1

Coarse-to-Fine Classification and Scene Labeling

1.1 Introduction

The semantic interpretation of natural scenes, so effortless for humans, is perhaps the main challenge of artificial vision, having largely resisted any satisfying solution, at least in searching for multiple objects in real, cluttered scenes with arbitrary illumination. This problem is the motivation for the work in this paper. Specifically, the models and algorithms presented here result from making computational efficiency the organizing principle for vision, a proposal recently explored in both theory (Jung 2001, Blanchard & Geman 2001) and practice (Amit & Geman 1999, Fleuret 2000, Fleuret & Geman 2001); see also Lambdan et al. 1988 and Geman et al. 1995 for related examples of efficient visual search. "Theory" refers to analyzing the efficiency of coarse-to-fine (CTF) search under various statistical models and cost structures; summarized here are the general mathematical framework, including an abstract formulation of CTF classification based on multiresolution "tests," and some results about optimal testing strategies. "Practice" refers to designing computer algorithms for detecting objects in natural scenes; several such experiments on face detection are included together with a brief description of how the tests are realized as image functionals.

We model scene interpretation as a dynamic and adaptive process, generating a sequence of increasingly precise interpretations. At the beginning the labels are crude and too plentiful; there are confusions among objects of interest and between objects and clutter. Eventually, the labels become more precise, for instance object categories and presentations are refined, and confusions are removed. Certain fundamental tradeoffs then evolve - between invariance and discrimination, and between false positive error and computation. Similar themes are explored in Riesenhuber and Poggio 1999 for visual processing in cortex.

For practical convenience, we separate the whole process of scene classification into two rough phases: non-contextual and contextual. (This distinction was previously explored in Amit & Geman 1999.) Noncontextual classification, or simply *detection*, comes first. The goal is to infer from the image data instances of highly visible objects of interest under the constraint that no objects be overlooked (no "missed detections"), but allowing for a limited number of false positives. *Detection is the focus in this paper.*

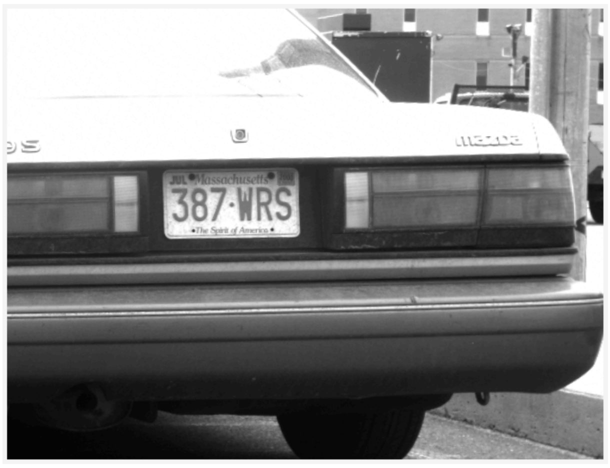

Figure 1.1.

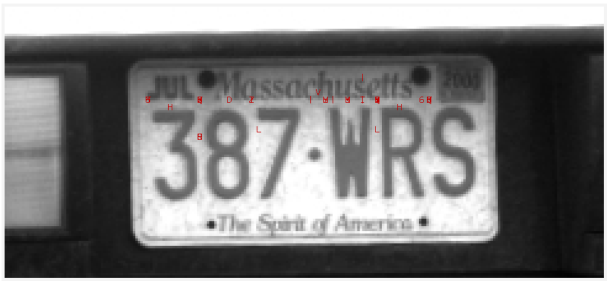

Figure 1.2. Characters detected during detection.

As conceived here, it is based on sparse representations and performed with very simple operations - essentially just counting local features. The result is a list of “classes” of objects and their “presentations” in the scene. The desired level of detail is application-dependent; for example, the class might be “face” rather than a specific individual and the presentation might be no more specific than a range of values for certain pose parameters (e.g., position, scale and orientation). Other aspects of the presentation might be of interest, such as the font of a character or the gender of a face.

An example of multifont optical character recognition is shown in Figure 1.1 where the objective is to identify the main symbols on the license plate. The detection phase is illustrated in Figure 1.2. For each of the six characters, there are multiple detections; some *but not all* of the detections “near” each symbol are erroneous, due to clutter or confusions. There are also false alarms away from the symbols of interest. This is ongoing work with Yali Amit and will be described elsewhere.

Contextual classification, not treated here, involves more intensive computation in the vicinity of detections in order to determine which of these are in fact objects of interest and to disambiguate among confusions, such as D's, 0's and O's detected at roughly the same location. The underlying process is again coarse-to-fine. Moreover, ultimately there is no way to avoid a fully contextual analysis in order to discover partially visible objects and other complex spatial arrangements. Processing which accounts for context and relationships is likely to require dense representations and be computationally intensive (e.g., involve online functional optimization). One proposal is "compositional vision" (cf. Geman et al 2001).

If scene labeling is driven by computational efficiency, a natural and effective mechanism is CTF classification. It is certainly *one way* of gaining (online) efficiency and I would argue that any other way ultimately boils down to something similar. CTF classification depends on a CTF *representation* for the family of interpretations under investigation. In other words, the representation of objects and presentations must be structured to accommodate coarse-to-fine search. Thus, the events of interest must be characterized by "attributes" at many levels of resolution. CTF search then means investigating those attributes in a particular order, namely from coarse ones to fine ones. This is the way we play *Twenty Questions.*

We develop an abstract formulation of CTF classification. Roughly speaking, we consider a series of nested (hence increasingly fine) partitions of the set of possible explanations, and we define a binary "test" for each cell Λ of each partition. The test X_Λ associated with Λ must always respond "yes" to interpretations in Λ. The tests also have varying levels of "cost" and "discrimination" (statistical power); both increase as cell size decreases, and hence there is a tradeoff with invariance. The "detector" $\hat{Y}$ is a (set-valued) function of these tests; it consists of all interpretations which are confirmed at all levels of resolution, and is the primary object of our mathematical analysis. More specifically, we ask: *Which sequential (test by test) adaptive evaluation of $\hat{Y}$ minimizes average computation?* The answer is that under wide-ranging assumptions on the statistical distribution of the tests and how cost is measured, and among all testing strategies based on performing tests one at a time until a decision is reached (i.e., $\hat{Y}$ is determined), CTF questioning minimizes the *mean* of the sum of the costs of the all the tests which are utilized.

Further remarks about invariance and discrimination, and about parallel vs. serial processing, follow in §2. The abstract formulation is given in §3, where the statistical framework is laid out, including the definitions of cost, invariance and discrimination, and the definition of $\hat{Y}$. In §4 we introduce the family of possible evaluations of the detector and a model for measuring the computational efficiency of each candidate. Several results on optimal strategies are mentioned in §5 without proof and the error rates of the detector are specificed in §6. In §7, we return to the scene interpretation problem and put everything in concrete terms, including how $\hat{Y}$ is constructed from image intensity data. Finally, some experiments on face detection and concluding remarks appear in §8.

1.2 Invariance vs. Discrimination

The rationale for CTF search is intuitive and transparent. Start with properties of objects and presentations which are simple and common, almost regardless of discriminating power; in other words, look for tests which *invariably* accept as many object/pose pairings as possible, even if many instances of clutter and non-targeted objects are found as well. Rejecting even a small percentage of background instances with cheap and universal tests is efficient. Then proceed to more discriminating properties, albeit more complex and specialized; whereas a greater number of tests must be designed or learned in order to "cover" all objects and poses, only relatively few of them will be needed during any given search due to pruning by coarser tests. Also the still significant false alarm rate is compensated by invariance (no missed detections) and low cost termination of the search. Finally, reserve computationally intensive, highly discriminating filters (basically, object-specific and pose-specific "template-matching") for the very end - for those inevitable and diabolical arrangements of clutter which "look" like objects in the eyes of the features.

This program amounts to creating "invariants" at many levels of power. But these are *not* the geometric and algebraic types sought after in continuum, shape-based approaches to object recognition. The invariants here are based on generic local features, not special points on curves, etc. And our requirements are more modest: Find binary image functionals which always respond positively for a given set of shapes but may respond positively to other shapes and image structures. It is only at the level of low invariance (specific poses) that we demand high discrimination. Consequently, during the course of processing there is then a steady progression from high invariance to low invariance and from low discrimination to high discrimination.

The image functionals we consider in §7 are of the form

$$X = \begin{cases} 1 & \text{if } \sum_{l \in L} \xi_l \geq t \\ 0 & \text{otherwise} \end{cases}$$

where each ξ_l is a local binary feature which signals an "edge" is present "near" location z_l and with orientation ϕ_l; L is a distinguished set of edges dedicated to a set of poses and t is an appropriate threshold. (How "near" depends on the desired level of invariance; see §7.) Thus, evaluating X consists of checking for at least t edges among a special ensemble which characterizes a particular set of shapes - certain types of objects at certain geometric poses. The complexity of such as test might simply be $|L|$. High discrimination and high complexity corresponds to "template-matching" and the set L might then provide a rather dense representation of the shapes. However, for such elementary tests, achieving high invariance (covering many poses) *and* high discrimination at the same time is likely to be impossible, regardless of cost.

It is clearly impossible to find common but localized attributes of two object presentations with significantly different (geometric) poses, say far apart in the scene. As a result, we use a simple, "divide-and-conquer" strategy based on object location. (Every object is assumed to have a visible, distinguished point.) One "base" detector, $\hat{Y}$, is designed to find all instances of objects with presentations in a "reference" cell, for example locations confined to a $k \times k$ block and scale confined to a $[\sigma_{min}, 2\sigma_{min}]$ where σ_{min} is the smallest scale entertained. The scene is partitioned into non-overlapping $k \times k$ blocks, and the detector $\hat{Y}$ is applied to the image data $I(z), z \in W$ in a window W centered at each such block; the dimension of W is sufficiently large to capture all objects at the given locations and scales. Objects at scales larger than σ_{min} are detected by repeatedly downsampling and parsing the scene in the same way.

In principle, the detector could be applied to each window simultaneously; this is the parallel component of the algorithm. The serial component - the CTF implementation of $\hat{Y}$ is each window - is the heart of the algorithm and the real source of efficient computation.

1.3 CTF Classification: Abstract Formulation

Let $\mathcal{S} = \{\lambda_1, \lambda_2, ...\}$ denote a set of *states* or *interpretations.* Each subset $\Lambda \subset \mathcal{S}$ will be called an *index*. In addition, fix a probability space $(\mathcal{I}, P)$ and suppose there is a *true index* $Y(I)$ for each $I \in \mathcal{I}$. Although we allow more than one true interpretation, we are primarily interested in the case in which either $Y = \{\lambda\}$ or $Y = \emptyset$.

In the application to detecting objects, λ is a pair (c, θ) where c is the "class" of an object and θ stands for the "presentation." Even one object class is challenging and frequently considered in computer vision. $\mathcal{I}$ is then the set of subimages $I = \{I(z), z \in W\}$ and P could be taken as an empirical measure; we shall be more specific about this later on. Finally, $Y(I)$ is the list of the objects and presentations appearing in I. In general, there is at most one object which is both visible and centered in a given W.

An important feature of the detection problem, and one that motivates an upcoming approximation of P, is that $P(Y = \emptyset) \gg P(Y = \Lambda)$ for any given Λ. We might even assume that $P(Y = \emptyset) \gg P(Y \neq \emptyset)$, so that the most likeliy interpretation is that there are no states "present" in I. Write P_λ for $P(.|\lambda \in Y)$ and P_0 for $P(.|Y = \emptyset)$.

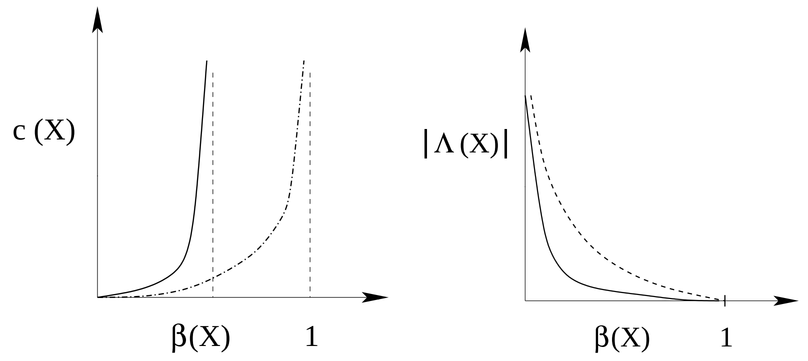

Figure 1.3. Left: The cost vs. discrimination tradeoff at two levels of invariance, "high" (solid line) and "low" (dashed line). Right: The invariance vs. discrimination tradeoff at two levels of cost, "low" (solid line) and "high" (dashed line).

Shortly we shall define a *detector* $\hat{Y}$ based on a family of functions $X : \mathcal{I} \longmapsto \{0,1\}$ called *tests*. The basic constraint on $\hat{Y}$ is zero false negative error:

$$P(Y \subset \hat{Y}) = 1 \tag{1.1}$$

Equivalently,

$$P_\lambda(\lambda \in \hat{Y}) = 1, \quad \forall \lambda \in \mathcal{S}.$$

Assume each test has a *cost* or *complexity* $c(X)$ which represents the amount of online computation (or time) necessary to evaluate X and of course depends on how X is constructed. The *invariant set* for X is $\Lambda(X) = \{\lambda : P_\lambda(X = 1) = 1\}$. Finally, the *discrimination* or *power* of X is defined as $\beta(X) = P_0(X = 0)$. The tradeoff between cost and discrimination at different levels of invariance is shown in the lefthand panel of Figure 1.3; the righthand panel shows the tradeoff between invariance and discrimination at different costs.

Suppose we are given a family of tests $\mathbf{X} = \{X_\Lambda, \Lambda \in \mathbf{\Lambda}\}$ where the notation X_Λ means that $\Lambda = \Lambda(X)$. The reason for indexing the tests by their invariant sets is that we will build tests to a set of specifications. Basically, we *first* design a hierarchy of subsets of $\mathcal{S}$ and *then*, for each Λ in the hierarchy, we build a test X_Λ which is invariant with respect to the classes and poses in Λ.

Now define $\hat{Y}(I) \subset \mathcal{S}, I \in \mathcal{I}$, by

$$\hat{Y}(I) = \hat{Y}(\mathbf{X}(I)) = \{\lambda : X_\Lambda(I) = 1 \ \forall \lambda \in \Lambda\}. \tag{1.2}$$

The rationale is that we accept a state λ as part of our interpretation if and only if this state it is "verified" at all levels of resolution, in the sense that each test X which "covers" λ (meaning $\lambda \in \Lambda(X)$) responds positively.

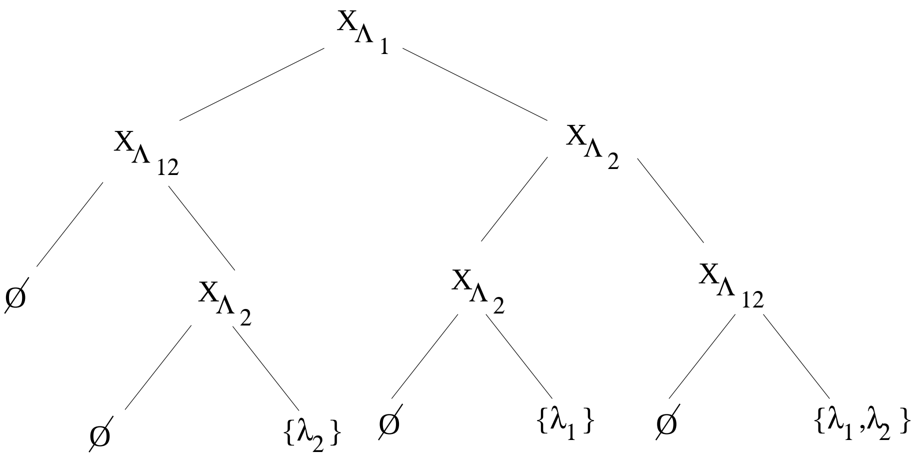

Figure 1.4. An example of a tree-structured evaluation of the detector $\hat{Y}$

1.4 Computation

Consider now adaptive (sequential) evaluations of $\hat{Y}$, i.e., tree-structured representations of $\hat{Y}$. Let $\mathcal{T}$ be the family of such evaluations. Each internal node of $T \in \mathcal{T}$ is labeled by a test X_Λ and each external node is labeled by an index - a subset of states. Our goal is to find the T which minimizes the mean cost of determining $\hat{Y}$ under assumptions on how $c(X)$ varies with $\beta(X)$, and how $\mathbf{X}$ is distributed under the "background model" P_0.

To illustrate such a computational procedure, take a simple example with $\mathcal{S} = \{\lambda_1, \lambda_2\}$ and three tests corresponding to $\Lambda_{1,2} = \{\lambda_1, \lambda_2\}$, $\Lambda_1 = \{\lambda_1\}$, $\Lambda_2 = \{\lambda_2\}$. One evaluation of $\hat{Y}$ is shown in Figure 1.4 where branching left means $X = 0$ and branching right means $X = 1$.

Notice that $\hat{Y} = \emptyset$ if and only if there is a *null covering* of $\mathbf{\Lambda}$: a subfamily $\{\Lambda_i\}$ such that $\bigcup_i \Lambda_i = \mathcal{S}$ and and $X_{\Lambda_i} = 0$ for each i. In Figure 1.4, one null covering corresponds to $\{X_{\Lambda_1} = 0, X_{\Lambda_2} = 0\}$ and one to $\{X_{\Lambda_{1,2}} = 0\}$.

1.4.1 Mean Cost

The *cost* of T is defined as

$$C(T) = \sum_{r \in T^\circ} \mathbf{1}_{H_r} c(X_r)$$

where T° is the set of internal nodes of T, X_r is the test at node r and H_r is the history of node r - the sequence of test results leading to r. (Equivalently, $C(T)$ is the aggregated cost of reaching the (unique) terminal node in T determined by $\mathbf{X}$.) Notice that $C(T)$ is a random variable. The *mean cost* $EC(T)$ is then

$$EC(T) = \sum_{r \in T^\circ} c(X_r)P(H_r) \tag{1.3}$$

$$= \sum_X c(X)P(X \; performed \; in \; T) \tag{1.4}$$

The second expression (1.4) is useful in proving the results mentioned in the following section.

Our optimization problem is then

$$\min_{T \in \mathcal{T}} EC(T) \tag{1.5}$$

In other words, find the best testing strategy. As it turns out, the best strategies are often far more efficient than T's constructed with top-down, greedy procedures, such as those utilized in machine learning for building decision trees; see Geman & Jedynak 2001 for comparisons.

1.4.2 Hierarchical Tests

In order to rank computational strategies we must impose some structure on both $\mathbf{\Lambda}$ and the law of $\mathbf{X}$. From here on we consider the case of *nested binary partitions* of $\mathcal{S}$:

$$\mathbf{\Lambda} = \{\Lambda_{m,j}, \; j = 1, ..., 2^m, \; m = 0, 1, ..., M\}.$$

Thus, $\Lambda_{0,1} = \Lambda_{1,1} \cup \Lambda_{1,2}$, $\Lambda_{1,1} = \Lambda_{2,1} \cup \Lambda_{2,2}$, etc. In §7, in experiments with a single object class, the hierarchy $\mathbf{\Lambda}$ is a "pose decomposition" wherein "cells" at level $m+1$ represent more constrained poses than cells at level m. As m increases, the level of invariance decreases and, in the models below, the level of discrimination increases. *This tree-structured hierarchy should not be confused with a tree-structured evaluation T of $\hat{Y}$.* The label $\hat{Y}$ at a terminal node of T depends on $\mathbf{X}$ and consists of all states in *any* level M cell of Λ for which there is a "1-chain" back to $\Lambda_{0,1}$, i.e., $X_{\Lambda'} = 1$ for every $\Lambda' \supset \Lambda$.

As a simple illustration, consider the case $M = 2$, in which case there are exactly seven sets in the hierarchy. Notice that the most refined cells $\Lambda_{2,j}, j = 1, 2, 3, 4$, may each contain numerous states λ, i.e., may provide an interpretation at a level of resolution which is still rather "coarse." Figure 1.5 shows $\hat{Y}(\mathbf{X})$ for two realizations of $\mathbf{X}$. For the same hierarchy and realizations of $\mathbf{X}$, the lefthand panel of Figure 1.6 illustrates a "depth-first" CTF evaluation of $\hat{Y}$ and the righthand panel a "breadth-first" CTF evaluation.

Another way to interpret $\hat{Y}$ is the following: For each level m in the hierarchy define

$$\hat{Y}_m(\mathbf{X}) = \bigcup_{j \in J_m(\mathbf{X})} \Lambda_{m,j}$$

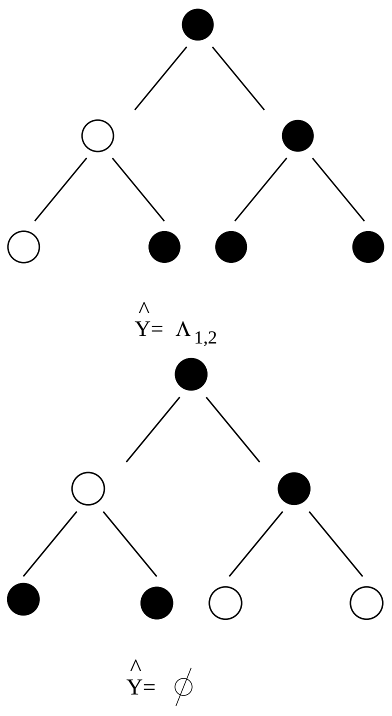

Figure 1.5. Examples of $\hat{Y}$ for two realizations of the tests in a nested hierarchy with three levels. Dark circles represent $X = 1$ and open circles represent $X = 0$. Top: There are two “chains of ones”. Bottom: There is a null covering.

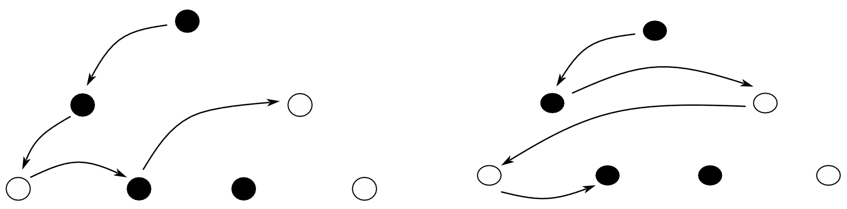

Figure 1.6. As in Figure 1.5, dark and light circles indicate positive and negative tests for a three level hierarchy. Left: A “depth-first” coarse-to-fine search. Right: A “breadth-first” coarse-to-fine search.

where $J_m = \{1 \leq j \leq 2^m : X_{m,j} = 1\}$. The detector $\hat{Y}$ is $\cap_m \hat{Y}_m$. We can think of $\hat{Y}_m$ as an estimate of Y at "resolution" m. Necessarily, $Y \subset \hat{Y}_m$ for each m, although in general it needn't happen that $\hat{Y}_{m+1} \subset \hat{Y}_m$ (or vice-versa).

1.4.3 An Approximation

We are going to assume that the mean cost is computed with respect to the background distribution P_0. This is motivated by the following argument. Recall that labeling the entire scene means applying the detector $\hat{Y}$ to subimages I centered on a sparse sublattice of the original scene. Whereas the likelihood of having some objects in the entire scene may be large, the *a priori* probability of the "null hypothesis" $Y = \emptyset$ is approximately one when evaluating $\hat{Y}$ for an arbitrary subimage at the scale of the objects. Therefore, the average amount of computation involved in executing our detection algorithm with a particular strategy T can be approximated by taking the expectation of $C(T)$ under P_0. Of course this approximation degrades along branches with many positive test responses, especially to discriminating tests associated with "small" class/pose cells, in which case the probability of an object being present may no longer be negligible. If one were to measure the mean computation under an appropriate (mixture) distribution, the conditional distribution of the tests under the various object hypotheses would come into play. Nonetheless, we continue to compute the likelihood of events under P_0 alone, thereby avoiding the need to model the behavior of the tests given objects are present.

1.5 Optimality Results

For simplicity, write $X_{m,j}$ for $X_{\Lambda_{m,j}}$. From here on, we make the following assumptions about the the distribution of $\mathbf{X}$:

- $\{X_{m,j}\}$ are independent random variables under P_0;
- $\beta_{m,j} \doteq P_0(X_{m,j} = 0) = \beta_m, \ m = 0, 1, ..., M$;
- $\beta_0 \leq \beta_1 \leq \cdots \leq \beta_M$;
- $c(X_{m,j}) = c_m, \ m = 0, 1, ..., M$;
- $c_m = \Phi(\beta_m)$ with $\Phi(0) = 0$ and Φ increasing.

The independence assumption is violated in practice, but we make it in order to facilitate a theoretical analysis. The other assumptions are realistic, partly by design, although the power of the tests may differ slightly within levels.

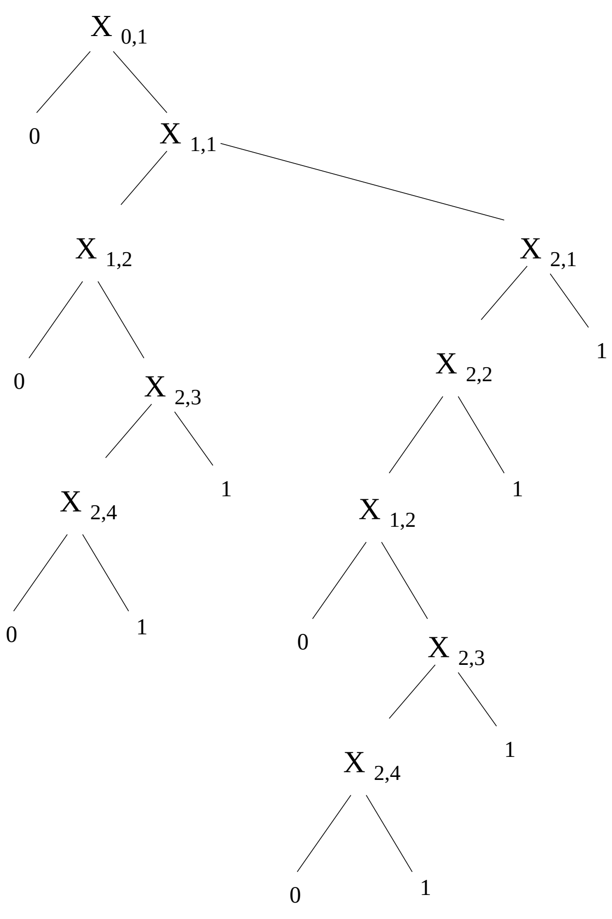

Figure 1.7. The CTF strategy tree for the case $M = 2$.

1.5.1 Testing $\hat{Y} = \emptyset$ *vs.*$\hat{Y} \neq \emptyset$

Consider first the problem of determining whether or not $\hat{Y} = \emptyset$, in other words, evaluating $\hat{Z} = \mathbf{1}_{\{\hat{Y} \neq \emptyset\}}$. The set-up is the same, except the terminal labels of T are simply "0" or "1." This problem was studied in Fleuret 1999 and Jung 2001 under the assumption that Φ is *convex*, i.e., cost is a convex, increasing function of power. One strategy is the depth-first CTF strategy, illustrated in Figure 1.7 for the case $M = 2$.

In Figure 1.8 two particular sample paths (branches) are depicted from a depth-first, CTF search of a five level hierarchy. The path on the left leads to the label $\hat{Z} = 0$ and the one on the right leads to $\hat{Z} = 1$.

The following result is proved in Jung 2001; earlier, Fleuret 2000 had shown that the coarsest test ($X_{0,1}$) is necessarily at the root. The convexity assumption can be relaxed to supposing that $\frac{c_m}{\beta_m}$ is increasing, $m = 0, ..., M$.

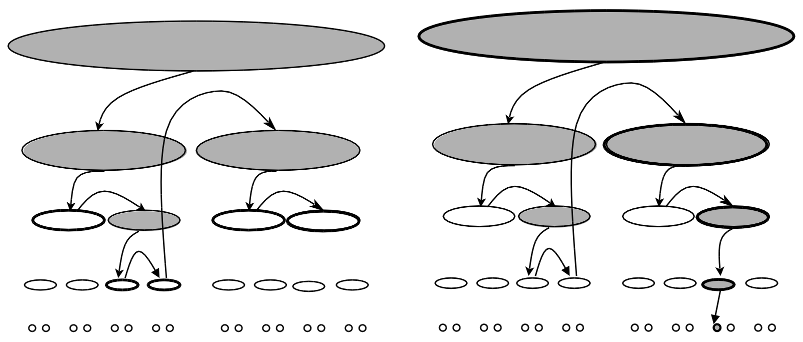

Figure 1.8. Two branches from a depth-first CTF search for a hierarchy with five levels. The tests are explored in the indicated order, with grey indicating a positive answer and white in bold outline indicating a negative answer; the other tests were not evaluated. Left: A null covering is encountered, ending the search. Right: A chain of ones is encountered (grey in bold outline), ending the search when the goal is to determine if $\hat{Y} = \emptyset$.

Theorem: *If Φ is convex, and $P = P_0$, depth-first CTF search is the optimal strategy for evaluating $\hat{Z}$.*

1.5.2 Determining $\hat{Y}$

Here the objective is to determine *all* 1-chains instead of merely whether or not one such chain exists. For either CTF strategy, the expected cost under P_0 is

$$E_0 C(T) = c_0 + \sum_{m=1}^{M} c_m 2^m \prod_{j=0}^{m-1} (1 - \beta_j).$$

As it turns out, this is the smallest possible mean cost:

Theorem: *The CTF strategy is optimal for any increasing cost sequence if $\beta_0 > .5$*

The proof of this result, and others based on varying cost models and test hierarchies, will appear in Blanchard & Geman 2002.

1.6 Error

We briefly discuss the theoretical error rates of the detector $\hat{Y}$ defined in (1.2).

In principle, the false negative rate is null. (Of course, in practice, this requires that X_Λ be invariant for Λ, which can be difficult to achieve.) The false positive error

$$\delta(\hat{Y}) \doteq P_0(\hat{Y} \neq \emptyset)$$

is determined by the (joint) distribution of $\{X_{m,j}\}$ under P_0, and depends on $\beta_0, ..., \beta_M$. The rate *per pixel* is then $\frac{\delta(\hat{Y})}{k^2}$; recall that the search for object locations is conducted in non-overlapping $k \times k$ blocks. A crude bound is to replace the probability of at least one 1-chain under P_0 by the sum of the probabilities, yielding

$$\begin{aligned} \delta(\hat{Y}) &\leq 2^M \prod_{m=0}^{M} (1 - \beta_m) \\ &\leq \frac{1}{2}\gamma^{M+1}, \ \gamma = 2(1 - \beta_0). \end{aligned}$$

To calculate $\delta(\hat{Y})$ exactly, notice that there is a one-to-one correspondence between realizations of a breath-first CTF evaluation of $\hat{Y}$ and realizations of a branching process with M generations starting from a single individual. Consequently,

Theorem: *The false positive error $\delta(\hat{Y})$ is the probability of no extinction for a non-homogeneous branching process with binomial family law $Bin(2, 1 - \beta_m)$ at generation $m = 0, ..., M$.*

1.7 Application to Face Detection

Recall that each $\lambda \in \mathcal{S}$ corresponds to the presentation or instantiation of an "object" in a predetermined library. The presentation includes information about the position, scale and other aspects of the geometric pose, and perhaps other properties of interest.

1.7.1 One Generic Class

Consider the simplified scenario of one generic object class and linear pose. More specifically, consider the problem of detecting all instances of frontal views of faces. The global procedure is to parse the scene at various scales, and at a sampling of locations, with a window of size 64×64, in each case applying a detector which computes the list of poses present in the window. In this case, the output $\hat{Y}$ of processing a window is simply a list of poses. What follows is a brief description of the algorithm; the details can be found in Fleuret & Geman 2001.

1.7.2 Pose Decomposition

The pose of a face is defined in the image plane, given by $\theta = (z, \sigma, \psi)$, where z is the center point between the eyes, σ is the distance (in pixels) between the eyes and ψ is the "tilt." The image is partitioned into non-overlapping 8×8 blocks, and the basic detector $\hat{Y}$ is applied to the image data in a window centered at each such block. These windows are the subimages in $\mathcal{I}$ in the previous sections. The detector $\hat{Y}$ produces a list of poses with $z \in [28, 36]^2, 8 \leq \sigma \leq 16, -20° \leq \psi \leq 20°$; of course in this case these are either false alarms or responses to the same face. Call this set of poses the "reference cell" $\Lambda_{0,1}$. Faces of scale $16 \leq \sigma \leq 32$ are found by downsampling the original scene and parsing again, etc.

The hierarchy $\{\Lambda_{m,j}\}$ of subsets of (reference) poses is constructed by recursively partitioning $\Lambda_{0,1}$ into a sequence of nested partitions; each cell Λ of each partition is a subset of poses which is included in exactly one of the cells in the preceding, coarser partition. At each level $m = 1, 2, ..., M$, one component of the pose θ is subdivided into equal parts - binary splits for scale and tilt and quaternary splits for position. Thus, for example, each cell $\Lambda_{1,j}, j = 1, 2, 3, 4$, of the the first partition corresponds constraining z to one of the four 4×4 subsquares of of the initial 8×8 square. There are two splits on z, two on σ and two on ψ, resulting in $M = 6$ levels (excluding the root).

1.7.3 Learning

Each test $X_{m,j} = X_{\Lambda_{m,j}}$ checks for a certain number of distinguished edge fragments, and hence is defined by a list $L_{m,j}$ of edges and a threshold (as described in §2) both determined during training. Consequently, the tests X are simply counting operators. Checking for an edge means evaluating a binary local feature ξ_l indexed by a position z_l and an orientation ϕ_l as described in §1.2.1. However, there is another, crucial, parameter - the "tolerance" of the edge - which allows one to achieve invariance to the poses in Λ. (The MAX filter in Riesenhuber and Poggio 1999 has the same aim.) The tolerance η is the length of a strip of pixels centered at z_l and perpendicular to the direction of the edge. The feature $\xi_l = 1$ if there is an edge at the given orientation at *any* location in the strip. Thus, η controls the amount of ORing, which in turn depends on the desired degree of invariance. All ξ in the list have the same η. The tolerance η is "large" for the coarse cells $\Lambda_{m,j}$ and "small" for the fine cells. It controls the tradeoff between invariance and discrimination.

All the tests $X_{m,j}$ in the hierarchy are built with the same learning algorithm; the lists differ due to varying training sets, corresponding to varying constraints on the set of poses. The experiments shown here are based on the ORL database which contains 400 grayscale face pictures of size 112×96 pixels. For each cell $\Lambda_{m,j}$, a synthetic training set of 1600 face images is constructed whose poses are in $\Lambda_{m,j}$. Again, the details are in Fleuret & Geman 2001, including how the lists of edges $\xi_l, l \in L$ are chosen.

1.8 Experiments and Conclusions

The experiments use a breadth-first CTF evaluation of $\hat{Y}$. Nearby detections are clustered, resulting in one estimated pose per face, indicated by a triangle. The false negative rate is not null, and there are false positives, on the order of $1-10$ per scene on average. Thus, when computation is efficiently organized, one can use very simple components (such as the counters X described in the previous section) and still achieve reasonable error rates, in fact comparable to the best ones reported in the literature for high resolution images; see for example Rowley 1999. In addition, detection is extremely fast, well under one second for a scene on the order of 400×400, which is faster than previously reported results. Two results are shown In Figures 1.9 and 1.11. As seen in Figure 1.12, coarse-to-fine processing leads to highly asymmetric scene processing in terms of spatial concentration, with orders of magnitude differences in the application of resources to different regions of the scene.

Recall that detection is far from a complete solution to scene interpretation. It leaves confusions unresolved and does not address occlusion and other complicating factors. Many examples of such specific class/pose confusions can be seen in a higher resolution rendering (not shown here) of the labeled detections in Figure 1.1. However, if highly visible objects are sure to be detected with a limited number of false alarms, it may then be computationally feasible to entertain very intense processing which is optimization-based but highly localized.

Finally, for detection, we can suppose that each test is constructed during an offline training phase, as in the previous section. Indeed, since we are not anticipating the specific confusions and occlusion patterns that might arise, we can afford to make a list of *all* the tests we wish to construct, learn them during training, and store all the instructions for execution. On the contrary, during contextual classification we might be obliged to generate hypotheses and construct tests online, i.e., during scene parsing. Such ideas are currently being explored in the context of character recognition.

1.9 References

1. Amit, Y. and D. Geman. 1999. A computational model for visual selection. *Neural Computation*, 11:1691-1715.

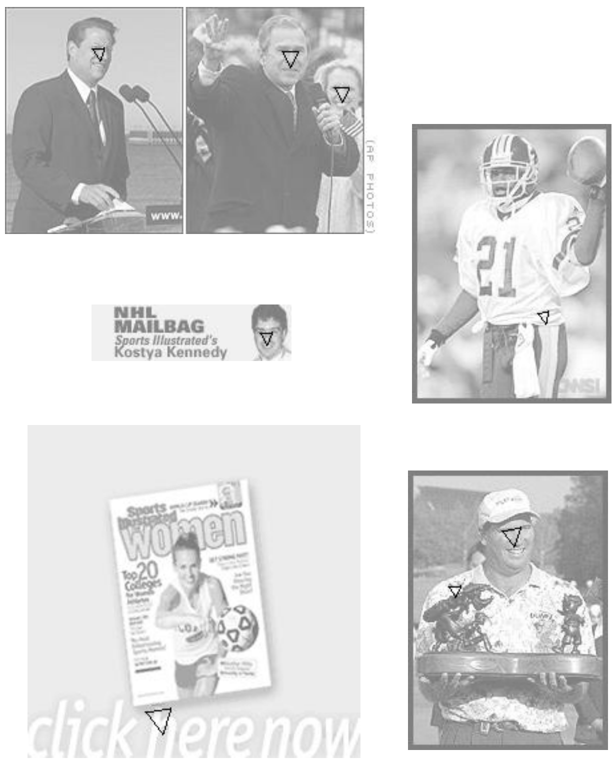

Figure 1.9. An experiment from the CNN website.

2. Blanchard, G. and D. Geman. 2002. Computational models for coarse-to-fine search, in preparation.

3. Fleuret, F. 2000. Detection hierarchique de visages par apprentissage statistique. Ph.D. thesis, University of Paris VI, Jussieu, France.

4. Fleuret, F. and D. Geman. 2001. Coarse-to-fine face detection. *Inter. J. Computer Vision*, 41:85-107.

5. Geman, D. and B. Jedynak. 2001. Model-based classification trees. *IEEE Trans. Info. Theory*, 47:1075-1082.

6. Geman, S., K. Manbeck, and D. McClure. 1995. Coarse-to-fine search and rank-sum statistics in object recognition. Technical Report, Division of Applied Mathematics, Brown University.

7. Geman, S., D. Potter, and Z. Chi. 2001. Composition systems. *Quarterly of Applied Mathematics*, to appear.

8. Jung, F. 2001. Reconnaissance d'objects par focalisation et detection de changements. PhD thesis, Ecole Polytechnique, Paris, France.

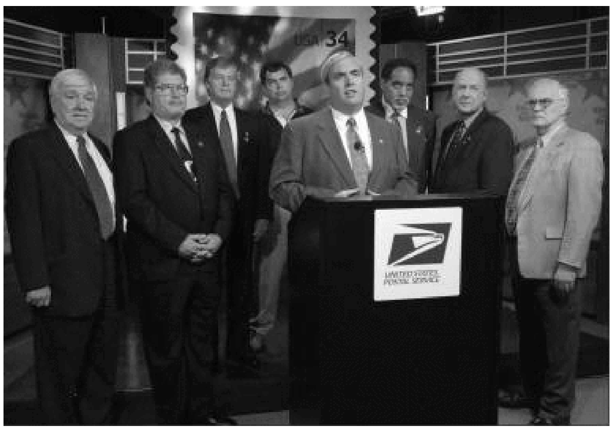

Figure 1.10. An experiment with a group photograph.

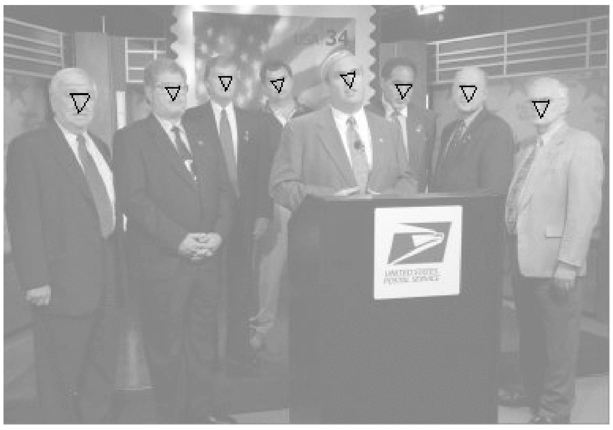

Figure 1.11. The detections for the group photo.

9. Lambdan, Y. and J.T. Schwartz and H.J. Wolfson. 1988. Object recognition by affine invariant matching. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 335-344.

10. Riesenhuber, M. and T. Poggio. 1999. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2:1019-1025.

11. Rowley, A.R. 1999. Neural network-based face detection. PhD Thesis, Carnegie Mellon University.

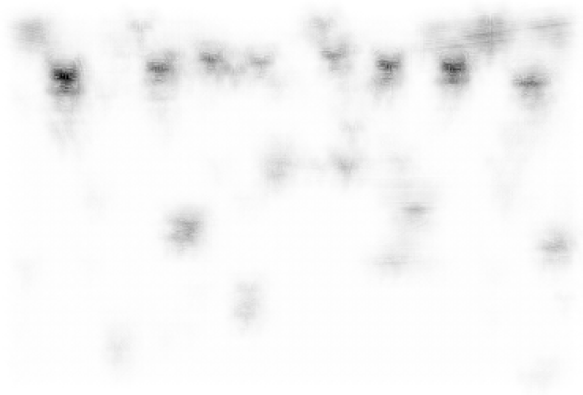

Figure 1.12. The coarse-to-fine nature of the algorithm is illustrated for the group photo by counting, for each pixel, the number of times the detector checks for the presence of an edge in its vicinity. The level of darkness is proportional to this count.