al. [31] of weighing down the unlabeled data, are not the best strategies for unlabeled data. We suggest that structure search, and in particular stochastic structure search, holds the most promise for handling large amount of unlabeled data and relatively scarce labeled data for classification. We also believe that the success of structure search methods for classification increases significantly the breadth of applications of Bayesian networks.

In a nutshell, when faced with the option of learning with labeled and unlabeled data, our discussion suggests following the following path. Start with Naive Bayes and TAN classifiers, learn with only labeled data and test whether the model is correct by learning with the unlabeled data, using EM and EM-TAN. If the result is not satisfactory, then SSS can be used to attempt to further improve performance with enough computational resources. If none of the methods using the unlabeled data improve performance over the supervised TAN (or Naive Bayes), active learning can be used, as long as there are resources to label some samples.

5 Experiments

For the experiments, we used our real time facial expression recognition system [10]. This is composed of a face detector which is used as an input to a facial feature detection module. Using the extracted facial features, a face tracking algorithm outputs a vector of motion features of certain regions of the face. The features are used as inputs to a Bayesian network classifier.

The face tracking we use in our system is based on a system developed by Tao and Huang [42] called the piecewise Bézier volume deformation (PBVD) tracker. The face tracker uses a model-based approach where an explicit 3D wireframe model of the face is constructed. A generic face model is then warped to fit the detected facial features. The face model consists of 16 surface patches embedded in Bézier volumes. The surface patches defined in 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. A snap shot of the system, with the face tracking and the corresponding recognition result is shown in Figure 1.

In Section 5.1, we start by investigating the use Bayesian network classifiers learned with labeled and unlabeled data for face detection. We present our results on two standard databases and show good results even if we use a very small set of labeled data. Subsequently, in Section 5.2, we present our facial feature detection module which uses the input given from the face detector and outputs the location of relevant facial features. Finally, in Section 5.3, we discuss the facial expression recognition results obtained by incorporating the facial feature detected inside the PBVD tracker.

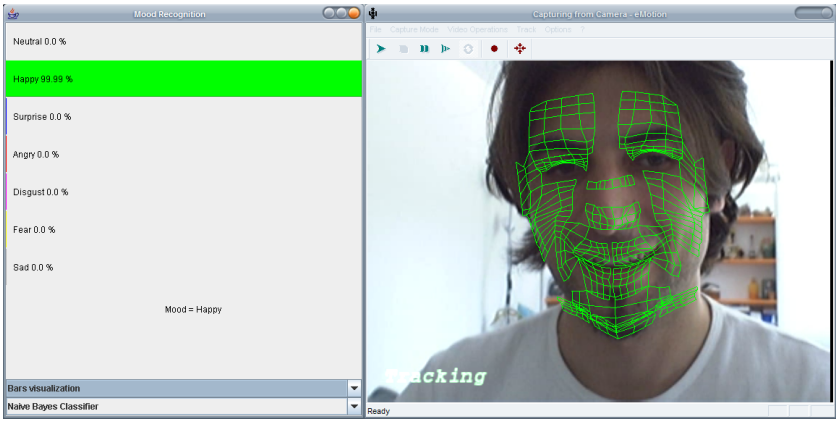


Fig. 1. A snap shot of our realtime facial expression recognition system. On the left side is a wireframe model overlayed on a face being tracked. On the right side the correct expression, Happy, is detected (the bars show the relative probability of Happy compared to the other expressions). The subject shown is from the Cohn-Kanade database.

5.1 Face Detection Experiments

In our face detection experiments we propose to use Bayesian network classifiers, with the image pixels of a predefined window size as the features in the Bayesian network. Among the different works, those of Colmenarez and Huang [11] and Wang et al. [44] are more related to the Bayesian network classification methods for face detection. Both learn some 'structure' between the facial pixels and combine them to a probabilistic classification rule. Both use the entropy between the different pixels to learn pairwise dependencies.

Our approach in detecting faces is an appearance based approach, where the intensity of image pixels serve as the features for the classifier. In a natural image, faces can appear at different scales, rotations, and location. For learning and defining the Bayesian network classifiers, we must look at fixed size windows and learn how a face appears in such windows, where we assume that the face appears in most of the window's pixels.

The goal of the classifier is to determine if the pixels in a fixed size window are those of a face or non-face. While faces are a well defined concept, and have a relatively regular appearance, it is harder to characterize non-faces. We therefore model the pixel intensities as discrete random variables, as it would be impossible to define a parametric probability distribution function (pdf) for non-face images. For 8-bit representation of pixel intensity, each pixel has 256 values. Clearly, if all these values are used for the classifier, the number of parameters of the joint distribution is too large for learning dependencies between the pixels (as is the case of TAN classifiers). Therefore, there is a need to reduce the number of values representing pixel intensity. Colmenarez and Huang [11] used 4 values per pixel using fixed and equal bin sizes. We use non-uniform discretization using the class conditional entropy as

the mean to bin the 256 values to a smaller number. We use the MLC++ software for that purpose as is described in [16].

Note that our methodology can be extended to other face detection methods which use different features. The complexity of our method is $O(n)$, where n is the number of features (pixels in our case) considered in each image window.

We test the different approaches described in Section 4, with both labeled and unlabeled data. For training the classifier we used a dataset consisting of 2,429 faces and 10,000 non-faces obtained from the MIT CBCL Face database #1[4]. Examples of face images from the database are presented in Figure 2. Each face image is cropped and resampled to a 19×19 window, thus we have a classifier with 361 features. We also randomly rotate and translate the face images to create a training set of 10,000 face images. In addition we have available 10,000 non-face images. We leave out 1,000 images (faces and non-faces) for testing and train the Bayesian network classifiers on the remaining 19,000. In all the experiments we learn a Naive Bayes, TAN, and a general generative Bayesian network classifier, the latter using the SSS algorithm.

Fig. 2. Randomly selected face examples.

In Table 1 we summarize the results obtained for different algorithms and in the presence of increasing number of unlabeled data. We fixed the false alarm to 1%, 5%, and 10% and we computed the detection rates. We first learn using all the training data being labeled (that is $19,000$ labeled images). The classifier learned with the SSS algorithm outperforms both TAN and NB classifiers, and all perform quite well, achieving high detection rates with a low rate of false alarm. Next we remove the labels of some of the training data and train the classifiers. In the first case, we remove the labels of 97.5% of the training data (leaving only 475 labeled images).

[4] http://www.ai.mit.edu/projects/cbcl

Table 1. Detection rates (%) for various numbers of false positives

Detector		False positives 1%	5%	10%
NB	19,000 labeled	74.31	89.21	92.72
	475 labeled	68.37	86.55	89.45
	475 labeled + 18,525 unlabeled	66.05	85.73	86.98
	250 labeled	65.59	84.13	87.67
	250 labeled + 18,750 unlabeled	65.15	83.81	86.07
TAN	19,000 labeled	91.82	96.42	99.11
	475 labeled	86.59	90.84	94.67
	475 labeled + 18,525 unlabeled	85.77	90.87	94.21
	250 labeled	75.37	87.97	92.56
	250 labeled + 18,750 unlabeled	77.19	89.08	91.42
SSS	19,000 labeled	90.27	98.26	99.87
	475 labeled + 18,525 unlabeled	88.66	96.89	98.77
	250 labeled + 18,750 unlabeled	86.64	95.29	97.93
SVM	19,000 labeled	87.78	93.84	94.14
	475 labeled	82.61	89.66	91.12
	250 labeled	77.64	87.17	89.16

We see that the NB classifier using both labeled and unlabeled data performs very poorly. The TAN based only on the 475 labeled images and the TAN based on the labeled and unlabeled images are close in performance, thus there was no significant degradation of performance when adding the unlabeled data. When only 250 labeled data are used (the labels of about 98.7% of the training data were removed), NB with both labeled and unlabeled data performs poorly, while SSS outperforms the other classifiers with no great reduction of performance compared to the previous cases. For benchmarking, we also implemented a SVM classifier (we used the implementation of Osuna et al. [34]). Note that this classifier starts off very good, but does not improve performance.

In summary, note that the detection rates for NB are lower than the ones obtained for the other detectors. Overall, the results obtained with SSS are the best. We see that even in the most difficult cases, there was sufficient amount of unlabeled data to achieve almost the same performance as with a large sized labeled dataset.

We also tested our system on the CMU test set [37] consisting of 130 images with a total of 507 frontal faces. The results are summarized in Table 2. Note that we obtained comparable results with the results obtained by Viola and Jones [43] and better than the results of Rowley et al. [37]. Examples of the detection results on some of the images of the CMU test are presented in Figure 3. We noticed similar failure modes as Viola and Jones [43]. Since, the face detector was trained only on frontal faces our system failes to detect faces if they have a significant rotation out of the plane (toward a profile view). The detector has also problems with the images in which the faces appear dark and the background is relatively light. Inevitably, we also detect false positive especially in some texture regions.

Fig. 3. Output of the system on some images of the CMU test using the SSS classifier learned with 19,000 labeled data. MFs represents the number of missed faces and FDs is the number of false detections.

5.2 Facial Feature Detection

In this section, we introduce a novel way to unify the knowledge of a face detector inside an active appearance model [12], using what we call a 'virtual structuring element', which limits the possible settings of the AAM in an appearance-driven

Table 2. Detection rates (%) for various numbers of false positives on the CMU test set.

Detector \ False positives		10%	20%
SSS	19,000 labeled	91.7	92.84
	475 labeled + 18,525 unlabeled	89.67	91.03
	250 labeled + 18,750 unlabeled	86.64	89.17
Viola-Jones [43]		92.1	93.2
Rowley et al. [37]		-	89.2

manner. We propose this visual artifact as a good solution for the background linking problems and respective generalization problems of basic AAMs.

The main idea of using an AAM approach is to learn the possible variations of facial features exclusively on a probabilistic and statistical basis of the existing observations (i.e., which relation holds in all the previously seen instances of facial features). This can be defined as a combination of shapes and appearances.

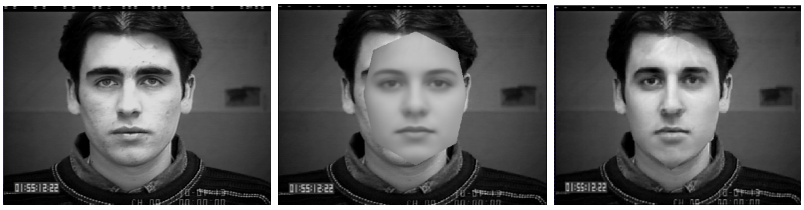
At the basis of AAM search is the idea to treat the fitting procedure of a combined shape-appearance model as an optimization problem in trying to minimize the difference vector between the image $\mathbf{I}$ and the generated model $\mathbf{M}$ of shape and appearance: $\delta\mathbf{I} = \mathbf{I} - \mathbf{M}$.

Cootes et al. [12] observed that each search corresponds to a similar class of problems where the initial and the final model parameters are the same. This class can be learned offline (when we create the model) saving high-dimensional computations during the search phase.

Learning the class of problems means that we have to assume a relation $\mathbf{R}$ between the current error image $\delta\mathbf{I}$ and the needed adjustments in the model parameters m. The common assumption is to use a linear relation: $\delta m = \mathbf{R}\delta\mathbf{I}$. Despite the fact that more accurate models were proposed [28], the assumption of linearity was shown to be sufficiently accurate to obtain good results. To find $\mathbf{R}$ we can conduct a series of experiments on the training set, where the optimal parameters m are known. Each experiment consists of displacing a set of parameters by a know amount and in measuring the difference between the generated model and the image under it. Note that when we displace the model from its optimal position and we calculate the error image $\delta\mathbf{I}$, the image will surely contain parts of the background.

What remains to discuss is an iterative optimization procedure that uses the found predictions. The first step is to initialize the mean model in an initial position and the parameters within the reach of the parameter prediction range (which depends on the perturbation used during training). Iteratively, a sample of the image under the initialization is taken and compared with the model instance. The differences between the two appearances are used to predict the set of parameters that would perhaps improve the similarity. In case a prediction fails to improve the similarity, it is possible to damp or amplify the prediction several times and maintain the one with the best result. For an overview of some possible variations to the original AAMs

algorithm refer to [13]. An example of the AAM search is shown in Fig. 4 where a model is fitted to a previously unseen face.



(a) Unseen face (b) Initialization (c) Converged model

Fig. 4. Results of an AAM search on an unseen face

One of the main drawbacks of the AAM is coming from its very basic concept: when the algorithm learns how to solve the optimization offline, the perturbation applied to the model inevitably takes parts of the background into account. This means that instead of learning how to generally solve the class of problems, the algorithm actually learns how to solve it only for the same or similar background. This makes AMMs domain-specific, that is, the AAM trained for a shape in a predefined environment has difficulties when used on the same shape immersed in a different environment. Since we always need to perturbate the model and to take into account the background, an often used idea is to constrain the shape deformation within predefined boundaries. Note that a shape constraint does not adjust the deformation, but will only limit it when it is found to be invalid.

To overcome these deficiencies of AAMs, we propose a novel method to visually integrate the information obtained by a face detector inside the AAM. This method is based on the observation that an object with a specific and recognizable feature would ease the successful alignment of its model. As the face detector we can choose between the one proposed by Viola and Jones [43] and the one presented in Section 5.1.

Since faces have many highly relevant features, erroneously located ones could lead the optimization process to converge to local minima. The novel idea is to add a virtual artifact in each of the appearances in the training and the test sets, that would inherently prohibit some deformations. We call this artifact a **virtual structuring element** (or **VSE**) since it adds structure in the data that was not available otherwise. In our specific case, this element adds visual information about the position of the face. If we assume that the face detector successfully detects a face, we can use that information to build this artifact.

After experimenting with different VSEs, we propose the following guideline to choose a good VSE. We should choose a VSE that: (1) Is big enough to steer the optimization process; (2) Does not create additional uncertainty by covering relevant features (e.g., the eyes or nose); (3) Scales accordingly to the dimension of the de-

tected face; and (4) Completely or partially removes the high variance areas in the model (e.g., background) with uniform ones.

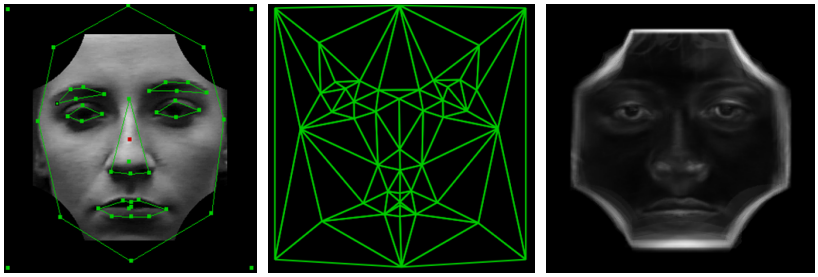


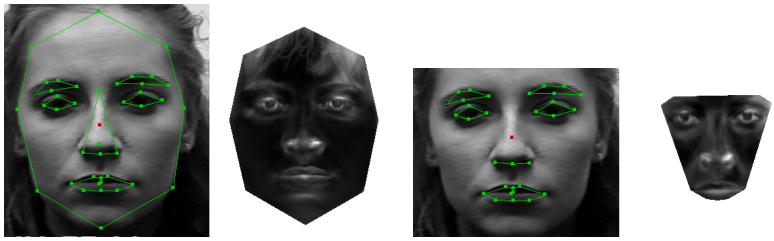
Fig. 5. The effect of a virtual structuring element to the annotation, appearance, and variance (white indicates a larger variance)

In the used VSE, a black frame with width equal to 20% of the size of the detected face is built around the face itself. Besides the regular markers that capture the facial features (see Fig. 5 and [10] for details) four new markers are added in the corners to stretch the convex hull of the shape to take in consideration the structuring element. Around each of those four points, a black circle with the radius of one third of the size of the face is added. The resulting annotation, shape, and appearance variance are displayed in Fig. 5. Note that in the variance map the initialization variance of the face detector is automatically included in the model (i.e., the thick white border delimitating the VSE).

This virtual structuring element visually passes information between the face detection and the AAM. We show in the experiments that VSE helps the basic AAMs in the model generalization and fitting performances.

Two datasets were used during the evaluation: (1) a part of the Cohn-Kanade [25] dataset consisting of 53 male and female subjects, showing neutral frontal faces in a controlled environment; (2) the Unilever dataset consinsting of 50 females, showing natural poses in an outdoor uncontrolled environment. The idea is to investigate the influence of the VSE when the background is unchanged (Cohn-Kanade) and when more difficult conditions are present (Unilever).

We evaluate two specific annotations, one named ‘relevant’ (Fig. 6(a)) describing the facial features that are relevant for the facial expression classifiers including the face contours that are needed for face tracking, and the other one named ‘inside’ (Fig. 6(b)) describing the facial features without the face contours. Note that the ‘inside’ model is surrounded only by face area (so not by not by background) so its variance is lower and the model is more robust. To assess the performance of the AAM we initialize the mean model (i.e., the mean shape with the mean appearance) shifted in the Cartesian plane with a predefined amount. This simulates some extremes in the initialization error obtained by the face detector.



(a) Relevant (b) Inside

Fig. 6. The annotations and their respective variance maps over the datasets

The common approach to assess performance of AAM is to compare the results to a ground truth (i.e., the annotations in the training set). The following measures are used: **Point to Point Error** is the Euclidean distance between each point of the true shape and the corresponding fitted shape; **Point to Curve Error** is the Euclidean distance between a fitted shape point and the closest point on the linear spline obtained from the true shape points; and **Mahalanobis Distance** defined as:

$$D^2 = \sum_{i=1}^{t} \frac{m_i^2}{\lambda_i} \tag{10}$$

where m_i represents the AAM parameters and λ_i their respective principal components.

We perform two types of experiments. In the person independent case we perform a leave-one-out cross validation. For the second experiment, the Generalized AAM test, we merge the two datasets and we create a model which includes all the different lighting conditions, backgrounds, subject features, and annotations (together with their respective errors). The goal of this experiment is to test whether the generalization problems of AAMs could be solved just by using a greater amount of training data.

	Cohn-Kanade			Unilever		
	Point-Point	Point-Curve	Mahalanobis	Point-Point	Point-Curve	Mahalanobis
Relev.	16.72 (5.53)	9.09 (3.36)	47.93 (4.90)	54.84 (10.58)	29.82 (6.22)	79.41 (6.66)
Relev. VSE	6.73 (0.21)	4.34 (0.15)	26.46 (1.57)	10.14 (2.07)	6.53 (1.30)	24.75 (3.57)
Inside	9.53 (3.48)	6.19 (2.47)	39.55 (3.66)	25.98 (7.29)	17.69 (5.16)	38.20 (4.52)
Inside VSE	5.85 (0.24)	3.76 (0.13)	27.14 (1.77)	8.99 (1.90)	6.37 (1.46)	23.45 (2.81)

Table 3. Mean and standard error in the person independent test for the two datasets

Table 3 shows the results obtained for the two datasets in the person independent experiment. Important to notice that the results obtained with Cohn-Kanade datasets are in most of the cases better than the one obtained with the Unilever dataset. This

has to do with the fact that, in the Unilver dataset, the effect of the uncontrolled lighting condition and background change is more relevant and the model fitting is more difficult. However, in both cases one can see that the use of VSE improved significantly the results. Another important aspect is that the use of VSE is more effective in the case of Unilever database and this is because the VSE is reducing the background influence to a larger extend. Interesting to note is that, while the use of a VSE does not excessively improve the accuracy of the 'inside' model, the use of VSE on the 'relevant' model drastically improves its accuracy making it even better than the basic 'inside' model. This result is surprising since in the 'relevant' model parts of the markers are covered by the VSE (i.e., the forehead and chin markers) we expected the final model to inherently generate some errors. Instead, it seems that the inner parts of the face might steer the outer markers to the optimal position. This could only mean that there is a proportional relation between the facial countours and the inside features, which is a very interesting and unexpected property.

In the generalized AAM experiment (see Table 4), we notice that the results are generally worse when compared with the person independent results on the 'controlled' Cohn-Kanade dataset, but better when compared with the same experiment on the 'uncontrolled' Unilever dataset. Also in this case the VSE implementation shows very good improvements over the basic AAM implementation. What is important to note is that the VSE implementation brings the results of the generalized AAM very close to the dataset specific results, improving the generalization of basic AAM.

	Generalized AAM		
	Point-Point	Point-Curve	Mahalanobis
Relevant	21.05 (0.79)	8.45 (0.27)	116.22 (3.57)
Relevant VSE	8.50 (0.20)	5.38 (0.12)	51.11 (0.91)
Inside	8.11 (0.21)	4.77 (0.10)	85.22 (1.98)
Inside VSE	7.22 (0.17)	4.65 (0.09)	52.84 (0.96)

Table 4. Mean and standard error for Generalized AAM

While the 'relevant VSE' model is better than the normal 'inside' model, the 'inside VSE' is the model of choice to obtain the best overall results on facial features detection. In our specific task, we can use the 'inside VSE' model to obtain the best results but we will additionally need some heuristics to correctly position the other markers which are not included in the model. These missing markers are relevant for robust face tracking and implicitly for facial expression classification so their accurate positioning is very important. Since in the case of 'inside VSE' model these markers are not detected explicitly, we indicate the 'relevant VSE' model as the best choice for our purposes.

To better illustrate the effect of using a VSE, Fig. 7 shows an example of the difference in the results when using a 'relevant' model and a 'relevant VSE' model. While the first failed to correctly converge, the second result is optimal for inner

(a) Relevant (b)Relevant VSE

Fig. 7. An example of the difference in the results between a 'relevant' and a 'relevant VSE' model

facial features. Empirically, VSE models showed to always overlap to the correct annotation, avoiding the mistakes generated by unsuccessful alignments like the one in Fig. 7(a).

5.3 Facial Expression Recognition Experiments

As mentioned previously, our system uses a generic face model consisting of 16 surface patches embedded in Bézier volumes which is warped to fit the detected facial features. This model is used for tracking the detected facial features. The recovered motions are represented in terms of magnitudes of some predefined motion of the 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 [17], and are numeric in nature, representing not only the activation of a facial region, but also the direction and intensity of the motion. The 12 MU's used in the face tracker are shown in Figure 8. The MU's are used as the features for the Bayesian network classifiers learned with labeled and unlabeled data.

There are seven categories of facial expressions corresponding to *neutral*, *joy*, *surprise*, *anger*, *disgust*, *sad*, and *fear*. For testing we use two databases, in which all the data is labeled. We removed the labels of most of the training data and learned the classifiers with the different approaches discussed in Section 4.

The first database was collected by Chen and Huang [5] and is a database of subjects that were instructed to display facial expressions corresponding to the six types of emotions. All the tests of the algorithms are performed on a set of five people, each one displaying six sequences of each one of the six emotions, starting and ending at the Neutral expression. The video sampling rate was 30 Hz, and a typical emotion sequence is about 70 samples long ($\sim$2s). The second database is the Cohn-Kanade database [25] introduced in the previous section. For each subject

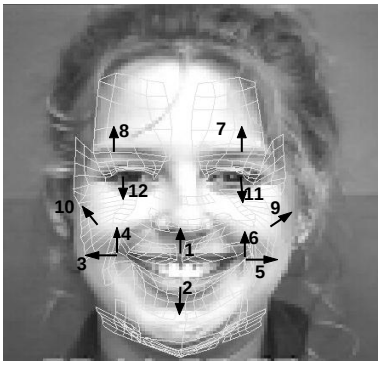


Fig. 8. The facial motion measurements

Table 5. The experimental setup and the classification results for facial expression recognition with labeled data (L) and labeled + unlabeled data (LUL). Accuracy is shown with the corresponding 95% confidence interval.

Dataset	Train		Test	NB-L	NB-LUL	TAN-L	TAN-LUL	SSS-LUL
	# lab.	# unlab.						
Chen-Huang	300	11,982	3,555	71.25±0.75%	58.54±0.81%	72.45±0.74%	62.87±0.79%	74.99±0.71%
Cohn-Kanade	200	2,980	1,000	72.50±1.40%	69.10±1.44%	72.90±1.39%	69.30±1.44%	74.80±1.36%

there is at most one sequence per expression with an average of 8 frames for each expression.

We measure the accuracy with respect to the classification result of each frame, where each frame in the video sequence was manually labeled to one of the expressions (including Neutral). The results are shown in Table 5, showing classification accuracy with 95% confidence intervals. We see that the classifier trained with the SSS algorithm improves classification performance to about 75% for both datasets. Model switching from Naive Bayes to TAN does not significantly improve the performance; apparently, the increase in the likelihood of the data does not cause a decrease in the classification error. In both the NB and TAN cases, we see a performance degradation as the unlabeled data are added to the smaller labeled dataset (TAN-L and NB-L compared to TAN-LUL and NB-LUL). An interesting fact arises from learning the same classifiers with all the data being labeled (i.e., the original database without removal of any labels). Now, SSS achieves about 83% accuracy, compared to the 75% achieved with the unlabeled data. Had we had more unlabeled data, it might have been possible to achieve similar performance as with the fully labeled database. This result points to the fact that labeled data are more valuable than unlabeled data (see [4] for a detailed analysis).

6 Conclusion

In this work we presented a complete system that aimes at human-computer interaction applications. We considered several instances of Bayesian networks and we showed that learning the structure of Bayesian networks classifiers enables learning good classifiers with a small labeled set and a large unlabeled set.

Our discussion of semi-supervised learning for Bayesian networks suggests the following path: when faced with the option of learning Bayesian networks with labeled and unlabeled data, start with Naive Bayes and TAN classifiers, learn with only labeled data and test whether the model is correct by learning with the unlabeled data. If the result is not satisfactory, then SSS can be used to attempt to further improve performance with enough computational resources. If none of the methods using the unlabeled data improve performance over the supervised TAN (or Naive Bayes), either discard the unlabeled data or try to label more data, using active learning for example.

In closing, it is possible to view some of the components of this work independently of each other. The theoretical results of Section 3 do not depend on the choice of probabilistic classifier and can be used as a guide to other classifiers. Structure learning of Bayesian networks is not a topic motivated solely by the use of unlabeled data. The three applications we considered could be solved using classifiers other than Bayesian networks. However, this work should be viewed as a combination of all three components; (1) the theory showing the limitations of unlabeled data is used to motivate (2) the design of algorithms to search for better performing structures of Bayesian networks and finally, (3) the successful applications to an human-computer interaction problem we are interested in solving by learning with labeled and unlabeled data.

Acknowledgments

We would like to thank Marcelo Cirelo, Fabio Cozman, Ashutosh Garg, and Thomas Huang for their suggestions, discussions, and critical comments. This work was supported by the Muscle NoE and MIAUCE European projects.

References

1. S. Baluja. Probabilistic modelling for face orientation discrimination: Learning from labeled and unlabeled data. In *Neural Information and Processing Systems*, pages 854–860, 1998.
2. M.J. Black and Y. Yacoob. Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion. In *Proc. International Conf. Computer Vision*, pages 374–381, 1995.
3. M. Brand. An entropic estimator for structure discovery. In *Neural Information and Processing Systems*, pages 723–729, 1998.

4. V. Castelli. *The relative value of labeled and unlabeled samples in pattern recognition*. PhD thesis, Stanford, 1994.
5. L.S. Chen. *Joint processing of audio-visual information for the recognition of emotional expressions in human-computer interaction*. PhD thesis, University of Illinois at Urbana-Champaign, 2000.
6. J. Cheng, R. Greiner, J. Kelly, D.A. Bell, and W. Liu. Learning Bayesian networks from data: An information-theory based approach. In *The Artificial Intelligence Journal, Volume 137*, pages 43–90, 2002.
7. C.K. Chow and C.N. Liu. Approximating discrete probability distribution with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467, 1968.
8. I. Cohen. *Semi-supervised learning of classifiers with application to human computer interaction*. PhD thesis, University of Illinois at Urbana-Champaign, 2003.
9. I. Cohen, F. Cozman, N. Sebe, M. Cirello, and T.S. Huang. Semi-supervised learning of classifiers: Theory, algorithms, and their applications to human-computer interaction. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(12):1553–1567, 2004.
10. I. Cohen, N. Sebe, A. Garg, L. Chen, and T.S. Huang. Facial expression recognition from video sequences: Temporal and static modelling. *Computer Vision and Image Understanding*, 91(1-2):160–187, 2003.
11. A.J. Colmenarez and T.S. Huang. Face detection with information based maximum discrimination. In *IEEE Conference on Computer Vision and Pattern Recogntion*, pages 782–787, 1997.
12. T. Cootes, G. Edwards, and C. Taylor. Active appearance models. *PAMI*, 23(6):681–685, 2001.
13. T. Cootes and P. Kittipanya-ngam. Comparing variations on the active appearance model algorithm. In *BMVC*, pages 837–846., 2002.
14. T. Cootes, C. Taylor, D. Cooper, and J. Graham. Active shape models - Their training and application. *CCVIU*, 61(1):38–59, 1995.
15. A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.
16. J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *International Conference on Machine Learning*, pages 194–202, 1995.
17. P. Ekman and W.V. Friesen. *Facial Action Coding System: Investigator's Guide*. Consulting Psychologists Press, 1978.
18. I.A. Essa and A.P. Pentland. Coding, analysis, interpretation, and recognition of facial expressions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(7):757–763, 1997.
19. B. Fasel and J. Luettin. Automatic facial expression analysis: A survey. *Pattern Recognition*, 36:259–275, 2003.
20. N. Friedman. The Bayesian structural EM algorithm. In *Proc. Conference on Uncertainty in Artificial Intelligence*, pages 129–138, 1998.
21. N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2):131–163, 1997.
22. B. Hajek. Cooling schedules for optimal annealing. *Mathematics of operational research*, 13:311–329, 1988.
23. E. Hjelmas and B.K. Low. Face detection:A survey. *Computer Vision and Image Understanding*, 83:236–274, 2003.
24. M. Jones and T. Poggio. Multidimensional morphable models. In *ICCV*, pages 683–688, 1998.

25. T. Kanade, J.F. Cohn, and Y. Tian. Comprehesive database for facial expression analysis. In *International Conf. on Automatic Face and Gesture Recognition*, pages 46–53, 2000.
26. M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *IJCV*, 1(4):321–331, 1987.
27. D. Madigan and J. York. Bayesian graphical models for discrete data. *Int. Statistical Review*, 63:215–232, 1995.
28. I. Matthews and S. Baker. Active appearance models revisited. *IJCV*, 60(2):135–164, 2004.
29. A.K. McCallum and K. Nigam. Employing EM in pool-based active learning for text classification. In *International Conf. on Machine Learning*, pages 350–358, 1998.
30. N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculation by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.
31. K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39:103–134, 2000.
32. 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.
33. T.J. O'Neill. Normal discrimination with unclassified obseravations. *Journal of the American Statistical Association*, 73(364):821–826, 1978.
34. E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 130–136, 1997.
35. M. Pantic and L.J.M. Rothkrantz. Automatic analysis of facial expressions: The state of the art. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(12):1424–1445, 2000.
36. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
37. H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.
38. H. Schneiderman. Learning a restricted Bayesian network for object detection. In *CVPR*, pages 639–646, 2004.
39. S. Sclaroff and J. Isidoro. Active blobs. In *ICCV*, 1998.
40. N. Sebe, I. Cohen, F.G. Cozman, and T.S. Huang. Learning probabilistic classifiers for human-computer interaction applications. *ACM Multimedia Systems*, 10(6):484–498, 2005.
41. B. Shahshahani and D. Landgrebe. Effect of unlabeled samples in reducing the small sample size problem and mitigating the Hughes phenomenon. *IEEE Transactions on Geoscience and Remote Sensing*, 32(5):1087–1095, 1994.
42. H. Tao and T.S. Huang. Connected vibrations: A modal analysis approach to non-rigid motion tracking. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 735–740, 1998.
43. P. Viola and M.J. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 57(2), 2004.
44. R.R. Wang, T.S. Huang, and J. Zhong. Generative and discriminative face modeling for detection. In *Automatic Face and Gesture recognition*, 2002.
45. H. White. Maximum likelihood estimation of misspecified models. *Econometrica*, 50(1):1–25, 1982.
46. M.-H. Yang, D. Kriegman, and N. Ahuja. Detecting faces in images: A survey. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(1):34–58, 2002.

47. M.-H. Yang, D. Roth, and N. Ahuja. SNoW based face detector. In *Neural Information Processing Systems*, pages 855–861, 2000.
48. T. Zhang and F. Oles. A probability analysis on the value of unlabeled data for classification problems. In *International Conf. on Machine Learning*, 2000.