Learning to Identify Facial Expression During Detection Using Markov Decision Process

Ramana Isukapalli
Lucent Technologies, Bell Labs Innovations
Whippany, NJ0 7981, USA
risukapalli@lucent.com

Ahmed Elgammal
Rutgers University
New Brunswick, NJ 08854, USA
elgammal@cs.rutgers.edu

Russell Greiner
University of Alberta, Edmonton, CA T6G 2E8
greiner@cs.ualberta.ca

Abstract

While there has been a great deal of research in face detection and recognition, there has been very limited work on identifying the expression on a face. Many current face detection methods use a Viola–Jones style "cascade" of Adaboost-based classifiers to detect faces. We demonstrate that faces with similar expression form "clusters" in a "classifier space" defined by the real-valued outcomes of these classifiers on the images and address the the task of using these classifiers to classify a new image into the appropriate cluster (expression). We formulate this as a Markov Decision Process and use dynamic programming to find an optimal policy — here a decision tree whose internal nodes each correspond to some classifier, whose arcs correspond to ranges of classifier values, and whose leaf nodes each correspond to a specific facial expression, augmented with a sequence of additional classifiers. We present empirical results that demonstrate that our system accurately determines the expression on a face during detection.

1 Introduction

The pioneering work of Viola and Jones [12] has led to a host of face detectors based on "cascade classifiers", where each classifier is learned by applying Adaboost [5] to a database of training images of faces and non-faces. The underlying principle in these algorithms is to learn multiple classifiers during the training phase, then (at performance time), run these classifiers as a "cascade" — *i.e.*, in a sequence one after another, on each region (at various resolutions) of the test image eliminating non-faces at each stage.

The learning algorithm takes as input several thousands of images correctly labeled as faces and non-faces (each of size 24×24 pixels) and produces a cascade of "boosted classifiers". Each classifier consists of several "linear separators", each built using features on a rectangular subregion of the training image. The learning algorithm selects these linear separators from over a hundred thousand possible candidate features, including the region between the two eyes and the region spanning the upper cheeks and the eyes. Each of these binary classifiers is actually a thresholded real value (see Definition 1), that we refer to as *SCO*-value.

Using these intermediate results of the N classifiers as features, we map each input image to a point in the N–dimensional "classifier space", where each dimension corresponds to the *SCO*-value of one classifier. Our empirical results confirm that faces with the same expression often form clusters in this space. We exploit the classifier space to build a *dynamic tree classifier*, DTC, that partitions the training images of known facial expression into different clusters, each corresponding to a single expression. This DTC is a fixed-depth *decision tree*, whose internal nodes each correspond to a classifier (feature) and whose arcs each correspond to a range of *SCO*-values of the classifiers. We associate a facial expression with each leaf node; see Figure 1.

Note that a *subset* of the N classifiers may be sufficient to distinguish facial expression#1 from facial expression#2; here, it would clearly be inefficient to consider all N classifiers. Unfortunately, a different subset may be necessary to separate expression#1 from expression#3, and a third subset for expression#2 vs expression#3, and so forth. This is why we did not want to use a single set of classifiers throughout, but instead use a dynamic process, that sequentially decides which classifier to use next when dealing with an input image, based on the values observed from the classifiers previously executed on this instance. The challenge is to learn the dynamic sequence of classifiers that can effective distinguish the clusters corresponding to different facial expressions. We formulate this task as a "Markov Decision Process" (MDP), and then use dynamic programming to learn the best *policy*, corresponding to the DTC decision tree.

Our learning system uses two sets of training data — (1) "face labeled training set", FLT, whose images are labeled as either faces (without expression labels) or non-faces; and (2) "face expression labeled training set", FELT, whose faces are labeled with expression. We build a cascade $\mathcal{C} = \langle C_1, C_2 \ldots C_N \rangle$ of N boosted classifiers using FLT. We use these as features when building DTC, which is trained using FELT. At performance time, the resulting system will scan through an unlabeled image. It applies DTC on each sub-image W; this will follow a path of (at most) d classifiers. If it reaches a leaf, DTC uses the *SCO*-value of these classifiers to find the best matching cluster, *i.e.*, facial expression. It then applies the remaining $(N - d)$ classifiers as a cascade to determine whether W is indeed a face. If any of the N classifiers label W as a non-face, the performance system stops processing W and proceeds to the next sub-image. Since we use the entire cascade of $N = d + (N - d)$ classifiers to detect faces, our detection algorithm is essentially Viola-Jones algorithm. However, rather than using a single static cascade, our performance algorithm applies the first d classifiers to each sub-image W dynamically, selecting the i^{th} classifier based on the results of the previous $(i - 1)$ classifiers. It uses the results to identify W with the best matching cluster, to identify an expression for this potential face. It then applies the remaining $(N - d)$ classifiers statically, to determine whether W is actually a face or not.

Section 2 formulates the problem as a MDP. Section 3 describes how we produce this DTC system from a collection of labeled images. In particular, this section shows how we use dynamic programming to sidestep the combinatorial issues — *e.g.*, the exponential number of possible decision trees. This section also explains the details of how we use DTC to detect faces and identify the associated expressions. Section 4 presents empirical results that illustrate the effectiveness of this approach. Section 5 presents relevant work related to our research.

2 Framework

2.1 Markov Decision Process

A *Markov Decision Process* (MDP) is a 4-tuple $\langle \mathcal{S}, A, M, R \rangle$ where $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ is a finite set of states, $A = \{a_1, a_2, \ldots, a_m\}$ is a finite set of actions, $M : \mathcal{S} \times A \times S \to [0, 1]$ is the transition model ($M^a_{s,s'} = P(\, s' \,|\, s, a \,)$ is the probability that taking action a in state $s \in \mathcal{S}$ leads to being in state s') and $R : \mathcal{S} \times A \to \Re$ is the reward an agent gets for taking an action $a \in A$, in s. Notice this is *Markovian* as the transition from state s_i to s_j using action a depends only on s_i and not the previous history. A *policy* $\pi : \mathcal{S} \to A$ is a mapping from states to actions. For any policy π, we can define a utility function U_π such that

$$U_\pi(s) \quad = \quad \max_a \left\{ R(s, a) \; + \; \sum_{s'} M^a_{s,s'} \times U_\pi(s') \right\}$$

corresponds to the expected cumulative rewards of executing the action a in state s, then following policy π after that. Given an MDP, we naturally seek an optimal policy π^* — *i.e.*, a policy that produces the *optimal* cumulative reward, $U^*(s)$, for each state s.

Dynamic programming provides a way to compute this optimal policy, by computing the utilities of the best actions. See Sutton and Barto [11] for details of MDPs and techniques for solving them.

2.2 Identifying facial expression as MDP

In this section we describe how we formulate the problem of identifying facial expression as a MDP.

States: We let "state representation" denote the results of evaluating a set of classifiers on an image region. When processing a test image, DTC will apply some sequence $\langle C_1, C_2, \ldots, C_{i-1} \rangle$ to a sub-image W, then use the results to select the next classifier C_i ($i \leq d$) to apply. Of course, we would like to apply the classifier sequence that has proved to be the most successful in identifying the facial expression of W in similar situations during training. To do this, at training time, our learner will explore every possible sequence of d classifiers on training images, to determine which depth-d tree of classifiers yields the most appropriate clusters. We assign utilities to the leaves of each of these trees — giving a high score if the leaf contains face images with the same expression (see Section 3.1). We then propagate these utility values up the tree. We capture this information in several states and use them to build the DTC.

We identify each node in DTC with a state s. With each state s we associate a "best classifier" to apply, so that it may lead to a cluster with one expression. We can identify each state s with both the sequence of classifiers $\langle C_1, \ldots, C_k \rangle$ on the path from the root $\langle\rangle$ (where no classifier is applied) to s, and a posterior probability distribution over the facial expressions. That is,

$$s = \langle [V_{min,1},\, V_{max,1}],\, \ldots [V_{min,k},\, V_{max,k}], \vec{P} \rangle$$

where $[V_{min,i},\, V_{max,i}]$ is the range of *SCO*-values of classifier C_i already applied and $\vec{P} = \langle P_e \rangle$ such that $P_e = P(\, expr(s) = e \,|\, \vec{E}\,)$ is the probability of that s's expression is e given the evidence $\vec{E}$ accumulated so far — which

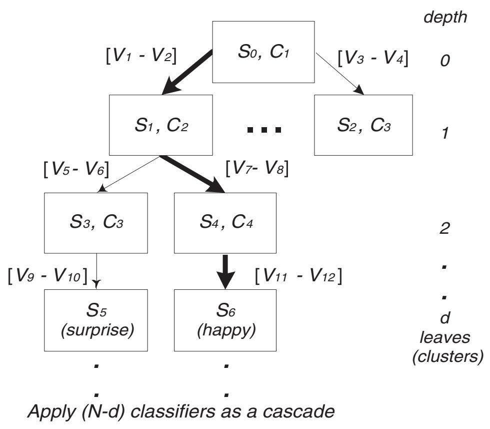

Figure 1. Each node in this DTC is identified with a state. Associated with each state is a best classifier to apply.

is SCO-values of the classifiers applied. Figure 1 shows a simple DTC. Each node in the tree is identified by a state. The *SCO*-value of the classifier determines which branch to take. Nodes at depth d are leaves, which correspond to clusters (which each represent a single facial expression). *E.g.*, in the figure, state s_8 is a cluster of happy faces. It can be identified by the sequence of classifiers $\langle C_1, C_2, C_4 \ldots \rangle$, each with its own range of SCO-values.

We define an equivalency property to compare two states. We say two states are "δ-equivalent", written $s_1 \approx_\delta s_2$, iff

- s_1 and s_2 have applied the same set of classifiers, not necessarily in the same order
- For every classifier C_i used in s_1 and s_2, $|V^{(1)}_{min,i} - V^{(2)}_{min,i}| \leq \delta$ and $|V^{(1)}_{max,i} - V^{(2)}_{max,i}| \leq \delta$, where δ is a pre-defined constant. We set $\delta = 70$ throughout this work.

Actions: Actions are the set of classifiers that can be applied to a state — *i.e.*, $A = \{C_1, C_2 \ldots C_N\}$.

Reward: We assign a high reward to states that group objects of the same expression together. We use the reward function

$$R(s) = \begin{cases} max_i \{P(\, expr(s) = i\,)\} & \text{if } depth = d \\ -\alpha \times FN(s) & \text{otherwise} \end{cases} \tag{1}$$

If the node's depth is d, this assigns the probability of the most likely expression (of the images that reach here); otherwise it penalizes by $\alpha \times FN(s)$ where α is predefined constant (set to 0.01) and $FN(s)$ is the number false negatives in state s.

Transition Model: The transition model $M^a_{s,s'} = P(\, s' \,|\, s, a\,)$ is the probability that taking action a in state s leads to s' — *i.e.*, applying a classifier a to an image from s will produce an image in s'. That is, when attempting to classify a test image I_t, let W be the current sub-image and $\langle a_1, a_2 \ldots a_{i-1} \rangle$ be the classifiers already applied on W. Assume all of these classifiers classify W as a positive instance (face) and let s be the $delta$-equivalent state (defined above) in DTC that is the closest to matching the outcomes (*SCO*-value) of these classifiers. Further, let a_i be the classifier that is applied next (as decided by DTC) on W and o_i be the *SCO*-value of a_i on W. Since we are interested in finding the expression of W, we set $M^a_{s,s'} = P(\, expr(s') = e \,|\, a = o_i\,)$. Since o_i can have a large range of values, we discretize it into "buckets" $B_r \subset \Re$, *i.e.*, $B_0 = [-500, -400)$, $B_1 = [-400, -300)$ and so on. Using Bayes rule,

$$\begin{aligned} &P(\, expr(s') = e \,|\, a_i = o_i\,) \\ &\quad = \frac{P(\, a_i{=}o_i \,|\, expr(s'){=}e\,) \times P(expr(s'){=}e)}{P(a_i{=}o_i)} \\ &\quad = \frac{P(\, o_i{\in}B_i \,|\, expr(s'){=}e\,) \times P(expr(s'){=}e)}{P(o_i{\in}B_i)} \end{aligned} \tag{2}$$

We estimate the terms on the right hand side of Equation 2 from training data. We apply every classifier on training images of each expression e in FELT, partition the SCO-values into different buckets (ranges), and compute $P(\, o_i \in B_i \,|\, expr(s') = e\,)$. We set $P(\, expr(s') = e\,) = \frac{1}{M}$ where M is the total number of expressions, which here is 5.

Further, to update $\vec{P}$ using the outcomes of two actions, a_1 and a_2 we use the Naive-Bayes assumption [4]

$$\begin{aligned} &P(\, expr(s') = e \,|\, a_1 = o_1, a_2 = o_2\,) \\ &\quad = \frac{P(\, a_1{=}o_1, a_2{=}o_2 \,|\, expr(s'{=}e)\,) \times P(expr(s'){=}e)}{P(a_1{=}o_1, a_2{=}o_2)} \\ &\quad = \frac{P(\, a_1{=}o_1 \,|\, expr(s'{=}e)\,) \times P(\, a_2{=}o_2 \,|\, expr(s'{=}e)\,) \times P(expr(s'){=}e)}{P(a_1{=}o_1, a_2{=}o_2)} \\ &\quad = \frac{P(\, expr(s'{=}e) \,|\, a_1{=}o_1\,) P(\, expr(s'{=}e) \,|\, a_2{=}o_2\,)}{P(expr(s'){=}e)} \times \\ &\qquad \frac{P(a_1{=}o_1) \times P(a_2{=}o_2)}{P(a_1{=}o_1, a_2{=}o_2)} \end{aligned} \tag{3}$$

Taking $P(expr(s')) = e = \frac{1}{M}$ (a constant) and $P(a_1 = o_1, a_2 = o_2) = P(a_1 = o_1) \times P(a_2 = o_2)$ we get

$$\begin{aligned} &P(\, expr(s') = e \,|\, a_1 = o_1, a_2 = o_2\,) \\ &\quad = \alpha \times P(\, expr(s' = e) \,|\, a_1 = o_1\,) \times \\ &\qquad P(\, expr(s' = e) \,|\, a_2 = o_2\,) \end{aligned} \tag{4}$$

where α is a normalizing constant.

3 Identifying Facial Expression

In this section we describe the details of constructing a DTC using dynamic programming. We also explain how we use DTC to identify facial expression during detection.

3.1 Use of dynamic programming to build DTC

Figure 2(a) presents the learning algorithm. It has two goals: first, to partition the images in the training set into

```
Learn_DTC(FLT, FELT: TrainingSets)
  • Build a cascade of Adaboost classifiers
    $\langle C_1, C_2, \ldots C_N\rangle$ using images in FLT.
  • Let $All$ be the set of all $\binom{N}{d}$ sequences of $d$ of classifiers
  • Build decision tree based on $C_{1:N}$. During tree expansion,
    at depth $i$ after applying any $C_i$ from each of $All$ sequences
    – Remove each image $C_i$ classifies as a non-face
    – Partition remaining images into two equal halves
      based on their $C_i$-based SCO-value
    – Apply each $C_{i+1}$ to each half and continue
  • Compute utility at each leaf (i.e., cluster), $U(s^{(d)})$
  • Propagate utility up tree
    – For state $s_i$ at depth $i < d$, $U(s^{(i)}) = max_j \left\{U(s_j^{(i+1)})\right\}$
    – Let $C_i^*$ yield maximum utility when applied on $s_i$
    – Associate $C_i^*$ with $s_i$, store $\langle s_i, C_i^*\rangle$
  • Merge all $\delta$-equivalent $s_i$ states ($i \leq d$) into one,
    store one classifier $C_i^*$ with the maximum utility
```

```
Use_DTC($I_t$ : Test Image)
  • Set ratio = 1.0
  ⋆ For each window $W$ (of 24 × 24 pixels) within $I_t$
    ○ For $i \leq d$
      – Find state $s_i$ "closest" to $W$
      – Apply $C_i^*$ associated with $s_i$
      – Update posterior distribution $\vec{P}$ over expressions
    ○ If $\langle C_1, C_2, \ldots C_d\rangle$ label the window as a face
      – Find the corresponding cluster, i.e., $s_d$
      – Note the most likely expression $e$ of $s_d$
      – Apply the other classifiers $\langle C_{d+1}, C_{d+2}, \ldots C_N\rangle$
      – If they label $W$ as a face, mark $W$ as a face
        with expression $e$
  • Set ratio := ratio × 0.8, resize $I_t$ by a factor of ratio.
  • If $I_t$.length ≥ 24 and $I_t$.width ≥ 24, goto ⋆
  • Return all marked faces with their expressions
```

Figure 2. (a) Learning algorithm to discover clusters and build DTC; (b) Dynamic classification algorithm

meaningful clusters of images with the same expression, and second, to find the most effective sequence of d classifiers for each cluster. First we define *SCO*-value.

Definition 1 *SCO-value: Let C_i be any boosted classifier with T linear separators. Viola Jones algorithm classifies any sub-image W as a face if $\sum_{i=1}^{T} \alpha_i \cdot h_i(W) \geq \frac{1}{2}\sum_{i=1}^{T} \alpha_i$ where α_i is the weight given to i^{th} linear separator h_i and $h_i(W)$ is the classification result of h_i on W as a face or non-face (see [12] for details). We refer to the quantity $\sum_{i=1}^{T} \alpha_i \cdot h_i(W)$, i.e., the weighted sum of the individual linear separators outcome as the* SCO*-value ("sum of classifier output values") of C_i on W.*

Exploring sequences of classifiers: We build a cascade of N boosted classifiers using the images of FLT. We then produce a depth-d decision tree (DTC) by exploring all possible sequences of d classifiers, from the N classifiers. If N is large (typically > 15), we choose the first $N' < N$ classifiers and build the DTC from N' classifiers. We use a dynamic programming tableau to learn an optimal depth-d decision tree. For any classifier C_i that we apply on the image set FELT, we compute the *SCO*-value of C_i on FELT and sort them based on their *SCO*-values. We reject each images that any C_i labels as negative. We partition the rest into two equal halves, $\langle T_i^L\rangle$ and $\langle T_i^R\rangle$ (to denote the left and right branches after partitioning). We then apply the next classifier C_{i+1} on $\langle C_i^L\rangle$ and $\langle C_i^R\rangle$ separately at depth $(i+1)$ and repeat the procedure until a total of d classifiers are applied. The leaves at depth d represent clusters.

Applying a classifier is equivalent to finding its projection on the i^{th} dimension in the classifier space. If we take such projections and partition the images repeatedly along d dimensions corresponding to d classifiers, then we can retrieve the clusters.

The single depth-0 node $\langle\rangle$ contains all of the images considered, FELT. To compute the images in the $\langle C_1^L\rangle$ node: First let T_1 be the subset of FELT that pass the C_1 classifier; assume these are sorted based on their *SCO*-value as $\langle w_1, \ldots, w_{m/2}, w_{m/2+1}, \ldots, w_m\rangle$, where $V_1(w_j) \geq V_1(w_k)$ when $j > k$. Then the $\langle T_1^L\rangle$ node contains $\{w_1, \ldots, w_{m/2}\}$, and $\langle T_1^R\rangle$ contains $\{w_{m/2+1}, \ldots, w_m\}$. We can then compute $\langle T_1^L, T_2^L\rangle$ and $\langle T_1^L, T_2^R\rangle$ that each contains one half of the images of $\langle T_1^L\rangle$ that classifier C_2 labels as faces. We continue for d levels, producing 2^d nodes at the leaves, each of which represents one cluster. When we consider $\binom{N}{d}$ sequences of classifiers, we get $\binom{N}{d} \times 2^d$ clusters. There may be over-segmentation but any test image can still be assigned the correct expression by being matched to one of the over-segmented clusters.

Computing $U(\,s\,)$: We want to determine the best decision tree within this tableau — the one that leads to the "purest" leaf nodes. Each leaf of the tree represents a possible cluster. We want the clusters that are as "pure" as possible, *i.e.*, which group images of only one expression together. We compute the various utility values using a dynamic programming approach. We first set the utility of the $s^{(d)}$ leaf states (clusters), then we propagate the utility up the tree using dynamic programming. We set the utility of any internal node as the maximum utility of its children,

$$\begin{aligned} U(s^{(d)}) &= \max_e\{P(\,expr(s^{(d)}) = e \,|\, \vec{E}\,)\} - \\ &\quad \textstyle\sum_{i=1}^{d} \alpha \cdot FN(C_i) \\ U(s^{(i)}) &= \max_j\{U(s_j^{(i+1)})\} \end{aligned} \tag{5}$$

All the terms used here have the same notation as in Section 2.2. The idea is to assign high utility value to clusters that group images of the same expression together and penalize them for having a lot of false negatives.

Building DTC**:** We collect the $\langle s_i, C_i^* \rangle$ tuples and also the corresponding utilities, for all depths $i < d$. C_i^* denotes the classifier that, when applied to s_i, transitions it to another state s_{i+1}^*, with the maximum utility among the children of s_i. For every two states s_i and s_j $(i \neq j)$ that are δ-equivalent we retain only one state that has a higher utility and the corresponding classifier. Note that the $\langle s_i, C_i^* \rangle$ tuples for $i \leq d$ tell us precisely the i classifiers applied so far, their individual *SCO*-values and the best classifier C_i^* to apply in s_i. This produces the decision tree, DTC.

3.2 Detection

Our detection algorithm, shown in Figure 2(b), uses classifiers built by the cascade classifiers method [12, 13]. It examines each 24×24 pixel window in the image; it then rescales length and width of the test image by a factor of 0.8 and repeats. For each window W, DTC first applies the classifier C_1^* associated with root. This might reject the window W; if so DTC continues with the next window. Otherwise, DTC computes the *SCO*-value associated with C_1^* on W, updates the probability distribution of facial expressions (using Equations 2 and 4) and uses this value to decide which subsequent classifier C_2^* to apply. Again this could reject W, but if not, C_2^*'s *SCO*-value identifies the next classifier C_3^* to apply on W. This can continue for d steps, until W reaches a leaf. If all the d classifiers label W as a positive instance, DTC finds the most likely expression e associated with the cluster. We then run the remaining $(N - d)$ classifiers $\langle C_{d+1}, \ldots C_N \rangle$ and declare W to be a face with expression e if all these classifiers label it as a positive instance. Otherwise, we reject W as a non-face.

4 Experimental Results

Our training set FLT has 1600 images of faces and 2320 images of non-faces[1]. The FELT set has a total of 551 face images of the five basic expressions (sad, fear, surprise, no-expression and happy). It also has 2320 non-face images. The size of each training image is 24×24 pixels.

During the training stage, we built a cascade of 21 classifiers using Adaboost. Using the first 10 classifiers, we explore all sequences of d classifiers, as a tree of depth $d = 3$, as explained in Section 3.1. Figure 3 shows face images of the same expression from some interesting clusters that our algorithm learned.

To detect faces in any given test image, we use the dynamic detection technique explained in Section 3.2. Figure 4 shows the performance of our detection algorithm on a number of face images with various expressions.

[1]All the images were downloaded from popular databases like Olivetti Research & AT&T, Caltech, Yale, JAFFE, PICS, etc.

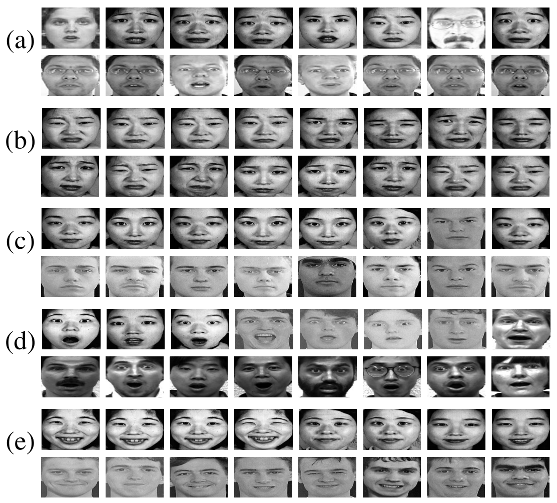

Figure 3. Various clusters discovered from the FELT data — (a) Fear (b) Sad (c) No Expression (d) Surprised (e) Happy

Accuracy: We ran our performance algorithm over 150 randomly selected images from Olivetti Research database face images. We manually assigned a facial expression to each image and compared it to the facial expression assigned by our performance algorithm. Our performance algorithm could identify the expression correctly in 92 images, *i.e.*, its accuracy was 61.33% in assigning expression. Note that facial expressions can be mixed — sad and fear, happy and surprised, etc. To account for this, we also noted the number of images for which the most likely *or* the second most likely expression assigned by our algorithm matched with the facial expression we assigned manually. In this case, our algorithm could performed correctly for 105 images, *i.e.*, its accuracy improved to 70.0%. To find out the effectiveness of our face detection algorithm, we ran it on 178 image face images of the MIT-CMU database with a total of over 532 faces and covering more than 76 million windows. Figure 5 gives the ROC curve on these faces.

Efficiency: Our test set has 150 images each of size 92×112 pixels. Our detection algorithm averaged (per image) 45 msec. to detect faces *and* assign an expression to each face. By comparison, Viola-Jones algorithm took 36.3 msec. We attribute the increase in computational time to two factors: (1) our overhead in assigning an expression and (2) the standard Viola-Jones algorithm is optimized for speed, as it applies classifiers with a small number of linear separators first. We need to choose the first d classifiers dynamically using DTC. Our algorithm may choose to apply complex classifiers first (to assign an expression), which can increase the run time. We still see that our algorithm has

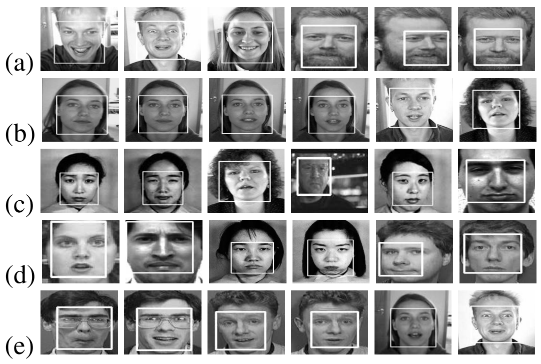

Figure 4. Performance on several test images — (a) happy (b) no expression (c) sad (d) fear (e) surprise.

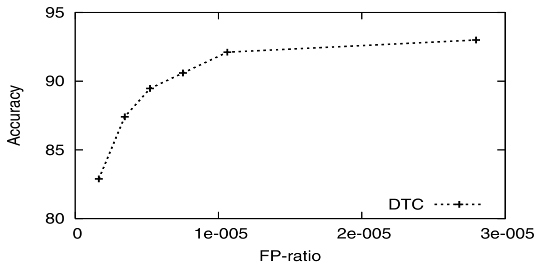

Figure 5. ROC curve

speed comparable to Viola-Jones algorithm.

5 Previous Work

As noted above, our work is closely related to Viola-Jones algorithm [12]; we extend it by identifying the facial expression of each detected face. There has been other results in this area. Liu and others [7, 3, 2] track facial features and analyze them for facial expressions. Almost all of the other methods [1] perform a local analysis of facial features, like mouth, eyes, etc. Our work is different from all these as we do not use sequence of images or analyze facial features explicitly, but instead use a training set to group face images of expression into clusters. We associate each detected face with a cluster to identify the expression.

We addressed related issues in a feature-based face-recognition system by posing the task as MDP [6]. We used dynamic programming to produce an optimal policy, π^*, that maps "states" to "actions" (feature detectors) for that MDP, then used that optimal policy to recognize faces *efficiently*. We use similar techniques here to assign a facial expression to each face during detection. Many researchers have recently proposed several methods for detecting faces in images; see [8, 10, 9] for a small sample.

6 Conclusions

Our main contribution is a system that assigns expressions to faces, during detection. We demonstrate empirically that face images of the same expression form clusters in the classifier space. We formulate the problem of finding an optimal sequence of classifiers to retrieve clusters as a MDP and use dynamic programming to solve it. Our training algorithm partitions training images into clusters of similar expressions. During detection, every detected face is matched to a cluster to identify the expression.

References

[1] B. Abboud and F. Davoine. Facial expression recognition and synthesis based on appearance model, *Signal Processing and Image Communication, Elsevier*, Vol. 19, No. 8, Sep. 2004

[2] J.F. Cohn, T. Kanade, T.K. Wu, Y.T. Lien and A.Zlochower. Facial Analysis: Preliminary analysis of a new image processing based method, *International Society for Research in Motion*, Toronto, 1996

[3] J.F. Cohn, A. Zlochower, J. Lien, Y.T. Wu, T. Kanade. Automated face coding: A computer vision based method of facial expression analysis, *Seventh European Conference on Facial Expression, Measurement and Meaning*, Salzburg, 1997.

[4] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.

[5] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting, *Computational Learning Theory: Eurocolt*, 1995.

[6] R. Isukapalli, and R. Greiner. Use of Off-line Dynamic Programming for Efficient Image Interpretation, *IJCAI*, 2003.

[7] Y. Liu, K. Schmidt, J.F. Cohn and S. Mitra. Facial asymmetry quantification for expression invariant human identification, *Computer Vision and Image Understanding*, vol 91, 2003.

[8] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection, *IEEE Transactions on PAMI*, 1998.

[9] D. Roth, M. Yang, and N. Ahuja. A snowbased face detector, In *NIPS*, 2000.

[10] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars, *ICCV*, 2000.

[11] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, 1998.

[12] P. Viola and M. Jones. Robust real-time face detection, *IJCV*, 57(2), 2004.

[13] J. Wu, J.M. Rehg, and M.D. Mullin. Learning a rare event detection cascade by direct feature selection, *NIPS*, 2003.