Journal of Machine Learning Research 7 (2006) 2087-2123 Submitted 2/05; Revised 6/06; Published 10/06

A Hierarchy of Support Vector Machines for Pattern Detection

Hichem Sahbi HS385@CAM.AC.UK
Machine Intelligence Laboratory
Department of Engineering
University of Cambridge
Trumpington Street
Cambridge, CB2 1PZ, UK

Donald Geman GEMAN@JHU.EDU
Center for Imaging Science
302 Clark Hall
Johns Hopkins University
3400 N. Charles Street
Baltimore, MD 21218, USA

Editor: Pietro Perona

Abstract

We introduce a computational design for pattern detection based on a tree-structured network of support vector machines (SVMs). An SVM is associated with each cell in a recursive partitioning of the space of patterns (hypotheses) into increasingly finer subsets. The hierarchy is traversed coarse-to-fine and each chain of positive responses from the root to a leaf constitutes a detection. Our objective is to design and build a network which balances overall error and computation.

Initially, SVMs are constructed for each cell with no constraints. This "free network" is then perturbed, cell by cell, into another network, which is "graded" in two ways: first, the number of support vectors of each SVM is reduced (by clustering) in order to adjust to a pre-determined, increasing function of cell depth; second, the decision boundaries are shifted to preserve all positive responses from the original set of training data. The limits on the numbers of clusters (virtual support vectors) result from minimizing the mean computational cost of collecting all detections subject to a bound on the expected number of false positives.

When applied to detecting faces in cluttered scenes, the patterns correspond to poses and the free network is already faster and more accurate than applying a single pose-specific SVM many times. The graded network promotes very rapid processing of background regions while maintaining the discriminatory power of the free network.

Keywords: statistical learning, hierarchy of classifiers, coarse-to-fine computation, support vector machines, face detection

1. Introduction

Our objective is to design and build a "pattern detection" system based on a tree-structured network of increasingly complex support vector machines (SVMs) (Boser et al., 1992; Osuna et al., 1997). The methodology is general, and could be applied to any classification task in machine learning in which there are natural groupings among the patterns (classes, hypotheses). The application which motivates this work is to detect and localize all occurrences in a scene of some particular object category based on a single, grey-level image. The particular example of detecting faces against

cluttered backgrounds provides a running illustration of the ideas where the groupings are based on pose continuity.

Our optimization framework is motivated by natural trade-offs among invariance, selectivity (background rejection rate) and the cost of processing the data in order to determine all detected patterns. In particular, it is motivated by the amount of computation involved when a single SVM, dedicated to a reference pattern (e.g., faces with a nearly fixed position, scale and tilt), is applied to many data transformations (e.g., translations, scalings and rotations). This is illustrated for face detection in Fig 1; a graded network of SVMs achieves approximately the same accuracy as a pattern-specific SVM but with order 100 to 1000 times fewer kernel evaluations, resulting from the network architecture as well as the reduced number of support vectors.

To design and construct such a graded network, we begin with a hierarchical representation of the space of patterns (e.g., poses of a face) in the form of a sequence of nested partitions, one for each level in a binary tree (Fleuret and Geman, 2001; Fleuret, 1999; Sahbi et al., 2002; Jung, 2001; Blanchard and Geman, 2005; Amit et al., 2004; Gangaputra and Geman, 2006a). Each cell - distinguished subset of patterns - encodes a simpler, sub-classification task and is assigned a binary classifier. The leaf cells represent the resolution at which we desire to "detect" the true pattern(s). There is also a "background class," for example, a complex and heterogeneous set of non-distinguished patterns, which is statistically dominant (i.e., usually true). A pattern is "detected" if the classifier for every cell which covers it responds positively.

Initially, SVMs are constructed for each cell in the standard way (Boser et al., 1992) based on a kernel and training data – positive examples (from a given cell) and negative examples ("background"). This is the "free network," or "f-network" $\{f_t\}$, where t denotes a node in the tree hierarchy. The "graded network," or "g-network" $\{g_t\}$, is indexed by the same hierarchy, but the number of intervening terms in each g_t is fixed in advance (by clustering those in f_t as in Schölkopf et al., 1998), and grows with the level of t. (From here on, the vectors appearing g_t will be referred to as "support vectors" even though, technically, they are constructed from the actual support vectors appearing in f_t.) Moreover, the decision boundaries are shifted to preserve all positive responses from the original set of training data; consequently, the false negative (missed detection) rate of g_t is at most that of f_t and any pattern detected by the f-network is also detected by the g-network. But the g-network will be far more efficient.

The limits on the numbers of support vectors result from solving a constrained optimization problem. *We minimize the mean computation necessary to collect all detections subject to a constraint on the rate of false detections.* (In the application to face detection, a false detection refers to finding a face amidst clutter.) Mean computation is driven by the background distribution. This also involves a model for how the selectivity of an SVM depends on complexity, which is assumed proportional to the number of support vectors, and invariance, referring to the "scope" of the underlying cell in the hierarchy.

In the free network, the complexity of each SVM decision function depends in the usual way on the underlying probability distribution of the training data. For instance, the decision function for a linearly separable training set might be expressed with only two support vectors, whereas the SVMs induced from complex tasks in object recognition usually involve many support vectors (Osuna et al., 1997). For the f-network, the complexity generally *decreases* as a function of depth due to the progressive simplification of the underlying tasks. This is illustrated in Fig 2 (left) for face detection; the classifiers f_t were each trained on 8000 positive examples and $50,000$ negative examples. Put differently, complexity increases with invariance.

Figure 1: Comparison between a single SVM (top row) dedicated to a nearly fixed pose and our designed network (bottom row) which investigates many poses simultaneously. The sizes of the three images are, left to right, 520×739, 462×294 and 662×874 pixels. The network achieves approximately the same accuracy as the pose-specific SVM but with order 100-1000 times fewer kernel evaluations. Some statistics comparing efficiency are given in Table 1.

Consider an SVM f in the f-network with N support vectors and dedicated to a particular hypothesis cell; this network is slow, but has high selectivity and few false negatives. The corresponding SVM g has a specified number n of support vectors with $n \leq N$. It is intuitively apparent that g is less selective; this is the price for maintaining the false negative rate and reducing the number of kernel evaluations. In particular, if n is very small, g will have low selectivity (cf. Fig 2 (right)). In general, of course, with no constraints, the fraction of support vectors provides a rough measure of the difficulty of the problem; here, however, we are *artificially* reducing the number of support vectors, thereby limiting the selectivity of the classifiers in the g-network.

Building *expensive* classifiers at the upper levels ($n \approx N$) leads to intensive early processing, even when classifying *simple* background patterns (e.g., flat areas in images), so the overall mean

Figures	"Mona Lisa"			"Singers"			"Star Trek"		
	1 SVM	**f-net**	**g-net**	**1 SVM**	**f-net**	**g-net**	**1 SVM**	**f-net**	**g-net**
# **Subimages Processed**	$2 \,.\, 10^5$	$2 \,.\, 10^3$	$2 \,.\, 10^3$	$5 \,.\, 10^4$	$8 \,.\, 10^2$	$8 \,.\, 10^2$	$2 \,.\, 10^5$	$4 \,.\, 10^3$	$4 \,.\, 10^3$
# **Kernel Evaluations**	$5 \,.\, 10^7$	10^7	$3 \,.\, 10^4$	$2 \,.\, 10^7$	$7 \,.\, 10^6$	10^4	$8 \,.\, 10^7$	$2 \,.\, 10^7$	$5 \,.\, 10^4$
Processing Time (s)	172.45	28.82	0.53	55.87	17.83	0.26	270.1	48.92	0.87
# **Raw Detections**	3	3	4	12	14	15	19	20	20

Table 1: Comparisons among i) a single SVM dedicated to a small set of hypotheses (in this case a constrained pose domain), ii) the f-network and iii) our designed g-network, for the images in Fig 1. For the single SVM, the position of the face is restricted to a 2×2 window, its scale to the range $[10, 12]$ pixels and its orientation to $[-5^0, +5^0]$; the original image is downscaled 14 times by a factor of 0.83 and for each scale the SVM is applied to the image data around each non-overlapping 2×2 block. In the case of the f and g-networks, we use the coarse-to-fine hierarchy and the search strategy presented here.

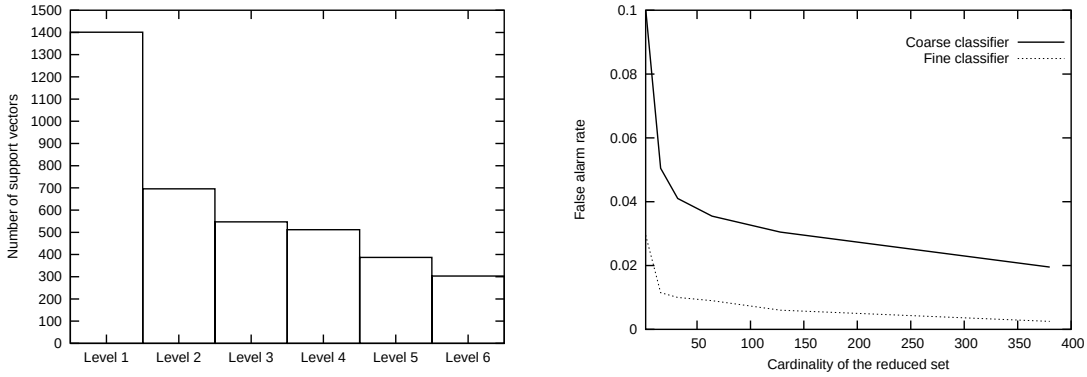


Figure 2: Left: The average number of support vectors for each level in an f-network built for face detection. The number of support vectors is decreasing due to progressive simplification of the original problem. Right: False alarm rate as a function of the number of support vectors using two SVM classifiers in the g-network with different pose constraints.

cost is also very large (cf. Fig 3, top rows). As an alternative to building the g-network, suppose we simply replace the SVMs in the upper levels of f-network with *very simple* classifiers (e.g., linear SVMs); then many background patterns will reach the lower levels, resulting in an overall loss of efficiency (cf. Fig 3, middle rows).

We focus in between these extremes and build $\{g_t\}$ to achieve a certain trade-off between cost and selectivity (cf. Fig 3, bottom rows). Of course, we cannot explore all possible designs so a model-based approach is necessary: The false alarm rate of each SVM is assumed to vary with complexity and invariance in a certain way. This functional dependence is consistent with the one proposed in Blanchard and Geman (2005), where the computational cost of a classifier is modeled as the product of an increasing function of scope and an increasing function of selectivity.

Finally, from the perspective of computer vision, especially image interpretation, the interest of this paper is the proposed architecture for aggregating binary classfiers such as SVMs for organized

Maximum Level Reached	1	2	3	4	5	6
# Samples (f-network)	1697	56	4	1	0	2
# Samples (heuristic)	936	555	135	17	54	63
# Samples (g-network)	1402	336	2	0	3	17

# **Kernel Evaluations**	$2-10$	$10-10^2$	10^2-10^3	10^3-10^4	10^4-10^5	10^5-10^6
# Samples (f-network)	0	0	0	1697	58	5
# Samples (heuristic)	936	755	67	2	0	0
# Samples (g-network)	1402	340	18	0	0	0

Figure 3: In order to illustrate varying trade-offs among cost, selectivity and invariance, and to demonstrate the utility of a principled, global analysis, we classified 1760 subimages of size 64×64 extracted from the image shown above using three different types of SVM hierarchies of depth six. In each case, the hierarchy was traversed coarse-to-fine. For each hierarchy type and each subimage, the upper table shows the distribution of the deepest level visited and the lower table shows the distribution of cost in terms of the total number of kernel evaluations. In both tables: Top row: The unconstrained SVM hierarchy ("f-network") with a Gaussian kernel at all levels; the SVMs near the top are very expensive (about 1400 support vectors at the root; see Fig 2) resulting in high overall cost. Middle row: An ad hoc solution: the same f-network, except with linear SVMs (which can be assumed to have only two support vectors) at the upper three levels in order to reduce computation; many images reach deep levels. Bottom row: The constrained SVM hierarchy ("g-network"), globally designed to balance error and computation; the number of (virtual) support vectors grows with depth.

scene parsing. For some problems, dedicating a single classifier to each hypothesis, or a cascade (linear chain) of classifiers to a small subset of hypotheses (see Section 2), and then training with existing methodology (even off-the-shelf software) might suffice, in fact provide state-of-the-art performance. This seems to the case for example with frontal face detection as long as large training sets are available, at least thousands of faces and sometimes billions of negative examples, for learning long, powerful cascades. However, those approaches are either very costly (see above) or may not scale to more ambitious problems involving limited data, or more complex and varied interpretations, because they rely too heavily on brute-force learning and lack the structure necessary to hardwire efficiency by simultaneously exploring multiple hypotheses.

We believe that hierarchies of classifiers provide such a structure. In the case of SVMs, which may require extensive computation, we demonstrate that building such a hierarchy with a global design which accounts for both cost and error is superior to either a single classifier applied a great many times (a form of template-matching) or a hierarchy of classifiers constructed independently, node-by-node, without regard to overall performance. We suspect that the same demonstration could be carried out with other "base classifiers" as long as there is a natural method for adjusting the amount of computation; in fact, the global optimization framework could be applied to improve other parsing strategies, such as cascades.

The remaining sections are organized as follows: A review of coarse-to-fine object detection, including related work on cascades, is presented in Section 2. In Section 3, we discuss hierarchical representation and search in general terms; decomposing the pose space provides a running example of the ideas and sets the stage for our main application - face detection. The f-network and g-network are defined in Section 4, again in general terms and the statistical framework and optimization problem are laid out in Section 5. This is followed in Section 6 by a new formulation of the "reduced set" method (Burges, 1996; Schölkopf et al., 1998), which is used to construct an SVM of specified complexity. These ideas are illustrated for a pose hierarchy in Section 7, including a specific instance of the model for chain probabilities and the corresponding minimization of cost subject to a constraint on false alarms. Experiments are provided in Section 8, where the g-network is applied to detect faces in standard test data, allowing us to compare our results with other methods. Finally, some conclusions are drawn in Section 9.

2. Coarse-to-Fine Object Detection

Our work is motivated by difficulties encountered in inducing semantic descriptions of natural scenes from image data. This is often computationally intensive due to the large amount of data to be processed with high precision. Object detection is such an example and has been widely investigated in computer vision; see for instance Osuna et al. (1997); Fleuret and Geman (2001); Kanade (1977); Schneiderman and Kanade (2000); Sung (1996); Viola and Jones (2001) for work on face detection. Nonetheless, there is as yet no system which matches human accuracy; moreover, the precision which is achieved often comes at the expense of run-time performance or a reliance on massive training sets.

One approach to computational efficiency is *coarse-to-fine processing*, which has been applied to many problems in computer vision, including object detection (Fleuret and Geman, 2001; Viola and Jones, 2001; Geman et al., 1995; Baker and Nayar, 1996; Amit and Geman, 1999; Rowley, 1999; Heisele et al., 2001), matching (Borgefors, 1988; Huttenlocher and Rucklidge, 1993; Gee and Haynor, 1996), optical flow (Battiti and Koch, 1991), tracking (Sobottka and Pittas, 1996) and

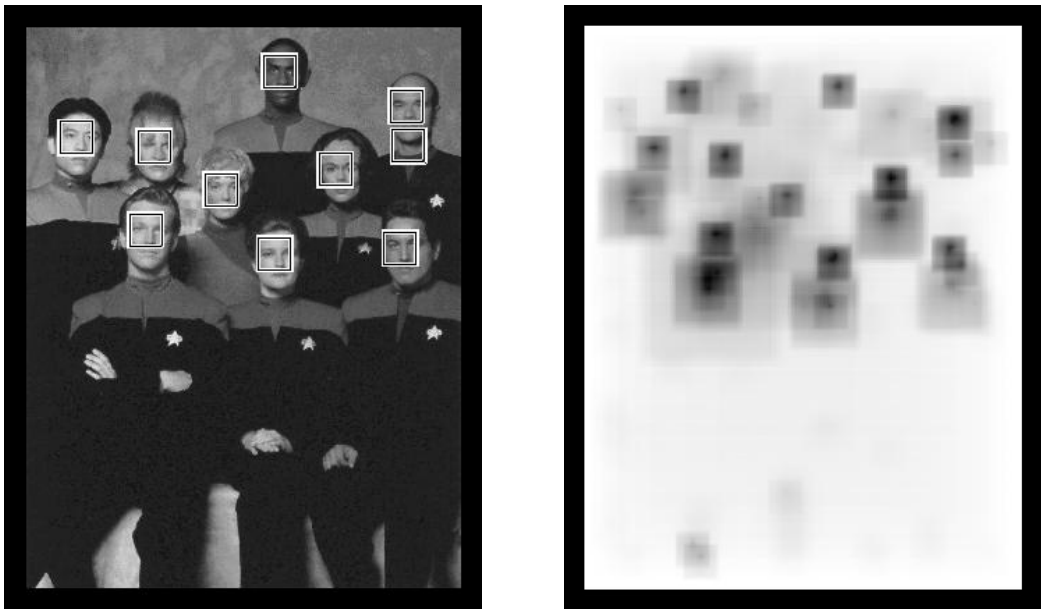


Figure 4: Left: Detections using our system. Right: The darkness of a pixel is proportional to the amount of local processing necessary to collect all detections.

other tasks such as compression, registration, noise reduction and estimating motion and binocular disparity. In the case of object detection, one strategy is to focus rapidly on areas of interest by finding characteristics which are common to many instantiations; in particular, background regions are quickly rejected as candidates for further processing (see Fig 4).

In the context of finding faces in cluttered scenes, Fleuret and Geman (2001) developed a fast, coarse-to-fine detector based on simple edge configurations and a hierarchical decomposition of the space of poses (location, scale and tilt). (Similar, tree-structured recognition strategies appear in Geman et al. (1995); Baker and Nayar (1996).) One constructs a family of classifiers, one for each cell in a recursive partitioning of the pose space and trained on a sub-population of faces meeting the pose constraints. A face is declared with pose in a leaf cell if all the classifiers along the chain from root to leaf respond positively. In general, simple and uniform structures in the scene are quickly rejected as face locations (i.e., very few classifiers are executed before all possible complete chains are eliminated) whereas more complex regions, for instance textured areas and face-like structures, require deeper penetration into the hierarchy. Consequently, the overall cost to process a scene is dramatically lower than looping over many individual poses, a form of template- matching (cf. Fig 1).

Work on cascades (Viola and Jones, 2001; Elad et al., 2002; Eveland et al., 2005; Keren et al., 2001; Socolinsky et al., 2003; Romdhani et al., 2001; Kienzle et al., 2004; Wu et al., 2005) is also motivated by an early rejection principle to exploit skewed priors (i.e., background domination). In that work, as in ours, the time required to classify a pattern (e.g., an input subimage) depends on the resemblance between that pattern and the objects of interest. For example, Viola and Jones (2001) developed an accurate, real-time face detection algorithm in the form of a cascade of boosted classifiers and computationally efficient feature detection. Other variations, such as those in Wu

et al. (2005); Romdhani et al. (2001) for face detection, and the cascade of inner products in Keren et al. (2001) for object identification, employ very simple linear classifiers. In nearly all cases the individual node learning problems are treated heuristically; an exception is Wu et al. (2005), where, for each node, the classifiers are designed to solve a (local) optimization problem constrained by desired (local) error rates.

There are several important differences between our work and cascades. Cascades are coarse-to-fine in the sense of background filtering whereas our approach is coarse-to-fine both in the sense of hierarchical pruning of the background class and representation of the space of hypotheses. In particular, cascades operate in a more or less brute-force fashion because every pose (e.g., position, scale and tilt) must be examined separately. In comparing the two strategies, especially our work with cascades of SVMs for face detection as in Kienzle et al. (2004); Romdhani et al. (2001), there is then a trade-off between very fast early rejection of individual hypotheses (cascades) and somewhat slower rejection of collections of hypotheses (tree-structured pruning).

No systematic comparison with cascades has been attempted. Moving beyond an empirical study would require a model for how cost scales with other factors, such as scope and selectivity. One such model was proposed in Blanchard and Geman (2005), in which the computational cost $C(f)$ of a binary classifier f dedicated to a set A of hypotheses (against a universal "background" alternative) is expressed as

$$C(f) = \Gamma(|A|) \times \Psi(1-\delta)$$

where δ is false positive rate of the classifier f (so $1-\delta$ is what we have called the selectivity) and Γ and Ψ are increasing functions with Γ subadditive and Ψ convex. (Some empirical justification for this model can be found in Blanchard and Geman (2005).) One can then compare the cost of testing a "small" set A of hypotheses (e.g., all poses over a small range of locations, scales and tilts, as in cascades) versus a "large" set $B \supset A$ (e.g., many poses simultaneously, as here). Under this cost model, and equalizing the selectivity, the subadditivity of Γ would render the test dedicated to B cheaper than doing the test dedicated to A approximately $\frac{|A|}{|B|}$ times, even ignoring the inevitable reduction in selectivity due to repeated tests.

More importantly, perhaps, it is not clear that cascades will scale to more ambitious problems involving many classes and instantiations since repeatedly testing a coarse set of hypotheses will lack selectivity and repeatedly testing a narrow one will require a great many implementations.

Finally, to our knowledge, the work presented in this paper is the first to consider a global construction of the system in an optimization framework. In particular, no global criteria appear in either Fleuret and Geman (2001) or Viola and Jones (2001); in the former, the edge-based classifiers are of roughly constant complexity whereas in the latter the complexity of the classifiers along the cascade is not explicitly controlled.

3. Hierarchical Representation and Search

Let Λ denote a set of "patterns" or "hypotheses" of interest. Our objective is to determine which, if any, of the hypotheses $\lambda \in \Lambda$ is true, the alternative being a statistically dominant "background" hypothesis $\{0\}$, meaning that most of the time 0 is the true explanation. Let Y denote the true state; $Y = 0$ denotes the background state. Instead of searching separately for each $\lambda \in \Lambda$, consider a coarse-to-fine search strategy in which we first try to exploit common properties ("shared features") of *all* hypotheses to "test" simultaneously for all $\lambda \in \Lambda$, that is, test the compound hypothesis $H : Y \in \Lambda$ against the alternative $H_0 : Y = 0$. If the test is negative, we stop and declare background; if

the test is positive, we separately test two disjoint subsets of Λ against H_0; and so forth in a nested fashion.

The tests are constructed to be very conservative in the sense that each false negative error rate is very small, that is, given that $Y \in A$, we are very unlikely to declare background if $A \subset \Lambda$ is the subset of hypotheses tested at a given stage. The price for this small false negative error is of course a non-negligible false positive error, particularly for testing "large" subsets A. However, this procedure is highly efficient, particularly under the background hypothesis. This "divide-and-conquer" search strategy has been extensively examined, both algorithmically (see for example Fleuret and Geman, 2001; Amit et al., 2004; Gangaputra and Geman, 2006a) and mathematically (Blanchard and Geman, 2005; Fleuret, 1999; Jung, 2001).

Note: There is an alternate formulation in which Y is directly modeled as a *subset* of Λ with $Y = \emptyset$ corresponding to the background state. In this case, at each node of the hierarchy, we are testing a hypothesis of the form $H : Y \cap A \neq \emptyset$ vs the alternative $Y \cap A = \emptyset$. In practice, the two formulations are essentially equivalent; for instance, in face detection, we can either "decompose" a set of "reference" poses which can represent at most one face and then execute the hierarchical search over subimages or collect all poses into one hierarchy with virtual tests near the root; see Section 7.1. We shall adopt the simpler formulation in which $Y \in \Lambda \cup \{0\}$.

Of course in practice we do all the splitting and construct all the "tests" in advance. (It should be emphasized that we are not constructing a decision tree; in particular, we are recursively partitioning the space of interpretations not features and, when the hierarchy is processed, a data point can travel down many branches and arrive at none of the leaves.) Then, on line, we need only execute the tests in the resulting hierarchy coarse-to-fine. Moreover, the tests are simply standard classifiers induced from training data - examples of $Y \in A$ for various subsets of A and examples of $Y = 0$. In particular, in the case of object detection, the classifiers are constructed from the usual types of image features, such as averages, edges and wavelets (Sahbi et al., 2002).

The nested partitions are naturally identified with a tree T. There is a subset Λ_t for each node t of T, including the root ($\Lambda_{root} = \Lambda$) and each leaf $t \in \partial T$. We will write $t = (l,k)$ to denote the k'th node of T at depth or level l. For example, in the case of a *binary* tree T with L levels, we then have:

$$\begin{cases} \Lambda_{1,1} = \Lambda \\ \Lambda_{l,k} = \Lambda_{l+1,2k-1} \cup \Lambda_{l+1,2k} \\ \Lambda_{l+1,2k-1} \cap \Lambda_{l+1,2k} = \emptyset \end{cases} \qquad l \in \{1,...,L-1\}, \quad k \in \left\{1,...,2^{l-1}\right\}.$$

The hierarchy can be manually constructed (as here, copying the one in Fleuret and Geman, 2001) or, ideally, learned.

Notice that the leaf cells $\Lambda_t, t \in \partial T$, needn't correspond to individual hypotheses. Instead, they represent the finest "resolution" at which we wish to estimate Y. More careful disambiguation among candidate hypotheses may require more intense processing, perhaps involving online optimization. It then makes sense to modify our definition of Y to reflect this possible coarsening of the original classification problem: the possible "class" values are then $\{0,1,...,2^{L-1}\}$, corresponding to "background" ($\{0\}$) and the 2^{L-1} "fine" cells at the leaves of the hierarchy.

Example: The Hierarchy for Face Detection. Here, the *pose* of an object refers to parameters characterizing its geometric appearance in the image. Since we are searching for instances of a

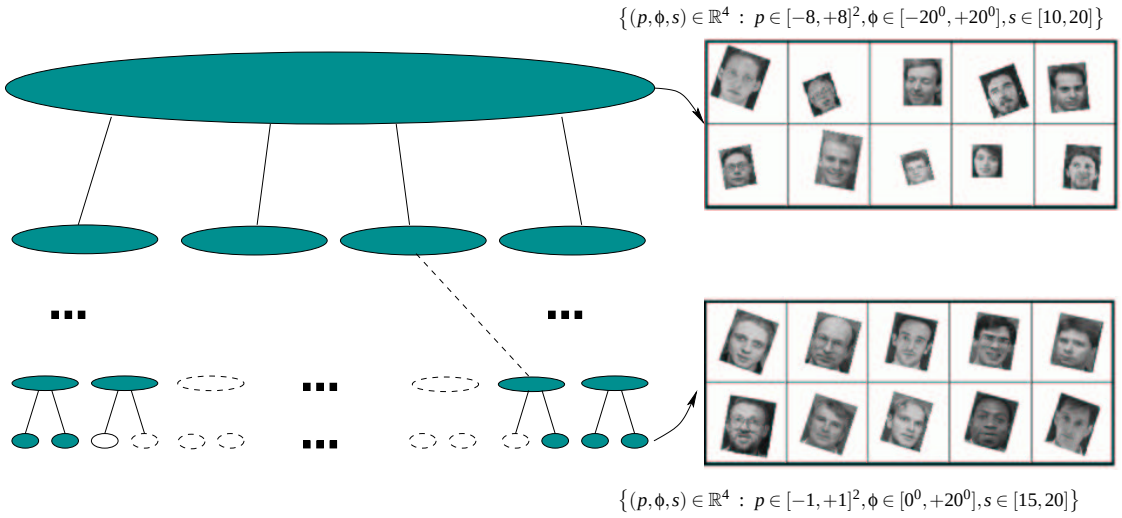


Figure 5: An illustration of the pose hierarchy showing a sample of faces at the root cell and at one of the leaves.

one object class – faces – the family of hypotheses of interest is a set of poses Λ. Specifically, we focus attention on the position, tilt and scale of a face, denoted $\theta = (p, \phi, s)$, where p is the midpoint between the eyes, s is the distance between the eyes and ϕ is the angle with the line orthogonal to the segment joining the eyes. We then define

$$\Lambda = \left\{(p,\phi,s) \in \mathbb{R}^4 \ : \ p \in [-8,+8]^2, \phi \in [-20^0,+20^0], s \in [10,20]\right\}.$$

Thus, we regard Λ as a "reference set" of poses in the sense of possible instantiations of a single face within a given 64×64 image assuming that the position is restricted to a subwindow (e.g., an 16×16 centered in the subimage) and the scale to the stated range. The "background hypothesis" is "no face" (with pose in Λ). The leaves of T do not correspond to individual poses $\theta \in \Lambda$; for instance, the final resolution on position is a 2×2 window. Hence, each "object hypothesis" is a small collection of fine poses.

The specific hierarchy used in our experiments is illustrated in Fig (5). It has six levels $(L = 6)$, corresponding to three quaternary splits in location (four 8×8 blocks, etc.) and one binary split both on tilt and scale. Therefore, writing ν_l for the number of cells in T at depth l: $\nu_1 = 1$, $\nu_2 = 4^1$, $\nu_3 = 4^2 = 16$, $\nu_4 = 4^3 = 64$, $\nu_5 = 2\ 4^3 = 128$ and $\nu_6 = 2^2\ 4^3 = 256$.

This is the same, manually-designed, pose hierarchy that was used in Fleuret and Geman (2001). The partitioning based on individual components, as well as the splitting order, is entirely ad hoc. The important issue of how to automatically design or learn the "divide-and-conquer" architecture is not considered here. Very recent work on this topic appears in Fan (2006) and Gangaputra and Geman (2006a).

Search Strategy:

Consider coarse-to-fine search in more detail. Let X_t be the test or classifier associated with node t, with $X_t = 1$ signaling the acceptance of $H_t : Y \in \Lambda_t$ and $X_t = 0$ signaling the acceptance

of $H_0 : Y = 0$. Also, let $\omega \in \Omega$ represent the underlying data or "pattern" upon which the tests are based; hence the true class of ω is $Y(\omega)$ and $X_t : \Omega \longrightarrow \{0,1\}$.

The result of coarse-to-fine search applied to ω is a *subset* $\mathbf{D}(\omega) \subset \Lambda$ of "detections", possibly empty, defined to be all $\lambda \in \Lambda$ for which $X_t(\omega) = 1$ for every test which "covers" λ, that is, for which $\lambda \in \Lambda_t$. Equivalently, $\mathbf{D}$ is the union over all $\Lambda_t, t \in \partial T$ such that $X_t = 1$ *and the test corresponding to every ancestor of* $t \in \partial T$ *is positive*, that is, all "complete chains of ones" (cf. Fig 6, B).

Both breadth-first and depth-first coarse-to-fine search lead to the same set $\mathbf{D}$. Breadth-first search is illustrated in Fig (6, C): Perform $X_{1,1}$; if $X_{1,1} = 0$, stop and declare $\mathbf{D} = \emptyset$; if $X_{1,1} = 1$, perform *both* $X_{2,1}$ *and* $X_{2,2}$ and stop only if both are negative; etc. Depth-first search explores the sub-hierarchy rooted at a node t before exploring the brother of t. In other words, if $X_t = 1$, we *visit recursively* the sub-hierarchies rooted at t; if $X_t = 0$ we "cancel" all the tests in this sub-hierarchy. In both cases, *a test is performed if and only if all its ancestors are performed and are positive*. (These strategies are not the same if our objective is only to determine whether or not $\mathbf{D} = \emptyset$; see the analysis in Jung (2001).)

Notice that $\mathbf{D} = \emptyset$ if and only if there is a "null covering" of the hierarchy in the sense of a collection of negative responses whose corresponding cells cover all hypotheses in Λ. The search is terminated upon finding such a null covering. Thus, for example, if $X_{1,1} = 0$, the search is terminated as there cannot be a complete chain of ones; similarly, if $X_{2,1} = 0$ and $X_{3,3} = X_{3,4} = 0$, the search is terminated.

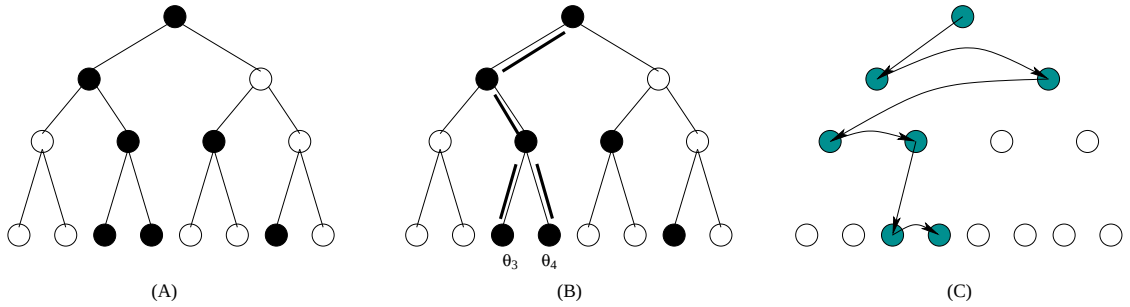


Figure 6: A hierarchy with fifteen tests. (A) The response to an input image were all the tests to be performed; the positive tests are shown in black and negative tests in white. (B) There are two complete chains of ones; in the case of object detection, the detected pose is the average over those in the two corresponding leaves. (C) The breadth-first search strategy with the executed tests are shown in color; notice that only seven of the tests would actually be performed.

Example: The Search Strategy for Face Detection. Images ω are encoded using a vector of wavelet coefficients; in the remainder of this paper we will write x to denote this vector of coefficients computed on a given 64×64 subimage. If $\mathbf{D}(x) \neq \emptyset$, the estimated pose of the face detected in ω is obtained by averaging over the "pose prototypes" of each leaf cell represented in $\mathbf{D}$, where the pose prototype of Λ_t is the midpoint (cf. Fig 6, B).

A scene is processed by visiting *non-overlapping* 16×16 blocks, processing the surrounding image data to extract the features (wavelet coefficients) and classifying these features using the

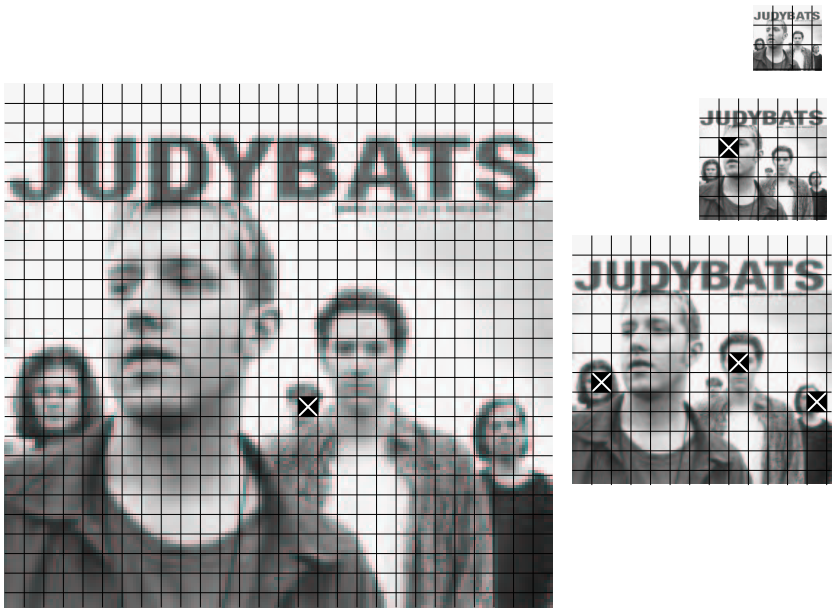


Figure 7: Multi-scale search. The original image (on the left) is downscaled three times. For each scale, the base face detector visits each *non-overlapping* 16×16 block, and searches the surrounding image data for all faces with position in the block, scale anywhere in the range $[10,20]$ and in-plane orientation in the range $[-20^o,+20^o]$.

search strategy described earlier in this section. This process makes it possible to detect all faces whose scale s lies in the interval $[10,20]$ and whose tilt belongs to $[-20^o,+20^o]$. Faces at scales $[20,160]$ are detected by repeated down-sampling (by a factor of 2) of the original image, once for scales $[20,40]$, twice for $[40,80]$ and thrice for $[80,160]$(cf. Fig 7). Hence, due to high invariance to scale in the base detector, only four scales need to be investigated altogether.

Alternatively, we can think of an extended hierarchy over all possible poses, with initial branching into disjoint 16×16 blocks and disjoint scale ranges, and with virtual tests in the first two layers which are passed by all inputs. Given color or motion information (Sahbi and Boujemaa, 2000), it might be possible to design a test which handles a set of poses larger than Λ; however, our test at the root (accounting simultaneously for all poses in Λ) is already quite coarse.

4. Two SVM Hierarchies

Suppose we have a training set $\mathcal{T} = \{(\omega_1, y_1), ..., (\omega_n, y_n)\}$. In the case of object detection, each ω is some 64×64 subimage taken, for example, from the Web, and either belongs to the "object examples" $\mathcal{L}$ (subimages ω for which $Y(\omega) \neq 0$) or "background examples" $\mathcal{B}$ (subimages for which $Y(\omega) = 0$).

All tests $X_t, t \in T$, are based on SVMs. We build one hierarchy, the *free network* or *f-network* for short, with no constraints, that is, in the usual way from the training data once a kernel and any other parameters are specified (Boser et al., 1992). The other hierarchy, the *graded network* or *g-network*, is designed to meet certain error and complexity specifications.

4.1 The f-network

Let f_t be an SVM dedicated to separating examples of $Y \in \Lambda_t$ from examples of $Y = 0$. (In our application to face detection, we train f_t based on face images ω with pose in Λ_t.) The corresponding test is simply $1_{\{f_t > 0\}}$. We refer to $\{f_t, t \in T\}$ as the *f-network*. In practice, the number of support vectors decreases with the depth in the hierarchy since the classification tasks are increasingly simplified; see Fig 2, left. We assume the false negative rate of f_t is very small for each t; in other words, $f_t(\omega) > 0$ for nearly all patterns ω for which $Y(\omega) \in \Lambda_t$. Finally, denote the corresponding data-dependent set of detections of the f-network by $\mathbf{D}_f$.

4.2 The g-network

The *g-network* is based on the same hierarchy $\{\Lambda_t\}$ as the f-network. However, for each cell Λ_t, a simplified SVM decision function g_t is built by reducing the complexity of the corresponding classifier f_t. The set of hypotheses detected by the g-network is denoted by $\mathbf{D}_g$. The targeted complexity of g_t is determined by solving a constrained minimization problem (cf. Section 5).

We want g_t to be *both* efficient *and* respect the constraint of a negligible false negative rate. As a result, for nodes t near the root of T the false positive rate of g_t will be higher than that of the corresponding f_t since low cost comes at the expense of a weakened background filter. Put differently, we are willing to sacrifice selectivity for efficiency, but not at the expense of missing (many) instances of our targeted hypotheses. Thus, for both networks, a positive test by no means signals the presence of a targeted hypothesis, especially for the very computationally efficient tests in the g-network near the top of the hierarchy.

Instead of imposing an absolute constraint on the false negative error, we impose one *relative* to the f-network, referred to as the *conservation hypothesis*: For each $t \in T$ and $\omega \in \Omega$:

$$f_t(\omega) > 0 \Rightarrow g_t(\omega) > 0.$$

This implies that an hypothesis detected by the f-network is also detected by the g-network, namely

$$\mathbf{D}_f(\omega) \subset \mathbf{D}_g(\omega), \ \forall \omega \in \Omega.$$

Consider two classifiers g_t and g_s in the g-network and suppose node s is deeper than node t. With the *same* number of support vectors, g_t will generally produce more false alarms than g_s since more invariance is expected of g_t (cf. Fig 2, right). In constructing the g-network, all classifiers at the same level will have the same number of support vectors and are then expected to have approximately the same false alarm rate (cf. Fig 8).

In the following sections, we will introduce a model which accounts for both the overall mean cost and the false alarm rate. This model is inspired by the trade-offs among selectivity, cost and invariance discussed above. The proposed analysis is performed under the assumption that there exists a convex function which models the false alarm rate as a function of the number of support vectors and the degree of "pose invariance".

5. Designing the g-network

Let P be a probability distribution on Ω. Write $\Omega = \mathcal{L} \cup \mathcal{B}$, where $\mathcal{L}$ denotes the set of all possible patterns for which $Y(\omega) > 0$, that is, $Y \in \{1, ..., 2^{L-1}\}$, hence a targeted pattern, and $\mathcal{B} = \mathcal{L}^c$

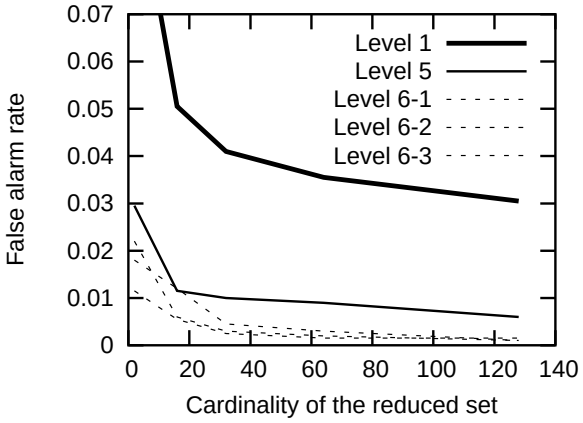


Figure 8: For the root cell, a particular cell in the fifth level and three particular pose cells in the sixth level of the g-network, we built SVMs with varying numbers of (virtual) support vectors. All curves show false alarm rates with respect to the number of (virtual) support vectors. For the sixth level, and in the regime of fewer than 10 (virtual) support vectors, the false alarm rates show considerable variation, but have the same order of magnitude. These experiments were run on background patterns taken from 200 images including highly textured areas (flowers, houses, trees, etc.)

contains all the background patterns. Define $P_0(.) = P(.|Y=0)$ and $P_1(.) = P(.|Y>0)$, the conditional probability distributions on background and object patterns, respectively. Throughout this paper, we assume that $P(Y=0) >> P(Y>0)$, which means that the presence of the targeted pattern is considered to be a rare event in data sampled under P.

Face Detection Example (cont): We might take P to be the empirical distribution on a huge set of 64×64 subimages taken from the Web. Notice that, given a subimage selected at random, the probability to have a face present with location near the center is very small.

Relative to the problem of deciding $Y = 0$ vs $Y \neq 0$, that is, deciding between "background" and "object" (some hypothesis in Λ), the two error rates for the f-network are $P_0(\mathbf{D}_f \neq \emptyset)$, the false positive rate, and $P_1(\mathbf{D}_f = \emptyset)$, the false negative rate. The total error rate is, $P_0(\mathbf{D}_f \neq \emptyset)P(Y=0) + P_1(\mathbf{D}_f = \emptyset)\ P(Y>0)$. Clearly this total error is largely dominated by the false alarm rate.

Recall that for each node $t \in T$ there is a subset Λ_t of hypotheses and an SVM classifier g_t with n_t support vectors. The corresponding test for checking $Y \in \Lambda_t$ against the background alternative is $1_{\{g_t>0\}}$. Our objective is to provide an optimization framework for specifying $\{n_t\}$.

5.1 Statistical Model

We now introduce a *statistical model* for the behavior of the g-network. Consider the event that a background pattern traverses the hierarchy up to node t, namely the event $\bigcap_{s\in\mathcal{A}_t}\{g_s > 0\}$, where $\mathcal{A}_t$ denotes the set of ancestors of t – the nodes from the parent of t to the root, inclusive. We will

assume that the probability of this event under P_0, namely

$$P_0(g_s > 0, s \in \mathcal{A}_t), \tag{1}$$

depends only on the level of t in the hierarchy (and not on the particular set $\mathcal{A}_t$) as well as on the numbers $n_1, ..., n_{l-1}$ of support vectors at levels one through $l-1$, where t is at level l. These assumptions are reasonable and roughly satisfied in practice; see Sahbi (2003).

The probability in (1) that a background pattern reaches depth l, that is, there is a chain of positive responses of length $l-1$, is then denoted by $\delta(l-1;\mathbf{n})$, where $\mathbf{n}$ denotes the sequence $(n_1, ..., n_L)$. Naturally, we assume that $\delta(l;\mathbf{n})$ is decreasing in l. In addition, it is natural to assume that $\delta(l;\mathbf{n})$ is a decreasing function of each $n_j, 1 \leq j \leq l$. In Section 7 we will present an example of a two-dimensional parametric family of such models.

There is an equivalent, and useful, reformulation of these joint statistics in terms of *conditional false alarm rates* (or conditional selectivity). One specifies a model by prescribing the quantities

$$P_0(g_{root} > 0),\ P_0(g_t > 0 | g_s > 0, s \in \mathcal{A}_t) \tag{2}$$

for all nodes t with $2 \leq l(t) \leq L$. Clearly, the probabilities in (1) determine those in (2) and vice-versa.

Note: We are not specifying a probability distribution on the entire family of variables $\{g_t, t \in T\}$, equivalently, on all labeled trees. However, it can be shown that any (decreasing) sequence of positive numbers $p_1, ..., p_L$ for the chain probabilities is "consistent" in the sense of providing a well-defined distribution on *traces*, the labeled subtrees that can result from coarse-to-fine processing, which necessarily are labeled "1" at all internal nodes; see Gangaputra and Geman (2006b).

In order to achieve efficient computation (at the expense of extra false alarms relative to the f-network), we choose $\mathbf{n} = (n_1, ..., n_L)$ to solve a constrained minimization problem based on the mean total computation in evaluating the g-network and a bound on the expected number of detected background patterns:

$$\begin{array}{ll} \min\limits_{\mathbf{n}} & C(n_1, ..., n_L) \\ \text{s.t.} & E_0(|\mathbf{D}_g|) \ \leq \ \mu \end{array} \tag{3}$$

where E_0 refers to expectation with respect to the probability measure P_0. We first compute this expected cost, then consider the constraint in more detail and finally turn to the problem of choosing the model.

Note: In our formulation, we are assuming that overall average computation is well-approximated by estimating total computation under the background probability by itself rather than with respect to a mixture model which accounts for object instances. In other words, we are assuming that background processing accounts for most of the work. Of course, in reality, this is not strictly the case, especially at the lower levels of the hierarchy, at which point evidence has accrued for the presence of objects and the conditional likelihoods of object and background are no longer extremely skewed in favor of the latter. However, computing under a mixture model would severely complicate the analysis. Moreover, since extensive computation is rarely performed, we believe our approximation is valid; whereas an expanded analysis might somewhat change the design of the lower levels, it would not appreciably reduce overall cost.

5.2 Cost of the g-network

Let c_t indicate the cost of performing g_t and assume

$$c_t = a\, n_t + b.$$

Here a represents the cost of kernel evaluation and b represents the cost of "preprocessing" – mainly extracting features from a pattern ω (e.g., computing wavelet coefficients in a subimage). We will also assume that all SVMs at the same level of the hierarchy have the same number of support vectors, and hence approximately the same cost.

Recall that ν_l is the number of nodes in T at level l; for example, for a binary tree, $\nu_l = 2^{l-1}$. The global cost is then:

$$\begin{aligned} Cost &= \sum_t 1_{\{g_t \text{ is performed}\}}\, c_t \\ &= \sum_{l=1}^{L} \sum_{k=1}^{\nu_l} 1_{\{g_{l,k} \text{ is performed}\}}\, c_{l,k} \end{aligned} \tag{4}$$

since g_t is performed in the coarse-to-fine strategy if and only if $g_s > 0\ \forall s \in \mathcal{A}_t$, we have, from equation (4), with $\delta(0;\mathbf{n}) = 1$,

$$\begin{aligned} \mathcal{C}(n_1,...,n_L) &= E_0\,(Cost) \\ &= \sum_{l=1}^{L} \sum_{k=1}^{\nu_l} P_0\,(\{g_{l,k} \text{ is performed}\})\ c_{l,k} \\ &= \sum_{l=1}^{L} \sum_{k=1}^{\nu_l} \delta(l-1;\mathbf{n})\, c_{l,k} \\ &= \sum_{l=1}^{L} \nu_l\, \delta(l-1;\mathbf{n})\, c_l \\ &= a \sum_{l=1}^{L} \nu_l\, \delta(l-1;\mathbf{n})\, n_l + b \sum_{l=1}^{L} \nu_l\, \delta(l-1;\mathbf{n}). \end{aligned}$$

The first term is the SVM cost and the second term is the total preprocessing cost. In the application to face detection we shall assume the preprocessing cost – the computation of Haar wavelet coefficients for a given subimage – is small compared with kernel evaluations, and set $a = 1$ and $b = 0$. Hence,

$$\mathcal{C}(n_1,...,n_L) = n_1 + \sum_{l=2}^{L} \nu_l\, n_l\, \delta(l-1;\mathbf{n}).$$

5.3 Penalty for False Detections

Recall that $\mathbf{D}_g(\omega)$ – the set of detections – is the union of the sets Λ_t over all *terminal* nodes t for which there is a complete chain of positive responses from the root to t. For simplicity, we assume that $|\Lambda_t|$ is the same for all terminal nodes t. Hence $|\mathbf{D}_g|$ is proportional to the total number of complete chains:

$$|\mathbf{D}_g| \propto \sum_{t \in \partial T} 1_{\{g_t > 0\}} \prod_{s \in \mathcal{A}_t} 1_{\{g_s > 0\}}.$$

It follows that

$$\begin{aligned} E_0|\mathbf{D}_g| &\propto E_0 \sum_{t\in\partial T} 1_{\{g_t>0\}} \prod_{s\in\mathcal{A}_t} 1_{\{g_s>0\}} \\ &= \sum_{t\in\partial T} \delta(L;\mathbf{n}) \\ &= \nu_L \delta(L;\mathbf{n}) \end{aligned}$$

By the Markov inequality,

$$P_0(\mathbf{D}_g \neq \emptyset) = P_0(|\mathbf{D}_g| \geq 1) \leq E_0|\mathbf{D}_g|.$$

Hence bounding the mean size of $\mathbf{D}_g$ also yields the same bound on the false positive probability. However, we cannot calculate $P_0(|\mathbf{D}_g| \geq 1)$ based only on our model $\{\delta(l;\mathbf{n})\}_l$ since this would require computing the probability of a *union* of events and hence a model for the dependency structure *among* chains.

Finally, since we are going to use the SVMs in the f-network to build those in the g-network, the number of support vectors n_l for each SVM in the g-network at level l is bounded by the corresponding number, N_l, for the f-network. (Here, for simplicity, we assume that N_t is roughly constant in each level; otherwise we take the minimum over the level.)

Summarizing, our constrained optimization problem (3) becomes

$$\min_{n_1,\ldots,n_L} \; n_1 + \sum_{l=2}^{L} \nu_l \, n_l \, \delta(l-1;\mathbf{n}) \;\; \text{s.t.} \;\; \begin{cases} \nu_L \delta(L;\mathbf{n}) \leq \mu \\ 0 < n_l \leq N_l. \end{cases} \tag{5}$$

5.4 Choice of Model

In practice, one usually stipulates a *parametric family* of statistical models (in our case the chain probabilities) and estimates the parameters from data. Let $\{\delta(l;\mathbf{n},\beta)\}, \beta \in B\}$ denote such a family where β denotes a parameter vector, and let $\mathbf{n}^* = \mathbf{n}^*(\beta)$ denote the solution of (5) for model β. We propose to choose β by comparing population and empirical statistics. For example, we might select the model for which the chain probabilities best match the corresponding relative frequencies when the g-network is constructed with $\mathbf{n}^*(\beta)$ and run on sample background data. Or, we might simply compare predicted and observed numbers of background detections:

$$\beta^* \doteq \arg\min_{\beta\in B} |E_0\left(|\mathbf{D}_g|;\mathbf{n}^*(\beta)\right) - \hat{\mu}_0(\mathbf{n}^*(\beta))|$$

where $\hat{\mu}_0(\mathbf{n}^*(\beta))$ is the average number of detections observed with the g-network constructed from $\mathbf{n}^*(\beta)$. In Section 7 we provide a concrete example of this model estimation procedure.

6. Building the g-network

Since the construction is node-by-node, we can assume throughout this section that t is fixed and that $\Lambda = \Lambda_t$ and $f = f_t$ are given, where f is an SVM with N_f support vectors. Our objective is to build an SVM $g = g_t$ with two properties:

- g is to have $N_g < N_f$ support vectors, where $N_g = n^*_{l(t)}$ and $\mathbf{n}^* = (n^*_1,\ldots,n^*_L)$ is the optimal design resulting from the upcoming reformulation (8) of (5) which incorporates our model for $\{\delta(l:\mathbf{n})\}$; and

- The "conservation hypothesis" is satisfied; roughly speaking this means that detections under f are preserved under g.

Let $x(\omega) \in \mathbb{R}^q$ be the feature vector; for simplicity, we suppress the dependence on ω. The SVM f is constructed as usual based on a kernel K which implicitly defines a mapping Φ from the input space $\mathbb{R}^q$ into a high-dimensional Hilbert space $\mathcal{H}$ with inner product denoted by $\langle\ \rangle$. Support vector training (Boser et al., 1992) builds a maximum margin hyperplane (w_f, b_f) in $\mathcal{H}$. Re-ordering the training set as necessary, let $\left\{\Phi(v^{(1)}), ..., \Phi(v^{(N_f)})\right\}$ and $\{\alpha_1, ..., \alpha_{N_f}\}$ denote, respectively, the support vectors and the training parameters. The normal w_f of the hyperplane is $w_f = \sum_{i=1}^{N_f} \alpha_i\, y^{(i)}\, \Phi(v^{(i)})$.

The decision function in $\mathbb{R}^q$ is non-linear:

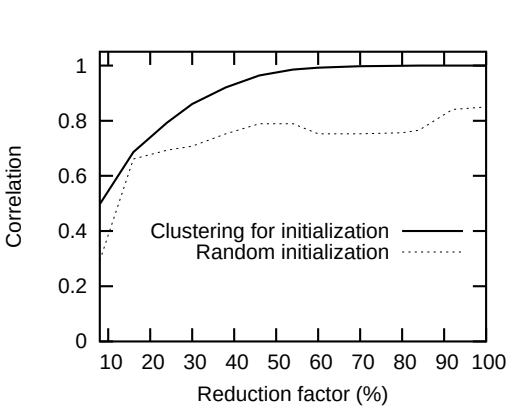
$$f(x) = \langle w_f, \Phi(x) \rangle + b_f = \sum_{i=1}^{N_f} \alpha_i\, y^{(i)}\, K(v^{(i)}, x) + b_f.$$

The parameters $\{\alpha_i\}$ and b_f can be adjusted to ensure that $\|w_f\|^2 = 1$.

The objective now is to determine (w_g, b_g), where

$$g(x) = \langle w_g, \Phi(x) \rangle + b_g = \sum_{k=1}^{N_g} \gamma_k\, K(z^{(k)}, x) + b_g.$$

Here $\mathcal{Z} = \left\{z^{(1)}, ..., z^{(N_g)}\right\}$ is the reduced set of support vectors (cf. Fig 9, right), $\gamma = \left\{\gamma_1, ..., \gamma_{N_g}\right\}$ the underlying weights, and the labels $y^{(k)}$ have been absorbed into γ.



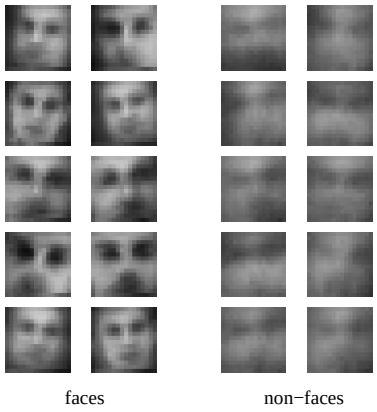


Figure 9: Left: The correlation is illustrated with respect to the reduction factor ($\frac{N_g}{N_f} \times 100$). These experiments are performed on a root SVM classifier using the Gaussian kernel (σ is set to 100 using cross-validation). Right: Visual appearance of some face and "non-face" virtual support vectors.

6.1 Determining w_g: Reduced Set Technique

The method we use is a modification of the *reduced set technique* (RST) (Burges, 1996; Burges and Schölkopf, 1997). Choose the new normal vector to satisfy:

$$w_g^* = \arg \min_{w_g : \|w_g\|=1} \langle w_g - w_f, w_g - w_f \rangle.$$

Using the kernel trick, and since $w_g = \sum_{k=1}^{N_g} \gamma_k \, \Phi(z^{(k)})$, the new optimization problem becomes:

$$\min_{\mathcal{Z},\gamma} \quad \sum_{k,l} \gamma_k \gamma_l \, K(z^{(k)}, z^{(l)}) \; + \; \sum_{i,j} \alpha_i \alpha_j \, y^{(i)} \, y^{(j)} \, K(v^{(i)}, v^{(j)}) - 2 \sum_{k,i} \gamma_k \alpha_i \, y^{(i)} \, K(z^{(k)}, v^{(i)}). \qquad (6)$$

For some kernels (for instance the Gaussian), the function to be minimized is not convex; consequently, with standard optimization techniques such as conjugate gradient, the normal vector resulting from the final solution $(\mathcal{Z}, \gamma)$ is a poor approximation to w_f (cf. Fig 9, left). This problem was analyzed in Sahbi (2003), where it is shown that, in the context of face detection, a good initialization of the minimization process can be obtained as follows: First cluster the initial set of support vectors $\{\Phi(v^{(1)}), ..., \Phi(v^{(N_f)})\}$, resulting in N_g centroids, each of which then represents a dense distribution of the original support vectors. Next, each centroid, which is expressed as a linear combination of original support vectors, is replaced by one support vector which best approximates this linear combination. Finally, this new reduced set is used to initialize the search in (6) in order to improve the final solution. Details may be found in Sahbi (2003) and the whole process is illustrated in Section 7.

6.2 Determining b_g: Conservation Hypothesis

Regardless of how b_g is selected, g is clearly less powerful than f. However, in the hierarchical framework, particularly near the root, the two types of mistakes (namely not detecting patterns in Λ and detecting background) are not equally problematic. Once a distinguished pattern is rejected from the hierarchy it is lost forever. Hence we prefer to severely limit the number of missed detections at the expense of additional false positives; hopefully these background patterns will be filtered out before reaching the leaves.

We make the assumption that the classifiers in the f-network have a very low false negative rate. Ideally, we would choose b_g such that $g(x(\omega)) > 0$ for every $\omega \in \Omega$ for which $f(x(\omega)) > 0$. However, this results in an unacceptably high false positive rate. Alternatively, we seek to minimize

$$P_0 (\, g \, \geq \, 0 \mid f \, < \, 0 \,)$$

subject to

$$P_1 (\, g \, < \, 0 \mid f \, \geq 0 \,) \leq \varepsilon.$$

Since we do not know the joint law of (f, g) under either P_0 or P_1, these probabilities are estimated empirically: for each b_g calculate the conditional relative frequencies using the training data and then choose the optimal b_g based on these estimates.

7. Application to Face Detection

We apply the general construction of the previous sections to a particular two-class problem – face detection – which has been widely investigated, especially in the last ten years. Existing methods

include artificial neural networks (Schneiderman and Kanade, 2000; Sung, 1996; Rowley et al., 1998; Féraud et al., 2001; Garcia and Delakis, 2004), networks of linear units (Yang et al., 2000), support vector machines (Osuna et al., 1997; Evgeniou et al., 2000; Heisele et al., 2001; Romdhani et al., 2001; Kienzle et al., 2004), Bayesian inference (Cootes et al., 2000), deformable templates (Miao et al., 1999), graph-matching (Leung et al., 1995), skin color learning (Hsu et al., 2001; Sahbi and Boujemaa, 2000), and more rapid techniques such as boosting a cascade of classifiers (Viola and Jones, 2001; Li and Zhang, 2004; Wu et al., 2005; Socolinsky et al., 2003; Elad et al., 2002) and hierarchical coarse-to-fine processing (Fleuret and Geman, 2001).

The face hierarchy was described in Section 3. We now introduce a specific model for (1), the probability of a chain under the background hypothesis, and finally the solution to the resulting instance of the constrained optimization problem expressed in (8) below. The probability model links the cost of the SVMs to their underlying level of invariance and selectivity. Afterwords, in Section 8, we illustrate the performance of the designed g-network in terms of speed and error on both simple and challenging face databases including the CMU and the MIT datasets.

7.1 Chain Model

Our model family is $\{\delta(l;\mathbf{n},\beta),\beta\in B\}$, where $\delta(l;\mathbf{n},\beta)$ is the probability of a chain of "ones" of depth $l-1$. These probabilities are determined by the conditional probabilities in (2). Denote these by

$$\delta(1;\mathbf{n},\beta)=P_0(g_{root}>0)$$

and

$$\delta(l\mid 1,...,l-1;\mathbf{n},\beta)=P_0(g_t>0|g_s>0,s\in\mathcal{A}_t).$$

Specifically, we take:

$$\begin{aligned}\delta(1;\mathbf{n},\beta) &= \frac{1}{\beta_1 n_1} && \beta_1>0\\ \delta(l\mid 1,...,l-1;\mathbf{n},\beta) &= \frac{\beta_1 n_1+...+\beta_{l-1}n_{l-1}}{\beta_1 n_1+...+\beta_{l-1}n_{l-1}+\beta_l n_l} && ,\quad \beta_1,...,\beta_l>0.\end{aligned}$$

Loosely speaking, the coefficients $\beta=\{\beta_j \quad j=1,...,L\}$ are inversely proportional to the degree of "pose invariance" expected from the SVMs at different levels. At the upper, highly invariant, levels l of the g-network, minimizing computation yields relatively small values of β_l and vice-versa at the lower, pose-dedicated, levels. The motivation for this functional form is that the conditional false alarm rate $\delta(l\mid 1,...,l-1;\mathbf{n},\beta)$ should be *increasing* as the number of support vectors in the upstream levels $1,...,l-1$ increases. Indeed, when $g_s>0$ for all nodes s upstream of node t, and when these SVMs have a large number of support vectors and hence are very selective, the background patterns reaching node t resemble faces very closely and are likely to be accepted by the test at t. Of course, fixing the numbers of support vectors upstream, the conditional selectivity (that is, one minus the false positive error rate) at level l grows with n_l. *Notice also that the model does not anticipate exponential decay, corresponding to independent tests (under P_0) along the branches of the hierarchy.*

Using the marginal and the conditional probabilities expressed above, the probability $\delta(l;\mathbf{n},\beta)$ to have a chain of ones from the root cell to any particular cell at level l is easily computed:

$$\delta(l;\mathbf{n},\beta) = \frac{1}{\beta_1 n_1}\,\frac{\beta_1\, n_1}{\beta_1\, n_1+\beta_2\, n_2}...\frac{\beta_1\, n_1+...+\beta_{l-1}n_{l-1}}{\beta_1 n_1+...+\beta_l n_l}$$

$$= \left(\sum_{j=1}^{l} \beta_j n_j \right)^{-1}. \tag{7}$$

Clearly, for any $\mathbf{n}$ and β, these probabilities decrease as l increases.

7.2 The Optimization Problem

Using (7), the constrained minimization problem (5) becomes:

$$\begin{array}{ll} \min\limits_{n_1,\ldots,n_L} & \left[n_1 + \sum\limits_{l=2}^{L} \left(\sum\limits_{i=1}^{l-1} \beta_i n_i \right)^{-1} \nu_l \; n_l \right] \\ \text{s.t.} & \left\{ \begin{array}{l} \nu_L \left(\sum\limits_{i=1}^{L} \beta_i n_i \right)^{-1} \leq \mu \\ 0 < n_l \leq N_l. \end{array} \right. \end{array} \tag{8}$$

This problem is solved in two steps:

- **Step I:** Start with the solution for a binary network (i.e., $\nu_{l+1} = 2\nu_l$). This solution is provided in Appendix A.
- **Step II:** Pass to a dyadic network using the solution to the binary network, as shown in Sahbi (2003, p. 127).

7.3 Model Selection

We use a simple function with two degrees of freedom to characterize the growth of $\beta_1, \ldots, \beta_L$:

$$\beta_l = \Psi_1^{-1} \; \exp\{\Psi_2 (l-1)\} \tag{9}$$

where $\Psi = (\Psi_1, \Psi_2)$ are positive. Here Ψ_1 represents the degree of pose invariance at the root cell and Ψ_2 is the rate of the decrease of this invariance. Let $\mathbf{n}^*(\Psi)$ denote the solution to (8) for β given by (9) and suppose we restrict $\Psi \in Q$, a discrete set. (In our experiments, $|Q| = 100$ corresponding to ten choices for each parameter Ψ_1 and Ψ_2, namely Ψ_1 ranges from 0.0125 to 0.2 and Ψ_2 ranges from 0.1 to 1.0, both in equal steps.) In other words, $\mathbf{n}^*(\Psi)$ are the optimal numbers of support vectors found when minimizing total computation (8) under the false positive constraint for a given fixed $\beta = \{\beta_1, \ldots, \beta_L\}$ determined by (9). Then Ψ^* is selected to minimize the discrepancy between the model and empirical conditional false positive rates:

$$\min_{\Psi \in Q} \sum_{l=1}^{L} \left| \delta(l \mid 1, \ldots, l-1; \mathbf{n}^*(\Psi), \Psi) - \hat{\delta}(l \mid 1, \ldots, l-1; \mathbf{n}^*(\Psi)) \right| \tag{10}$$

where $\hat{\delta}(l \mid 1, \ldots, l-1; \mathbf{n}^*(\Psi))$ is the underlying empirical probability of observing $g_t > 0$ given that $g_s > 0$ for all ancestors s of t, averaged over all nodes t at level l, when the g-network is built with $\mathbf{n}^*(\Psi)$ support vectors.

In practice, it takes 3 days (on a 1-Ghz pentium-III) to implement this design, which includes building the f-network (SVM training), solving the minimization problem (8) for different instances

Algorithm: *Design of the g-network.*

- Build the f-network using standard SVM training and learning set $\mathcal{L} \cup \mathcal{B}$
- **for** $(\Psi_1, \Psi_2) \in Q$ **do**
  - $\beta_l \leftarrow \Psi_1^{-1} \exp\{\Psi_2(l-1)\}$, $l = 1, ..., L$
  - Compute $\mathbf{n}^*(\Psi)$ using (8).
  - Build the g-network using the reduced set method and the specified costs $\mathbf{n}^*(\beta)$.
  - Compute the model and empirical conditional probabilities $\delta(l \mid 1, ..., l-1; \mathbf{n}^*(\Psi), \Psi)$ and $\hat{\delta}(l \mid 1, ..., l-1; \mathbf{n}^*(\Psi))$.

end

- $\Psi^* \leftarrow$ (10)
- The specification for the g-network is $\mathbf{n}^*(\Psi^*)$

of $\Psi = (\Psi_1, \Psi_2)$ and applying the reduced set technique (6) for each sequence of costs $n^*(\Psi)$ in order to build the g-network.

When solving the constrained minimization problem (8) (cf. Appendix A), we find the optimal numbers $n_1^*, ..., n_6^*$ of support vectors, rounded to the nearest even integer, are given by:

$$\mathbf{n}^* = \{2,\ 2,\ 2,\ 4,\ 8,\ 22\},$$

(cf. table 2), corresponding to $\Psi^* = ((\Psi_1^{-1})^*, \Psi_2^*) = (7.27, 0.55)$, resulting in $\beta^* = \{7.27,\ 25.21,\ 87.39,\ 302.95,\ 525.09,\ 910.11\}$. For example, we estimate

$$P_0(g_{root} > 0) = \frac{1}{2 \times 7.27} = 0.069$$

the false positive rate at the root.

The empirical conditional false alarms were estimated on background patterns taken from 200 images including highly textured areas (flowers, houses, trees, etc.). The conditional false positive rates for the model and the empirical results are quite similar, so that the cost in the objective function (8) approximates effectively the observed cost. In fact, when evaluating the objective function in (8), the average cost was 3.379 kernel evaluations per pattern whereas in practice this average cost was 3.196 per pattern taken from scenes including highly textured areas.

Again, the coefficients β_l^* and the complexity n_l^* of the SVM classifiers are increasing as we go down the hierarchy, which demonstrates that the best architecture of the g-network is low-to-high in complexity.

7.4 Features and Parameters

Many factors intervene in fitting our cost/error model to real observations (the conditional false alarms), including the size of the training sets and the choice of features, kernels and other parameters, such as the bound on the expected number of false alarms. Obviously the nature of the resulting g-network can be sensitive to variations of these factors. We have only used wavelet features, the Gaussian kernel, selecting the parameters by cross-validation, and the very small ORL database.

Whereas we have not done systematic experiments to analyze sensitivity, it is reasonable to suppose that having more data or more powerful features would increase performance. For instance,

Ψ_1	Ψ_2	L_1 error	Number of support vectors per level					
	$\Psi_2 = .70$	0.980	1.03	1.08	1.42	1.87	4.98	16.375
$\Psi_1 = .2000$	$\Psi_2 = .55$	0.809	1.49	2.35	3.69	4.99	11.64	32.12
	$\Psi_2 = .30$	1.207	2.70	8.13	17.45	24.36	42.89	93.50
	$\Psi_2 = .70$	1.129	1.00	1.08	1.46	1.98	5.40	18.12
$\Psi_1 = .1875$	$\Psi_2 = .55$	0.750	1.39	2.20	3.46	4.68	10.91	30.12
	$\Psi_2 = .30$	1.367	2.53	7.62	16.36	22.83	40.20	87.62
	$\Psi_2 = .70$	0.995	1.00	1.18	1.70	2.46	7.20	25.50
$\Psi_1 = .1750$	$\Psi_2 = .55$	0.766	1.30	2.05	3.23	4.37	10.19	28.12
	$\Psi_2 = .30$	1.401	2.36	7.12	15.27	21.32	37.56	81.87
	$\Psi_2 = .70$	0.981	1.00	1.29	2.00	3.13	9.84	37.00
$\Psi_1 = .1625$	$\Psi_2 = .55$	0.752	1.21	1.91	3.00	4.06	9.46	26.12
	$\Psi_2 = .30$	1.428	2.19	6.61	14.18	19.80	34.86	76.00
	$\Psi_2 = .70$	0.839	1.00	1.41	2.38	4.03	13.74	55.12
$\Psi_1 = .1500$	$\Psi_2 = .55$	0.605	1.11	1.76	2.77	3.75	8.74	24.12
	$\Psi_2 = .30$	1.339	2.02	6.10	13.09	18.27	32.17	70.12
	$\Psi_2 = .70$	1.137	1.00	1.56	2.87	5.30	19.72	85.12
Ψ_1=.1375	**Ψ_2=.55**	**0.557**	**1.02**	**1.61**	**2.54**	**3.43**	**8.01**	**22.12**
	$\Psi_2 = .30$	1.580	1.85	5.59	11.99	16.74	29.47	64.25
	$\Psi_2 = .70$	1.230	1.00	1.75	3.53	7.16	29.29	137.12
$\Psi_1 = .1250$	$\Psi_2 = .55$	0.808	1.00	1.71	2.89	4.21	10.56	30.62
	$\Psi_2 = .30$	1.441	1.69	5.08	10.91	15.23	26.83	58.50
	$\Psi_2 = .70$	NS	-	-	-	-	-	-
$\Psi_1 = .1000$	$\Psi_2 = .55$	0.958	1.00	2.21	4.69	8.60	27.23	93.37
	$\Psi_2 = .30$	0.687	1.35	4.06	8.72	12.18	21.44	46.75
	$\Psi_2 = .70$	NS	-	-	-	-	-	-
$\Psi_1 = .0750$	$\Psi_2 = .55$	NS	-	-	-	-	-	-
	$\Psi_2 = .30$	1.242	1.01	3.05	6.55	9.14	16.11	35.12

Table 2: A sample of the simulation results. Shown, for selected values of (Ψ_1, Ψ_2), are the L_1 error in (10) and also the numbers of support vectors which minimize cost. In practice 10×10 possible values of Ψ_1 and Ψ_2 are considered ($\Psi_1 \in [0, 0.2]$ and $\Psi_2 \in [0.1, 1.0]$). (*NS* stands for "no solution", L_1 refers to the sum of absolute differences, and the bold line is the optimal solution.)

with highly discriminating features, the separation between the positive and negative examples used for training the f-network might be sufficient to allow even linear SVMs to produce accurate decision boundaries, in which case very few support vectors might be required in the f-network. The leave-one-out error bound (see for instance Vapnik, 1998) would then suggest low error rates. Accordingly, in principle, the g-network could be designed with few (virtual) support vectors while satisfying the false alarm bound in (3). The features we use – the Haar wavelet coefficients – are generic and not especially powerful. The only other ones we tried were Daubechies wavelets, which

were abandoned due to extensive computation; their performance is unknown. Similar arguments apply to the choice of kernels and their parameters; for instance the scale of the Gaussian kernel controls influences both the error rate and the number of support vectors in the f-network (and also in the g-network.)

8. Experiments

All the training images of faces are based on the Olivetti database of 400 gray level pictures – ten frontal images for each of forty individuals. The coordinates of the eyes and the mouth of each picture were labeled manually. Most other methods (see below) use a far larger training set, in fact, usually ten to one hundred times larger. In our view, the smaller the better in the sense that the number of examples is a measure of performance along with speed and accuracy. Nonetheless, this criterion is rarely taken into account in the literature on face detection (and more generally in machine learning).

In order to sample the pose variation within Λ_t, for each face image in the original Olivetti database, we synthesize 20 images of 64×64 pixels with randomly chosen poses in Λ_t. Thus, a set of $8,000$ faces is synthesized for each pose cell in the hierarchy. Background information is collected from a set of $1,000$ images taken from 28 different topical databases (including auto-racing, beaches, guitars, paintings, shirts, telephones, computers, animals, flowers, houses, tennis, trees and watches), from which $50,000$ subimages of 64×64 pixels are randomly extracted.

Given coarse-to-fine search, the "right" alternative hypothesis at a node is "path-dependent". That is, the appropriate "negative" examples to train against at a given node are those data points which pass all the tests from the root to the parent of the node. As with cascades, this is what we do in practice; more precisely, we merge a fixed collection of background images with a "path-dependent" set (for details see Sahbi, 2003, chap. 4).

Each subimage, either a face or background, is encoded using the 16×16 low frequency coefficients of the Haar wavelet transform computed efficiently using the integral image (Sahbi, 2003; Viola and Jones, 2001). Thus, only the coefficients of the third layer of the wavelet transform are used; see Chapter 2 of Sahbi (2003). The set of face and background patterns belonging to Λ_t are used to train the underlying SVM f_t in the f-network (using a Gaussian kernel).

8.1 Clustering Detections

Generally, a face will be detected at several poses; similarly, false positives will often be found in small clusters. In fact, every method faces the problem of clustering detections in order to provide a reasonable estimate of the "false alarm rate," rendering comparisons somewhat difficult.

The search protocol was described in Section 7.1. It results in a set of detections $\mathbf{D}_g$ for each non-overlapping 16×16 block in the original image and each such block in each of three downsampled images (to detect larger faces). All these detections are initially collected. Evidently, there are many instances of two "nearby" poses which cannot belong to two distinct, fully visible faces. Many ad hoc methods have been designed to ameliorate this problem. We use one such method adapted to our situation: For each hierarchy, we sum the responses of the SVMs at the leaves of each complete chain (i.e., each detection in $\mathbf{D}_g$) and remove all the detections from the aggregated list unless this sum exceeds a learned threshold τ, in which case $\mathbf{D}_g$ is represented by a single "average" pose. In other words, we declare that a block contains the location of a face if the *aggregate SVM score* of the classifiers in the leaf-cells of complete chains is above τ. In this way, the false negative rate does

not increase due to pruning and yet some false positives are removed. Incompatible detections can and do remain.

Note: One can also implement a "voting" procedure to arbitrate among such remaining but incompatible detections. This will further reduce the false positive rate but at the expense of some missed detections. We shall not report those results; additional details can be found in Sahbi (2003). Our main intention is to illustrate the performance of the g-network on a real pattern recognition problem rather than to provide a detailed study of face detection or to optimize our error rates.

8.2 Evaluation

We evaluated the g-network in term of precision and run-time in several large scale experiments involving still images, video frames (TF1) and standard datasets of varying difficulty, including the CMU+MIT image set; some of these are extremely challenging. All our experiments were run under a 1-Ghz pentium-III mono-processor containing a 256 MB SDRM memory, which is today a standard machine in digital image processing.

The Receiver Operator Characteristic (ROC) curve is a standard evaluation mechanism in machine perception, generated by varying some free parameter (e.g., a threshold) in order to investigate the trade-off between false positives and false negatives. In our case, this parameter is the threshold τ for the aggregate SVM score of complete chains discussed in previous section. Several points on the ROC curve are given for the TF1 and CMU+MIT test sets whereas only a single point is reported for easy databases (such as FERET).

8.2.1 FERET AND TF1 DATASETS

The FERET database (FA and FB combined) contains $3,280$ images of single and frontal views of faces. It is not very difficult: The detection rate is 98.8 % with 245 false alarms and examples are shown in the top of Fig 10. The average run time on this set using a 1Ghz is 0.28 (s) for images of size 256×384.

The TF1 corpus involves a News-video stream of 50 minutes broadcasted by the French TV channel TF1 on May 5th, 2002. (It was used for a video segmentation and annotation project at INRIA and is not publicly available.) We sample the video at one frame each $4(s)$, resulting into 750 good quality images containing 1077 faces. Some results are shown on the bottom of Fig 10 and the performance is described in Table 8.2.1 for three points on the ROC curve. The *false alarm rate* is the total number of false detections divided by the total number of hierarchies traversed, that is, the total number of 16×16 blocks visited in processing the entire database.

8.2.2 ARF DATABASE AND SENSITIVITY ANALYSIS

The full ARF database contains 4000 images on ten DVDs; eight of these DVDs – $3,200$ images with faces of 100 individuals against uniform backgrounds – are publicly available at (http://rvl1.ecn.purdue.edu/$\sim$aleix/aleix_face_DB.html). This dataset is still very challenging due to large differences in expression and lighting, and especially to partial occlusions due to scarves, sunglasses, etc. The g-network was run on this set; sample results are given in the rows 2-4 of Fig 10. Among the 10 face images for a given person, two images show the person with sunglasses, three with scarves and five with some variation in the facial expression and/or strong lighting effects. Our face detection rate is only 78.79 % with 3 false alarms. Among the missed faces, 32.12 %

Figure 10: Sample detections on three databases: FERET (top), ARF (middle), TF1 (bottom).

are due to occlusion of the mouth, 56 % due to occlusion of the eyes (presence of sun glasses) and 11.88 % due to face shape variation and lighting effects.

8.2.3 CMU+MIT DATASET

The CMU subset contains frontal (upright and in-plane rotated) faces whereas the MIT subset contains lower quality face images. Images with an in-plane rotation of more than 20^0 were removed, as well as "half-profile" faces in which the nose covers one cheek. This results in a subset of 141 images from the CMU database and 23 images from the MIT test set. These 164 images contain 556 faces. A smaller subset was considered in Rowley et al. (1998) and in Viola and Jones (2001),

Threshold	# Missed faces	Detection rate	# False alarms	False alarm rate	Average run-time
$\tau = 0$	017	98.4 %	333	1/9,632	0.351(s)
$\tau = 1$	109	89.8 %	143	1/22,430	0.357(s)
$\tau = 2$	151	85.9 %	096	1/33,411	0.343(s)

Table 3: Performance on the TF1 database of 750 frames with 1077 faces. The three rows correspond to three choices of the threshold for clustering detections. The false alarm rate is given as the number of background pattern declared as faces over the total number of background patterns. The average run-time is reported for this corpus on images of size 500×409.

namely 130 images containing 507 faces, although in the former study other subsets, some accounting for half-profiles, were also considered (see Table 4).

	Sahbi & Geman	Viola and Jones (2001)	Rowley et al. (1998)
# of images	164	130	130
# of faces	556	507	507
False alarms	112	95	95
Detection rate	89.61 %	90.8 %	89.2 %
Time (384×288)	$0.20(s)$	$\frac{1}{3} \times 0.20(s)$	$5 \times 0.20(s)$

Table 4: Comparison of our work with other methods which achieve high performance.

The results are given in Table 4. The g-network achieves a detection rate of 89.61 % with 112 false alarms on the 164 images. These results are very comparable to those in Rowley et al. (1998); Viola and Jones (2001): for 95 false alarms, the detection rate in Viola and Jones (2001) was 90.8 % and in Rowley et al. (1998) it was 89.2 %. Put another way, we have an equivalent number of false alarms with a larger test set but a slightly smaller detection rate; see Table 4. Our performance could very likely be improved by utilizing a larger training set, exhibiting more variation than the Olivetti set, as in Viola and Jones (2001); Rowley et al. (1998); Schneiderman and Kanade (2000), where training sets of sizes 4916, 1048 and 991 images, respectively, are used.

Scenes are processed efficiently; see Fig 11. The run-time depends mainly on the size of the image and its complexity (number of faces, presence of face-like structures, texture, etc). Our system processes an image of 384×288 pixels (the dimensions reported in cited work) in 0.20(s); this is an average obtained by measuring the total run time on a sample of twenty images of varying sizes and computing the equivalent number of images (approximately 68) of size 384×288. This average is about three times slower than in Viola and Jones (2001), approximately five times faster than the fast version in Rowley et al. (1998) and 200 times faster than in Schneiderman and Kanade (2000). Notice that, for tilted faces, the fast version of Rowley's detector spends 14(s) on images of 320×240 pixels.

Figure 11: Detections using the CMU+MIT test set. More results can be found on http://www-rocq.inria.fr/who/Hichem.Sahbi/Web/face_results/

Threshold	Detection rate	# False alarms	False alarms rate
$\tau = 0$	92.95 %	312	1/2,157
$\tau = 0.5$	89.61 %	112	1/6,011
$\tau = 1$	87.2 %	096	1/7,013
$\tau = 10$	34.94 %	004	1/168,315

Table 5: Evaluation of our face detector on the CMU+MIT databases.

8.2.4 HALLUCINATIONS IN TEXTURE

Performance degrades somewhat on highly texture scenes. Some examples are provided in Fig 12. Many features are detected, triggering false positives. However, there does not appear to an "explosion" of hallucinated faces, at least not among the roughly 100 such scenes we processed, of which only a few had order ten detections (two of these are shown in Fig 12).

9. Summary

We presented a general method for exploring a space of hypotheses based on a coarse-to-fine hierarchy of SVM classifiers and applied it to the special case of detecting faces in cluttered images. As opposed to a single SVM dedicated to a template, or even a hierarchical platform for coarse-to-fine template-matching, but with no restrictions on the individual classifiers (the f-network), the proposed framework (the g-network) allows one to achieve a desired balance between computation and error. This is accomplished by controlling the number of support vectors for each SVM in the hierarchy; we used the reduced set technique here, but other methods could be envisioned. The design of the network is based on a model which accounts for cost, selectivity and invariance. Naturally, this requires assumptions about the cost of an SVM and the probability that any given SVM will be evaluated during the search.

We used one particular statistical model for the likelihood of a background pattern reaching a given node in the hierarchy, and one type of error constraint, but many others could be considered. In particular, the model we used is not realistic when the likelihood of an "object" hypothesis becomes comparable with that of the "background" hypothesis. This is in fact the case at deep levels of the hierarchy, at which point the conditional selectivity of the classifiers should ideally be calculated with respect to *both* object and background probabilities. A more theoretical approach to these issues, especially the cost/selectivity/invariance tradeoff, can be found in Blanchard and Geman (2005), including conditions under which coarse-to-fine search is optimal.

Extensive experiments on face detection demonstrate the huge gain in efficiency relative to either a dedicated SVM or an unrestricted hierarchy of SVMs, while at the same time maintaining precision. Efficiency is due to both the coarse-to-fine nature of scene processing, rejecting most background regions very quickly with highly invariant SVMs, *and* to the relatively low cost of most of the SVMs which are ever evaluated.

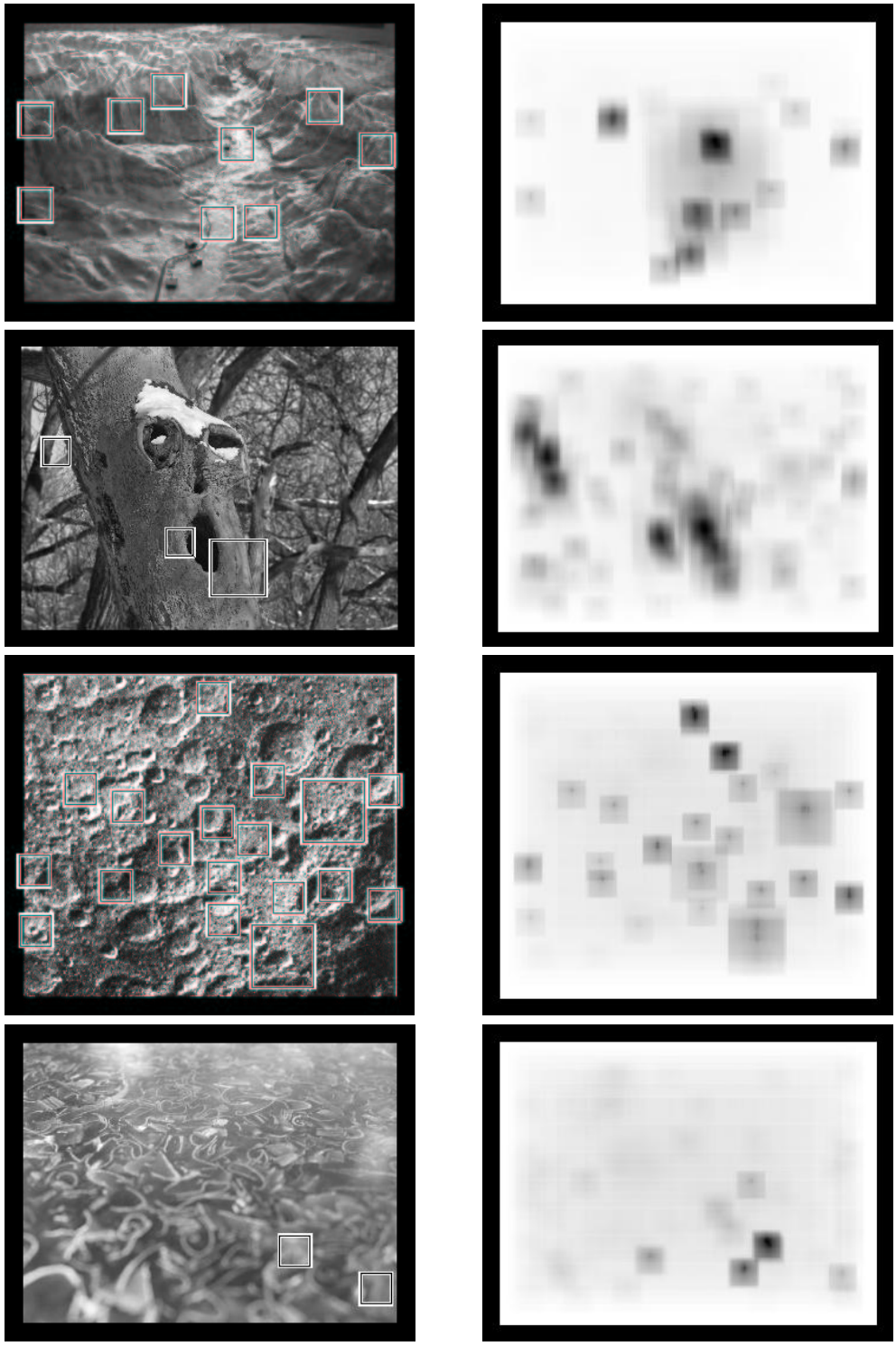


Figure 12: Left: Detections on highly textured scenes. (We thank Larry Jackal for the second ("face in a tree") image.) Right: The darkness of a pixel is proportional to the amount of local processing necessary to collect all detections. The average number of kernel evaluations, per block visited, are respectively 11, 6, 14 and 4.

Acknowledgments

The authors are grateful to both referees for very thoughtful and comprehensive reviews, which represent the reviewing process at its best. The research of the second author was supported in part by NSF grants ITR 0219016 and ITR 0427223 and by ARO grant DAAD19-02-1-0337.

Appendix A.

In this appendix, we will show how an approximate solution of the constrained minimization problem (8) can be obtained for the case of a binary hierarchy (i.e., $\nu_l = 2^{l-1}$). An extension to any arbitrary hierarchy can be found in Sahbi (2003).

Suppose β is fixed and consider the optimization problem in (8). Clearly, the *unconstrained* problem is degenerate, minimized by choosing $n_l \equiv 0$; indeed this minimizes cost. We start by minimizing cost for a *fixed value* of n_L and for real-valued $n_l, l = 1,...,n_{L-1}$. In this case, the values of $n_1, n_2, ..., n_{L-1}$ which satisfy $\frac{\partial \mathcal{C}}{\partial n_l} = 0$, $l = 1,...,L-1$, are given by:

$$n_l = \begin{cases} \left(2^{\frac{L(L-1)}{2}} \left(\prod_{i=1}^{L-1} \beta_i \right)^{-1} n_L \right)^{1/L} & \text{if } l = 1 \\ 2^{-\frac{l(l-1)}{2}} \left(\prod_{i=1}^{l-1} \beta_i \right) \beta_l^{-1} \, n_1^{l-1} \, (\beta_l n_1 - 2^{l-1}) & l \in \{2,...,L-1\} \\ n_L & l = L. \end{cases} \tag{11}$$

The proof is straightforward:

$$\begin{aligned} \frac{\partial \mathcal{C}}{\partial n_1} &= 1 - \sum_{l=2}^{L} \left[\frac{\beta_1 2^{l-1} n_l}{(\sum_{i=1}^{l-1} \beta_i n_i)^2} \right] \\ \frac{\partial \mathcal{C}}{\partial n_j} &= \frac{2^{j-1}}{(\sum_{i=1}^{j-1} \beta_i n_i)} - \sum_{l=j+1}^{L} \left[\frac{\beta_j 2^{l-1} n_l}{(\sum_{i=1}^{l-1} \beta_i n_i)^2} \right], j \in \{2, L-1\}. \end{aligned} \tag{12}$$

We have:

$$\frac{\partial \mathcal{C}}{\partial n_{j+1}} = 0 \quad \Rightarrow \quad \sum_{l=j+2}^{L} \left[\frac{2^{l-1} n_l}{(\sum_{i=1}^{l-1} \beta_i n_i)^2} \right] = \frac{2^j}{(\sum_{i=1}^{j} \beta_i n_i)} \frac{1}{\beta_{j+1}}$$

$$\frac{\partial \mathcal{C}}{\partial n_j} = 0 \quad \Rightarrow \quad \frac{2^{j-1}}{(\sum_{i=1}^{j-1} \beta_i n_i)} - \beta_j \frac{2^j n_{j+1}}{(\sum_{i=1}^{j} \beta_i n_i)^2} - \beta_j \sum_{l=j+2}^{L} \left[\frac{2^{l-1} n_l}{(\sum_{i=1}^{l-1} \beta_i n_i)^2} \right] = 0.$$

The above two equations imply:

$$\frac{2^{j-1}}{(\sum_{i=1}^{j-1}\beta_i n_i)} - \beta_j \frac{2^j\, n_{j+1}}{(\sum_{i=1}^{j}\beta_i n_i)^2} - \beta_j \frac{2^j}{\beta_{j+1}\,(\sum_{i=1}^{j}\beta_i n_i)} = 0,\ j \neq 1. \tag{13}$$

Suppose n_1 is known; we show by a recursion that the general term n_l is given by (11): Combining $\frac{\partial \mathcal{C}}{\partial n_1} = 0$ and $\frac{\partial \mathcal{C}}{\partial n_2} = 0$, we obtain:

$$1 - \frac{\beta_1\, 2\, n_2}{\beta_1^2\, n_1^2} - \frac{2\,\beta_1}{\beta_2\,\beta_1\, n_1} = 0 \quad \Rightarrow \quad n_2 = \frac{1}{2}\,\frac{\beta_1}{\beta_2}\, n_1\,(\beta_2 n_1 - 2).$$

Assume for $2 \leq j \leq l (l \in \{2, L-2\})$

$$n_j = 2^{-\frac{j(j-1)}{2}} \left(\prod_{i=1}^{j-1}\beta_i\right) \beta_j^{-1}\, n_1^{j-1}\,(\beta_j n_1 - 2^{j-1}). \tag{14}$$

We now demonstrate that:

$$n_{l+1} = 2^{-\frac{(l+1)l}{2}} \left(\prod_{i=1}^{l}\beta_i\right) \beta_{l+1}^{-1}\, n_1^{l}\,(\beta_{l+1} n_1 - 2^{l}). \tag{15}$$

By (14), $\forall j \in \{2, ..., l\}$, we have:

$$\begin{aligned}\left(\sum_{i=1}^{j}\beta_i n_i\right) &= \beta_1 n_1 + \beta_2\,\frac{1}{2}\,\beta_1\beta_2^{-1}\, n_1\,(\beta_2 n_1 - 2) + \beta_3\,\frac{1}{8}\,\beta_1\beta_2\beta_3^{-1}\, n_1^2\,(\beta_3 n_1 - 4) + ... \\ &+ \beta_{j-1}\left(2^{-\frac{(j-1)(j-2)}{2}}\right)\left(\prod_{i=1}^{j-2}\beta_i\right)\beta_{j-1}^{-1}\, n_1^{j-2}\,\left(\beta_{j-1} n_1 - 2^{j-2}\right) \\ &+ \beta_j\left(2^{-\frac{j(j-1)}{2}}\right)\left(\prod_{i=1}^{j-1}\beta_i\right)\beta_j^{-1}\, n_1^{j-1}\left(\beta_j n_1 - 2^{j-1}\right).\end{aligned}$$

Hence, $\forall j \in \{2, ..., l\}$:

$$\left(\sum_{i=1}^{j}\beta_i n_i\right) = 2^{-\frac{j(j-1)}{2}} \left(\prod_{i=1}^{j}\beta_i\right) n_1^j. \tag{16}$$

Let $\pi_j = \prod_{i=1}^{j}\beta_i$. Using (16) and for $j = l$, we rewrite (13) as:

$$\frac{2^{l-1}}{2^{-\frac{(l-1)(l-2)}{2}}\,\pi_{l-1}\, n_1^{l-1}} - \beta_l \frac{2^l n_{l+1}}{\left(2^{-\frac{l(l-1)}{2}}\,\pi_l\, n_1^l\right)^2} - \frac{\beta_l}{\beta_{l+1}}\,\frac{2^l}{2^{-\frac{l(l-1)}{2}}\,\pi_l\, n_1^l} = 0$$

$$\Rightarrow \quad n_{l+1} = 2^{-\frac{(l)(l+1)}{2}}\,\pi_l\,\beta_{l+1}^{-1}\, n_1^l\,(\beta_{l+1} n_1 - 2^l)$$

which proves (15). As for n_1, using (12) for $j = L-1$,

$$\frac{\partial \mathcal{C}}{\partial n_{L-1}} = 0 \quad \Rightarrow \quad n_1 = \left(\left(\frac{1}{\pi_{L-1}} \right) 2^{L(L-1)/2} \, n_L \right)^{1/L} \quad \square$$

We can now rewrite (8) as:

$$\min_{n_L} L \left(\frac{1}{\pi_{L-1}} 2^{L(L-1)/2} \, n_L \right)^{1/L} - \sum_{l=2}^{L} \left(\frac{2^{l-1}}{\beta_l} \right)$$
$$\text{s.t.} \quad \begin{cases} \nu_L \left(\sum_{i=1}^{L} \beta_i n_i \right)^{-1} \leq \mu \\ 0 < n_l \leq N_l. \end{cases}$$

Using (11), this can be written entirely in terms of n_L. We use a "generate-and-test" (brute-force search) strategy: First, the parameter n_L is varied from 1 to its upper bound N_L (with some quantization). Then, for each value of this parameter, we check the consistency of the candidate solution, that is, whether the first constraint (on expected false alarms) is satisfied and whether each n_l is bounded N_l. The value n_L minimizing the cost function is retained.

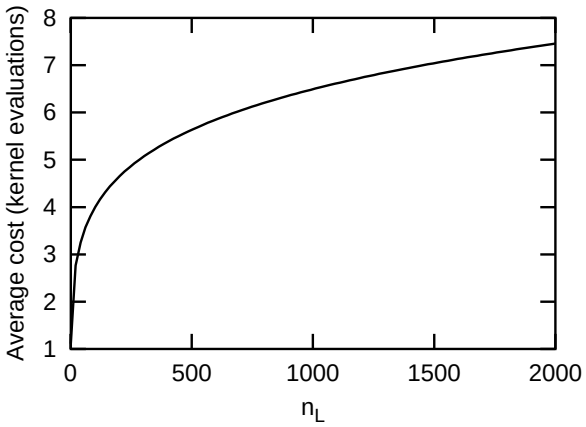


Figure 13: The average cost $\mathcal{C}(n_1, ..., n_L)$ is an increasing function of n_L.

For small values of n_L, the objective function in (8) (the average cost) typically takes small values (cf. Fig 13) and the upper bound constraints related to $\{n_l\}$ are generally satisfied, but the mean false alarm constraint might not be satisfied. For large values of n_L, the bounds on $\{n_l\}$ might not be satisfied and the average cost increases, although the mean false alarm constraint is typically satisfied.

Finally, we allow β to vary with Ψ according to (9). Notice that (8) might not have a solution for any Ψ; obviously we only consider values for which the constraints are satisfied for some n_L.

References

Y. Amit and D. Geman. A computational model for visual selection. *Neural Computation.*, 11(7): 1691–1715, 1999.

Y. Amit, D. Geman, and X. Fan. A coarse-to-fine strategy for multi-class shape detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(12):1606–1621, 2004.

S. Baker and S. Nayar. Pattern rejection. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 544–549, 1996.

R. Battiti and C. Koch. Computing optical flow across multiple scales: a coarse-to-fine approach. *International Journal of Computer Vision*, 6(2):133–145, 1991.

G. Blanchard and D. Geman. Sequential testing designs for pattern recognition. *Annals of Statistics*, 33:1155–1202, 2005.

G. Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10:849–865, 1988.

B. E. Boser, I. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Workshop on Computational Learning Theory.*, pages 144–152, 1992.

C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector machines. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Proceedings of the Advances in Neural Information Processing Systems*, volume 9, pages 375–381. The MIT Press, 1997.

C.J.C. Burges. Simplified support vector decision rules. In *Proceedings of the International Conference on Machine Learning*, pages 71–77, 1996.

T. Cootes, K. Walker, and C. Taylor. View-based active appearance models. In *Proceedings of the IEEE International Conference on Face and Gesture Recognition.*, pages 227–232, 2000.

M. Elad, Y. Hel-Or, and R. Keshet. Pattern detection using a maximal rejection classifier. *Pattern Recognition Letters*, 23(12):1459–1471, 2002.

C.K. Eveland, D.A. Socolinsky, C.E. Priebe, and D.J. Marchette. A hierarchical methodology for class detection problems with skewed priors. *Journal of Classification*, 22:17–48, 2005.

T. Evgeniou, M. Pontil, C. Papageorgiou, and T. Poggio. Image representations for object detection using kernel classifiers. In *Proceedings of the Asian Conference on Computer Vision*, pages 687–692, 2000.

X. Fan. *Learning a hierarchy of classifiers for multi-class shape detection*. PhD thesis, Johns Hopkins University, Department of Electrical Engineering, 2006.

R. Féraud, O.J. Bernier, J.E. Viallet, and M. Collobert. A fast and accurate face detector based on neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1):42–53, 2001.

F. Fleuret. *Détection hiérarchique de visages par apprentissage statistique*. PhD thesis, University of Paris VI, Jussieu., 1999.

F. Fleuret and D. Geman. Coarse-to-fine face detection. *International Journal of Computer Vision*, 41(2):85–107, 2001.

S. Gangaputra and D. Geman. A design principle for coarse-to-fine classification. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1877–1884, 2006a.

S. Gangaputra and D. Geman. The trace model for object detection and tracking. In *Proceedings of Workshop on Object Recognition, Taromina, Sicily, 2004, Lecture Notes in Computer Science*, 2006b.

C. Garcia and M. Delakis. Convolutional face finder: A neural architecture for fast and robust face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1408–1423, 2004.

J. C. Gee and D. R. Haynor. Rapid coarse-to-fine matching using scale-specific priors. In *Proceedings of the SPIE Medical Imaging: Image Processing, M. H. Loew and K. M. Hanson, eds., Bellingham, WA:SPIE*, pages 416–427, 1996.

S. Geman, K. Manbeck, and D.E. McClure. Coarse-to-fine search and rank-sum statistics in object recognition. Technical report, Brown University, 1995.

B. Heisele, T. Serre, S. Mukherjee, and T. Poggio. Feature reduction and hierarchy of classifiers for fast object detection in video images. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 18–24, 2001.

R.L. Hsu, M. Abdel-Mottaleb, and A. K. Jain. Face detection in color images. In *Proceedings of the IEEE International Conference on Image Processing*, pages 1046–1049, 2001.

D. P. Huttenlocher and W.J. Rucklidge. A multi-resolution technique for comparing images using the hausdorff distance. In *Proceedings of the Conference on Computer Vision and Pattern Recogntion*, pages 705–706, 1993.

F. Jung. *Reconnaissance d'objets par focalisation et détection de changements*. PhD thesis, Ecole Polytechnique, Paris, France, 2001.

T. Kanade. *Picture processing system by computer complex and recognition of human faces*. PhD thesis, Kyoto University, Department of Information Science, 1977.

D. Keren, M. Osadchy, and C. Gotsman. Antifaces: A novel, fast method for image detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(7):747–761, 2001.

W. Kienzle, G.H. Bakir, M. Franz, and B. Schölkopf. Face detection - efficient and rank deficient. In *Proceedings of the Eighteenth Annual Conference on Neural Information Processing Systems*, 2004.

T. Leung, M.C. Burl, and P Perona. Finding faces in cluttered scenes using random labelled graph matching. In *Proceedings of the International Conference on Computer Vision*, pages 637–644, 1995.

S.Z. Li and Z. Zhang. Floatboost learning and statistical face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1112–1123, 2004.

J. Miao, B. Yin, K. Wang, L. Shen, and X. Chen. A hierarchical multiscale and multiangle system for human face detection in complex background using gravity center template. *Pattern Recognition*, 32(7):1237–1248, 1999.

E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 130–136, 1997.

S. Romdhani, P. Torr, B. Schölkopf, and A. Blake. Computationally efficient face detection. In *Proceedings of the International Conference on Computer Vision*, pages 695–700, 2001.

H. Rowley. *Neural network-based face detection*. PhD thesis, Carnegie Mellon University, 1999.

H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.

H. Sahbi. *Coarse-to-fine support vector machines for hierarchical face detection*. PhD thesis, TU-798, Versailles University, 2003.

H. Sahbi and N. Boujemaa. Coarse-to-fine skin and face detection. In *Proceedings of the ACM International Conference on Multimedia.*, pages 432–434, 2000.

H. Sahbi, D. Geman, and N. Boujemaa. Face detection using coarse-to-fine support vector classifiers. In *Proceedings of the IEEE International Conference on Image Processing.*, pages 925–928, 2002.

H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pages 746–752, 2000.

B. Schölkopf, P. Knirsch, A. Smola, and C. Burges. Fast approximation of support vector kernel expansions and an interpretation of clustering as approximation in feature spaces. In *Proceedings of the Levi M. Schanz R.-J. Ahlers and F. May editors, Mustererkennung DAGM-Symposium Informatik aktuell*, pages 124–132, 1998.

J. Sobottka and I. Pittas. Segmentation and tracking of faces in color images. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition.*, pages 236–241, 1996.

D.A. Socolinsky, Neuheisel, C.E. Priebe, D.J. Marchette, and J. DeVinney. A boosted cccd classifier for fast face detection. *Computing Science and Statistics*, 35, 2003.

K-K. Sung. *Learning and example selection for object and pattern detection,*. PhD thesis, Massachusetts Institute of Technology, Electrical Engineering and Computer Science, 1996.

V.N. Vapnik. Statistical learning theory. *A Wiley-Interscience Publication*, 1998.

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 511–518, 2001.

J. Wu, M.D. Mullin, and J.M. Rehg. Linear asymmetric classifier for cascade detectors. In *Proceedings of the International Conference on Machine Learning*, pages 988–995, 2005.

M.H. Yang, D. Roth, and N. Ahuja. A snow-based face detector. In *Proceedings of Advances in Neural Information Processing Systems 12*, pages 855–861, 2000.