# Learning to Locate Informative Features for Visual Identification

**DRAFT: IJCV special issue** **www.cs.berkeley.edu/∼ferencz/vid/**

Andras Ferencz
Computer Science, U.C. Berkeley
ferencz@cs.berkeley.edu

Erik G. Learned-Miller
Computer Science, UMass Amherst
elm@cs.umass.edu

Jitendra Malik
Computer Science, U.C. Berkeley
malik@cs.berkeley.edu

## Abstract

*Object identification (OID) is specialized recognition where the category is known (e.g. cars) and the algorithm recognizes an object's exact identity (e.g. Bob's BMW). Two special challenges characterize OID. (1) Inter-class variation is often small (many cars look alike) and may be dwarfed by illumination or pose changes. (2) There may be many classes but few or just one positive "training" examples per class. Due to (1), a solution must locate possibly subtle object-specific salient features (a door handle) while avoiding distracting ones (a specular highlight). However, (2) rules out direct techniques of feature selection. We describe an on-line algorithm that takes one model image from a known category and builds an efficient "same" vs. "different" classification cascade by predicting the most discriminative feature set for that object. Our method not only estimates the saliency and scoring function for each candidate feature, but also models the dependency between features, building an ordered feature sequence unique to a specific model image, maximizing cumulative information content. Learned stopping thresholds make the classifier very efficient. To make this possible, category-specific characteristics are learned automatically in an off-line training procedure from labeled image pairs of the category, without prior knowledge about the category. Our method, using the same algorithm for both cars and faces, outperforms a wide variety of other methods.*

## 1. Introduction

Figure 1 shows six cars. The two leftmost cars were captured by one camera; the right four cars were seen later by another camera from a different angle. Suppose one wants to determine which images, if any, show the *same vehicle*. We call this task *visual object identification*. Object identification is a specialized form of object recognition in which the category is known (e.g. faces or cars) and one must recognize the exact identity of objects. Most existing identification systems are aimed at biometric applications such as identifying fingerprints or faces.

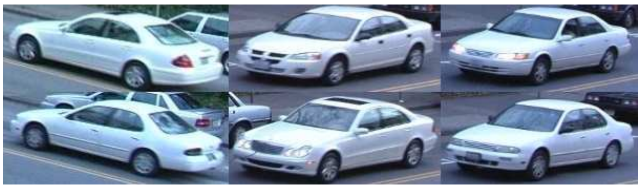

Figure 1: *An Identification Problem: Which cars match?* The two cars on the left were photographed from camera 1. Which of the four images on the right, taken by camera 2, match the cars on the left?

The general term *object recognition* refers to a whole hierarchy of problems for detecting an object and placing it into a group of objects. These problems can be organized by the generality and composition of the groups into which objects are placed. The goal of "recognition" can be to put objects in a very broad group such as vehicles, a narrower one such as cars, a highly specific group such as red sedans, or the narrowest possible group, a single element group containing a specific object, such as "Bob's BMW".

Here our focus is *identification*, where the challenge is to distinguish between visually similar objects of one category (e.g. faces, cars), as opposed to *categorization* where the algorithm must group together objects that belong to the same category but may be visually diverse [1, 11, 21, 25]. Identification is also distinct from *object localization*, where the goal is locating a specific object in scenes in which distractors have little similarity to the target object [16].

These differences are more than semantic: the Object Identification (OID) problem poses different challenges than its coarser cousin, Object Categorization (OC). Specifically, OID problems are characterized by the following two properties.

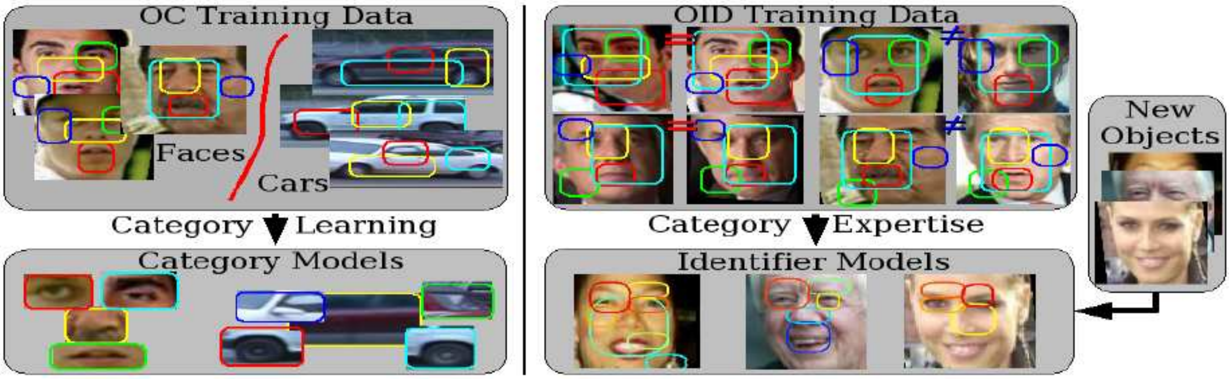

Figure 2: *Object Categorization vs. Identification:* this figure highlights the different learning involved in categorization and identification. The training sets for OC, shown on the left side, typically contain many examples of each category (e.g. faces and cars), which are then turned into a fixed model for each in a generative system or a decision boundary in a discriminative system. A training set for OID contains pairs of images from a known category, with a label of "same" or "different" (denoted by $=$ and $\neq$) for each pair. From these labeled pairs, the system must learn how to generate an identifier model given a new object (e.g. Mr. Carter) from the category (thus identification assumes correct categorization). For these models to work well, they should highlight distinctive regions of the object. Thus the models are different for each object.

1. The inter-class variation is often small (many cars look alike), and this variation is often dwarfed by illumination or pose changes (see Figure 1).
2. There are many classes (each object is a separate class), but few (in our case just one) positive "training" examples per class (e.g. one image representing "Bob's BMW").

People are good at identifying individual objects from familiar categories after seeing them only once. Consider faces. We zero in on discriminative features *for a particular person* such as a prominent mole or unusually thick eyebrows, yet are not distracted by equally unusual but non-repeatable features such as a messy strand of hair or illumination artifacts. Domain specific expertise makes this possible: having seen many faces one learns that a messy strand of hair is not often a reliable feature. Human vision researchers report that acquisition of this expertise is accompanied by significant behavioral and physiological changes. Diamond et al. [9] showed that dog experts perform dog identification differently than non experts; Tarr et al. [22] argued that the brain's fusiform face area does visual processing of categories for which expertise has been gained.

Categorization algorithms such as [1, 26, 24, 5] learn to recognize objects that belong to a category. Here, we are attempting to go one step beyond this by becoming category experts, where instead of having a fixed set of features that we look for to recognize new objects, we are able to predict the features of the new object that will be the most informative for distinguishing it from other objects of the same category. Figure 2 highlights this difference. Note that categorization is a prerequisite for identification, because identification systems such as ours assume that the given objects are from the known category.

The processes that occur during Object Categorization (OC) and Object Identification (OID) can be formally characterized. In functional notation, the stages for OC are

1. (Off-line) trainer $\mathcal{T}_{cat}$: class training images $\mapsto \mathcal{C}_{cat}$,
2. (On-line) classifier $\mathcal{C}_{cat}$: test image $\mapsto$ class label.

There is nothing novel here, just the standard paradigm of statistical learning. It relies implicitly on having enough examples of each class to learn discriminative features.

For OID, we assume off-line access to plenty of examples of the category (cars, dogs, faces). We then must develop an on-line classifier for a future image of Bob's BMW, given only one example of it. We decompose the on-line process into two stages: (a) producing an "identifier", a classifier specialized to reidentify a specific object based on a single example of it, and (b) running the "identifier" on the incoming data stream. These on-line stages are preceded by the off-line process of learning *category specific* characteristics, resulting in an "identifier generator". Thus, the three stages for OID are

1. (Off-line) trainer $\mathcal{T}_{id}$: category training images $\mapsto \mathcal{H}_{id}$,
2. (On-line) identifier generator $\mathcal{H}_{id}$: object image $\mapsto \mathcal{C}_{id}$,
3. (On-line) classifier $\mathcal{C}_{id}$: test image $\mapsto \{$same, different$\}$.

We stress that step 1 learns *category specific* characteristics, while step 2 creates an *object specific* classifier. Now we address details.

First we need to pick a family of classifiers $\mathcal{C}_{id}$. Motivated by the success of patch (a.k.a. part or fragment) based representations [24, 26] for OC, we use them for OID as well. Specifically, we develop an OID system whose generated classifier $\mathcal{C}_{id}$ (step 3) is a patch-based classification cascade similar to that of Vidal-Naquet and Ullman [24], where evidence from features is accumulated incrementally until a "same" or "different" decision can be made. The tricky part is to give $\mathcal{H}_{id}$ the ability to pick out object specific discriminative features (e.g. a prominent door handle in one car, a roof rack in another). But how can we know that a patch containing a prominent door handle is discriminative, based on *a single image*, when we have never seen a door handle exactly like it before?

The core of our approach is to use *hyper-features*, which are generic position and appearance characteristics of a patch. Examples include location of a patch, edge contrast in the patch and the dominant oriented energy in the patch. We might, in the process of becoming a car identification expert, expect to learn that patches about half-way up with strong edge contrast and a dominant horizontal orientation are particularly informative. When given the specific example of Bob's BMW, the identifier generator $\mathcal{H}_{id}$ could produce an object-specific cascade with the first test based on the patch containing the door handle. Whereas for Jen's Ford, the same set of hyper-features will result in a different ordering of salient patches, resulting in a different classification cascade with the first test using a patch containing the roof rack (see Figure 9).

More precisely, to instantiate $\mathcal{C}_{id}$ (step 2), the function $\mathcal{H}_{id}$ is given a *single* model image of the object (e.g. Bob's BMW) and produces a sequence of patches ordered from most informative to least. To estimate the likely information content of a patch and then to score its correspondence in a test image, our technique uses generalized linear models (GLMs) to estimate a generative model for the dissimilarity between matched (model-test) patch pairs. The "same" and "different" distributions for each patch are estimated using the GLM based on the hyper-features of that patch. By estimating bivariate "same" and "different" distributions for neighboring patches, we model the dependency relationships, allowing us to compute a sequence of patches with high joint information content. This sequence is object-specific, and may emphasize different parts of each object.

The off-line training $\mathcal{T}_{id}$ (step 1), given a set of image pairs from the category with each pair labeled "same" or "different," produces a class-specific $\mathcal{H}_{id}$, learning (a) the GLM based on position and appearance hyper-features, (b) the dependency model between image patches based on *similarity* of their hyper-features, and (c) a set of thresholds for the cascade. The specific hyper-features used are themselves automatically selected during this training step from a large pool of candidate patch characteristics.

Section 3 summarizes the three stages of our algorithm, $\mathcal{T}_{id}$, $\mathcal{H}_{id}$, and $\mathcal{C}_{id}$. Section 4 details our model for estimating "same" and "different" distributions for a patch. Section 5 describes our patch dependency model that allows us to generate a sequence of informative patches. From this sequence, we build the cascade in Section 6 by finding stopping thresholds for making "same" or "different" decisions. Section 7 details our extensive experiments on multiple car and face data sets.

## 2. Previous Work

In this section, we highlight previous papers that have influenced our work.

### 2.1. Part-Based Recognition

Breaking an image into local subparts, where each part is encoded and matched separately, is a popular technique for object recognition (both categorization and identification) [5, 2, 12, 14, 16, 19, 20, 24, 27, 26]. This strategy helps to mitigate the effects of distortion due to pose variation, as local regions are more likely than the whole object to be related by simple transformations. It also contains the disturbance due to occlusion and localized illumination effects such as specularities. Finally, it separates modeling of appearance and position. The key idea is that the parts, which are allowed to move relative to one other, can be treated as semi-independent assessments for the recognition task. The algorithms then combine this evidence, optionally using the positional configuration of the detected parts as an additional cue, to determine the presence or absence of the object. The choices of representations and comparison metrics for the parts vary widely in the above systems, and must be chosen to fit the task at hand (see Section 3.2.2).

### 2.2. Learning from Few Examples

Identification can be thought of as a special case of the traditional supervised classification problem. Here, the classes to be distinguished are not "cars" and "non-cars" but rather images of a particular car, say Bob's BMW, versus images of other cars. If we are given only a single image of Bob's car, we cannot use standard supervised feature selection methods [24, 25, 26] that determine saliency by comparing each feature (in our case, a local image part) to a set of matching and non-matching (in-class and out-of-class) examples.

One possible solution to this problem is to try to pick universally good features, such as corners [14, 16], for detecting salient points. Such features are likely to be suboptimal as they are not category specific: we expect Bob to use different kinds of image features when distinguishing his car from other cars versus when he is distinguishing his dog from another dog.

Another possibility is to build generative models for each class including such characteristics as the typical illuminations, likely deformations, and modeling the effects of variation in viewing direction. With a precise enough model, an algorithm should be able to find good features for discriminating instances of the category from each other [7]. Alternatively, the good features are sometimes explicitly coded into the algorithm [27]. However, this tends to be complicated and time consuming, and must be done individually for a particular class (see Section 2.3 below).

Instead, we wish to teach our system to automatically generalize what features tend to be salient by looking at other labeled matching and non-matching pairs of the same category (cars) but from different "classes" (which in this case are individual objects) such as Jen's car or Bob's car. Thus when the system is given a novel model image, it should be able to scan through all of its candidate features and determine, by analogy with similar features found in the training set of labeled pairs, how salient the feature is likely to be. In our case, the set of candidate features are made up of all possible sub-image patches.

Two related pieces of work that leverage the modeling of one class or set of classes to improve the modeling of new classes are [18] and [11]. In the former, distributions over parameters of a similarity transformation that are learned from one group of classes (letters) are then used to model other classes (digits) for which only a single example is provided. In the latter, priors are learned from one set of classes to train a detector for a new class given only a small number of positive examples of that class. In particular, [11] learns the parameters for a constellation model with a fixed number of parts. In both of these works, the set of hidden variables (the transformations in [18], or the parameters of the constellation model) are predefined and the generalization from other categories can be thought of as learning priors for these fixed sets of variables.

In contrast, our $\mathcal{T}_{id}$ actually learns how to identify an arbitrary number of good features for the given category. Thus our final classifier $\mathcal{C}_{id}$, while always a cascade of image patches taken from the model object, will have a different set of patches (in size, location, and count) for each object.

### 2.3. Face Identification

Our goal in this work is to develop an identification system that is not designed for any particular category, but instead automatically learns category-specific characteristics. Nonetheless, it is useful to consider previous identification systems that were designed with a particular category in mind. Here we highlight a few face identification systems that are representative and relevant for our work. For an extensive survey of the field, we refer the reader to [28].

Eigenfaces [23] (PCA) and later fisherfaces [3] (FDA/LDA) closely follows the three step procedure laid out above. These are "holistic" methods in that they use the whole face region as raw input to the recognition system. Specifically, they take registered and intensity normalized faces (or labeled collections of images in the case of the FDA/LDA techniques) and find a lower dimensional subspace that, it is hoped, is more conducive to identification (this is analogous our step 1, $\mathcal{T}_{id}$). To build a classifier, the model image is projected into this subspace, and the classifier compares the model and test images within this subspace.

More complex, feature-based methods typically use more face-specific models and hand labeled data. Two techniques in this category that have had a significant impact are Elastic Bunch Graph Matching [27], where hand selected fiducial points are matched within a graph that defines their relative positions, and [7], which maps images onto a 3D morphable face model.

## 3. Algorithm Overview

In this section, we outline the basic components of our system. We describe the training ($\mathcal{T}_{id}$), classifier (identifier) generating ($\mathcal{H}_{id}$), and classification ($\mathcal{C}_{id}$) functions in reverse order, starting with the final form of the object-specific classifier.

### 3.1. Preprocessing: Detection and Alignment

Our algorithm, as most identification systems, assumes that all images are known to contain objects of the given category (e.g. cars or faces) and have been brought into rough correspondence. For our algorithm, an approximate alignment is sufficient, because we search for matching patches in a small neighborhood around the expected location. The specific detection and alignment methods used for our various data sets are described in Section 7. For example, for the "Faces" data set, a face detector was followed by a parts-based model that aligns the eyes, nose and mouth.

### 3.2. Classifier $\mathcal{C}_{id}$

The classifier $\mathcal{C}_{id}$ decides if a test (a.k.a. right) image $I^R$ is the same ($C = 1$) or different ($C = 0$) than the model (a.k.a. left) image $I^L$ it was trained for.

#### 3.2.1. Patches

Our strategy is to break up the whole image comparison problem into multiple local matching problems, where we encode a small patch $F_j^L$ ($1 \leq j \leq n$) of the model image and compare each piece separately [24, 26]. Although the exact choice of features, their encoding and comparison metric is not crucial to our technique (we could have, for example, used features such as vehicle length, height, average color, etc., within the same framework), we wanted to use features that were general enough to use in a wide variety of settings, but informative enough to capture the relative locality of object markings as well as small and large details of objects.

We begin by computing a Gaussian pyramid for each image. For each patch, based on its size, the image pixels are extracted from a level of the pyramid such that the number of pixels in the representation is approximately constant. Then we encode the pixels by applying a first derivative Gaussian odd-symmetric filter to the patch at four orientations (horizontal, vertical, and two diagonal), giving four signed numbers per pixel.

### 3.2.2. Matching

To compare a model patch $F_j^L$ to an equally encoded area of the right image $F_j^R$, we compute the normalized correlation

$$d_j = 1 - \mathit{CorrCoef}(F_j^L, F_j^R) \tag{1}$$

between the arrays of orientation vectors. Thus $d_j$ is a patch appearance distance where $0 \leq d_j \leq 2$.

As the two car images are in rough alignment, we need only to search a small area of $I^R$ to find the best corresponding patch $F_j^R$ - i.e. the one that minimizes $d_j$. We will refer to such a matched left and right patch pair $F_j^L, F_j^R$, together with the derived distance $d_j$, as a *bi-patch*. This appearance distance $d_j$ is used as evidence for deciding if $I^L$ and $I^R$ are the same ($C = 1$) or different ($C = 0$).

In choosing this representation and comparison function, we compared a number of commonly used encodings, including Lowe's SIFT features [16] and shape contexts [4]. However, we found that due to the nature of the problem - where distinct objects can look very similar except for a few subtle differences - these techniques, which were developed to be robust to small differences, did not perform well. Specifically, using SIFT features as described in [16] (without category specific learning) resulted in false-positive error rates that were an order of magnitude larger than our best results and a factor of 2-3 worse than our baseline "No Hyper-Features" results (at the same recall rate). Among dense patch features, we chose normalized correlation of filter outputs after experiments comparing this distance function to L1 and L2 distances, and the encoding to raw pixels and edges as in [26].

### 3.2.3. Likelihood Ratio Score

We pose the task of deciding if the a test image $I^R$ is the same as a model image $I^L$ as a decision rule

$$\begin{aligned} R &= \frac{P(C=1|I^L, I^R)}{P(C=0|I^L, I^R)} & (2)\\ &= \frac{P(I^L, I^R|C=1)P(C=1)}{P(I^L, I^R|C=0)P(C=0)} > \lambda. & (3) \end{aligned}$$

where $\lambda$ is chosen to balance the cost of the two types of decision errors. The priors are assumed to be known.[1] Specifically, for the remaining equations in this paper, the priors are assumed to be equal, and hence are dropped from subsequent equations.

With our image decomposition into patches, the posteriors from Eq. (2) will be approximated using the bi-patches $F_1, ..., F_n$ as $P(C|I^L, I^R) \approx P(C|F_1, ..., F_m) \propto P(F_1, ..., F_m|C)$. Furthermore, we will assume a naive Bayes model in which, conditioned on $C$, the bi-patches are assumed to be independent (see Section 5 for our efforts to ensure that the selected patches are, in fact, as independent as possible). That is,

$$\begin{aligned} R = \frac{P(I^L, I^R|C=1)}{P(I^L, I^R|C=0)} &\approx \frac{P(F_1, ..., F_m|C=1)}{P(F_1, ..., F_m|C=0)} & (4)\\ &= \prod_{j=1}^{m} \frac{P(F_j|C=1)}{P(F_j|C=0)}. & (5) \end{aligned}$$

In practice, we compute the log of this likelihood ratio, where each patch contributes an additive term (denoted $\mathcal{LLR}_i$ for patch $i$). Modeling the likelihoods $P(F_j|C)$ in this ratio is the central focus of this paper.

In our current system, the only information from bi-patch $F_j$ that we use for scoring is the distance $d_j$ (we don't, for example, use the relative position where the matching patch $F_j^R$ was found). Thus, to convert $d_j$ to a score, $\mathcal{C}_{id}$ stores probability distributions $P(D_j|C=1)$ and $P(D_j|C=0)$ for each patch and computes the log likelihood ratio. (Note: $d_j$ refers to the specific measured distance for a given model and test image, while $D_j$ denotes the random variable from which $d_j$ is a sample). After $m$ patches have been matched, assuming independence, we score the match between images $I^L$ and $I^R$ using the sum of log likelihood ratios of matched patches:

$$R = \sum_{j=1}^{m} \log \frac{P(D_j = d_j|C=1)}{P(D_j = d_j|C=0)}. \tag{6}$$

To compute this, we must evaluate $P(D_j = d_j|C=1)$ and $P(D_j = d_j|C=0)$. In our system, both of these will take the form of gamma distributions $\Gamma(d_j; \theta_j^{C=1})$ and $\Gamma(d_j; \theta_j^{C=0})$, where the parameters $\theta_j^{C=1}$ and $\theta_j^{C=0}$ are defined as part of the classifier $\mathcal{C}_{id}$ for each patch and are set by $\mathcal{H}_{id}$ based on hyper-features.

### 3.2.4. Making a Decision

In the above $\mathcal{C}_{id}$ matched a fixed number of patches ($m$), computed the score R by Eq. 6, and compared it to a threshold $\lambda$. $R > \lambda$ meant that $I^L$ and $I^R$ are the same. Otherwise they are declared different. In Section 6, we define a cascade from the sequence of patches by applying thresholds after the first $k$ patches have been matched. These thresholds allow $\mathcal{C}_{id}$ to stop and declare a result after only matching $k$ patches.

[1] For our car tracking application (see Section 7.3), dynamic models of traffic flow can supply the prior on $P(C)$.

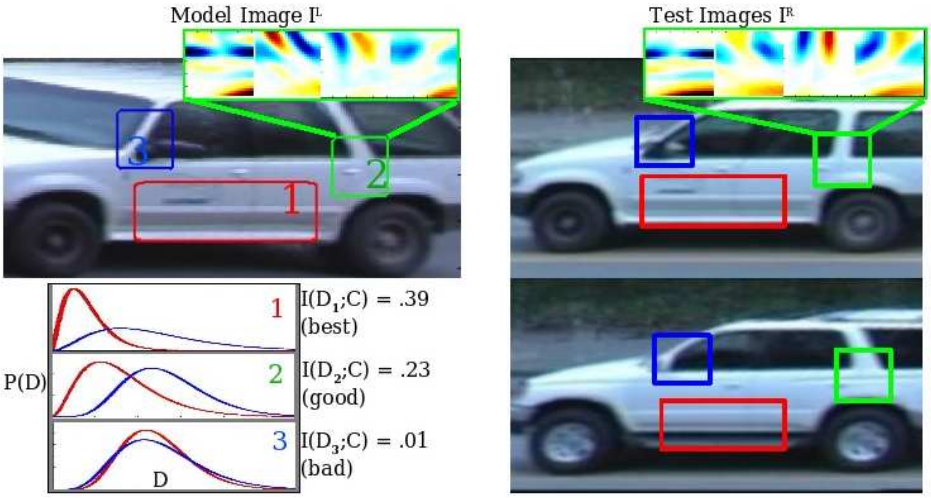

Figure 3: *The Classifier* $\mathcal{C}_{id}$. On the left, a model image $I^L$ is shown with a classifier composed of three patches (these would not be the actual top three patches selected by $\mathcal{H}_{id}$). The classifier generator $\mathcal{H}_{id}$ estimated same and different distributions (red and blue curves, respectively) for these three patches. Our patch encoding using oriented filter channels is shown for patch 2. The classifier matches the patches to the test images, computes the log likelihood ratio score for each using the estimated distributions, and makes a same vs. different decision based on the sum $R$ (the top image is the correct match). Looking at the images, compare the informativeness of patches 1 and 3: matching patch 1 should be very informative, since the true matching patch (top) is much more similar then the best matching patch in the other "different" image (bottom); matching patch 3 should be much less so, as both matching test image patches look completely dissimilar to the model. This fact is correctly estimated by $\mathcal{H}_{id}$ based on the position and appearance of these patches in the model image (see the mutual information values $I(D_j|C)$ next to the distributions).

#### 3.2.5. Summary of $\mathcal{C}_{id}$

To summarize, the classifier $\mathcal{C}_{id}$ is defined by:

1. a sequence of patches of varying sizes $F_j^L$ taken from the model image $I^L$,
2. for each patch $F_j^L$, a pair of parameters $\Theta_j^{C=1}$ and $\Theta_j^{C=0}$ that define the distributions $P(D_j|C = 1)$ and $P(D_j|C = 0)$, and
3. optionally, a pair of thresholds $\lambda_k^{C=1}$ and $\lambda_k^{C=0}$ applied after matching the $k$-th patch.

For an example, refer to Figure 3.

### 3.3. Classifier Generator $\mathcal{H}_{id}$

The classifier generator $\mathcal{H}_{id}$ must take in a single model image $I^L$ of a new object from the given category and produce a sequence of patches $F_1^L, ..., F_m^L$ and their associated gamma distribution parameters, $\Theta_1^{C=1}, ..., \Theta_m^{C=1}$ and $\Theta_1^{C=0}, ..., \Theta_m^{C=0}$, for scoring based on the appearance distance measurement $d_j$ (which is measured when the patch $F_j^L$ is matched to a location in a test image $I^R$).

#### 3.3.1. $Q^C$: Estimating $P(D_j|C)$ (forward declaration)

Since being able to estimate good distributions $\Theta_j^{C=1}$ and $\Theta_j^{C=0}$ ($\Theta_j^C$ for short) for any model patch $F_j^L$ is also the key to picking good patches, we first summarize this step. Conceptually, we want $\Theta_j^C$ to be influenced by what patch $F_j^L$ looks like and where it is on the object. That is, we want a pair of functions $Q^{C=1}$ and $Q^{C=0}$ that map the position and appearance of the patch $F_j^L$ to the parameters of the gamma distribution $\Theta_j^{C=1}$ and $\Theta_j^{C=0}$:

$$Q^{C=1} : F_j^L \mapsto \Theta_j^{C=1}$$

$$Q^{C=0} : F_j^L \mapsto \Theta_j^{C=0}$$

These functions are described in detail in Section 4.

#### 3.3.2. Estimating Saliency

If we define the saliency of a patch as the amount of information about the decision likely to be gained if the patch were to be matched, then it is straightforward to estimate saliency given $P(D_j|C = 1)$ and $P(D_j|C = 0)$. Intuitively, if $P(D_j|C = 1)$ and $P(D_j|C = 0)$ are similar distributions, we don't expect much useful information from a value of $d_j$. On the other hand, if the distributions are very different, then $d_j$ can potentially tell us a great deal about our decision. Formally, this can be measured as the mutual information between the decision variable $C$ and the random variable $D_j$ (we assume equal priors on C, $P(C = 0) = P(C = 1) = 0.5$):

$$I(D_j; C) = H(D_j) - H(D_j|C).$$

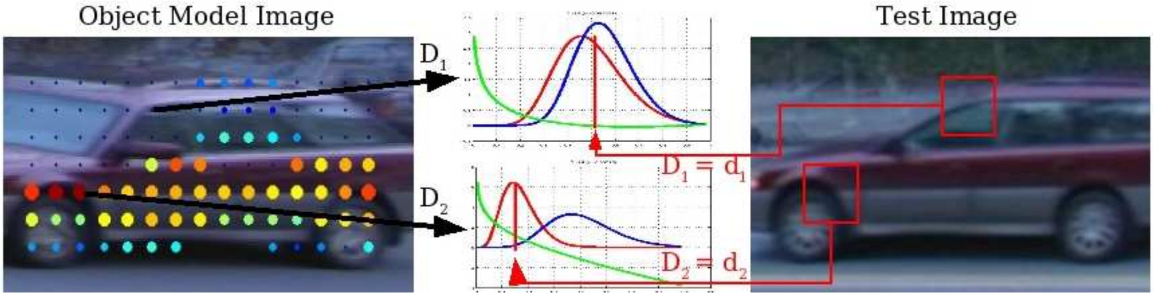

Figure 4: *Estimating the Distributions and Informativeness of Patches.* The classifier generator $\mathcal{H}_{id}$ takes an object model image, samples patches, estimates the same and different distributions (from the hyper-features using functions $Q^C$) and mutual information score for each, and selects the sequence of patches to use for classifying test images. The distributions were computed based on 10 selected hyper-features derived from the position and appearance of each patch $F_j^L$. In the model image (left), each candidate patch is marked by a dot at its center, where the size and color represent the mutual information score (bigger and redder means more informative). The estimated distributions for two patches is shown in the center (red and blue curves), together with the log likelihood ratio score (green line). When the patches are matched to a test image, the resulting appearance distance $d_j$ is indicated as a red vertical line.

Here $H()$ is Shannon entropy. The key fact to notice is that this measure can be computed just from the estimated distributions of $D_j$ (which, in turn, were estimated from the position and appearance of the model patch $F_j^L$) before the patch has been matched.

#### 3.3.3. Finding Good Patches

The above mutual information formula allows us to estimate the saliency of any patch. Thus defining a sequence of patches to examine in order, from among all candidate patches, seems straightforward:

1. for each candidate patch
   (a) estimate the distributions $P(D_j|C)$ from $F_j^L$ using the functions $Q^C$
   (b) compute the mutual information $I(D_j; C)$
2. choose the top $m$ patches sorted by $I(D_j; C)$

The problem with this procedure is that the patches are not independent. Once we have matched a patch $F_j^L$, the amount of *additional* information we are expected to derive from matching a patch $F_i^L$ that overlaps $F_j^L$ is likely to be less then the mutual information $I(D_i; C)$ would suggest. We discuss a solution to this problem in Section 5.

However, assuming that this dependency problem can be solved, and given the functions $Q^C$, we have a complete algorithm for generating the classifier $\mathcal{C}_{id}$ from a single image.

### 3.4. Off-line Training $\mathcal{T}_{id}$

The task of the off-line training step $\mathcal{T}_{id}$ is to define the two functions $Q^{C=1}$ and $Q^{C=0}$, that estimate the distributions $P(D_j|C = 1)$ and $P(D_j|C = 0)$ from the position and appearance of the patch $F_j^L$ (see Section 4). Additionally, $\mathcal{T}_{id}$ builds the dependency model of Section 5 and defines the cascade thresholds $\lambda_k^{accept}$ and $\lambda_k^{reject}$ as described in Section 6.

This off-line training is given a large collection of image pairs from the category (see Section 7 for details about our data sets), where each left-right image pair is labeled as "same" or "different". A large number of patches $F_j^L$ are sampled from the left images and matched to the right images (by finding the best matching $F_j^R$) in the same manner as during classification $\mathcal{C}_{id}$ (see Matching in Section 3.2), and the appearance distance $d_j$ is recorded.

## 4. Hyper-Features and Generalized Linear Models

In this section, we define the form of the functions $Q^C$ for $C = \{0, 1\}$ that map the position and appearance of a model image patch $F_j^L$ to the parameters $\Theta_j^C$ of the gamma distributions for $P(D_j|C)$, and show how to learn the free parameters of these functions from the training data during off-line category training $\mathcal{T}_{id}$.

For this section, Figure 5 shows the performance of our models on the *Cars 1* data set, with no patch selection (i.e. we use 105 patches sampled at fixed, equally spaced locations) and with patch sizes fixed to 25x25. The two bottom curves are baseline experiments. The *direct image comparison* method compares the center part of the images using normalized correlation on a combination of intensity and filter channels and attempts to overcome slight misalignment. The *patch-based baseline* assumes a global distribution for $D_j$ that is the same for all patches.

We want to differentiate patches by producing distributions $P(D_j|C = 1)$ and $P(D_j|C = 0)$ tuned for patch $F_j^L$. In this section, we will make this dependence on the

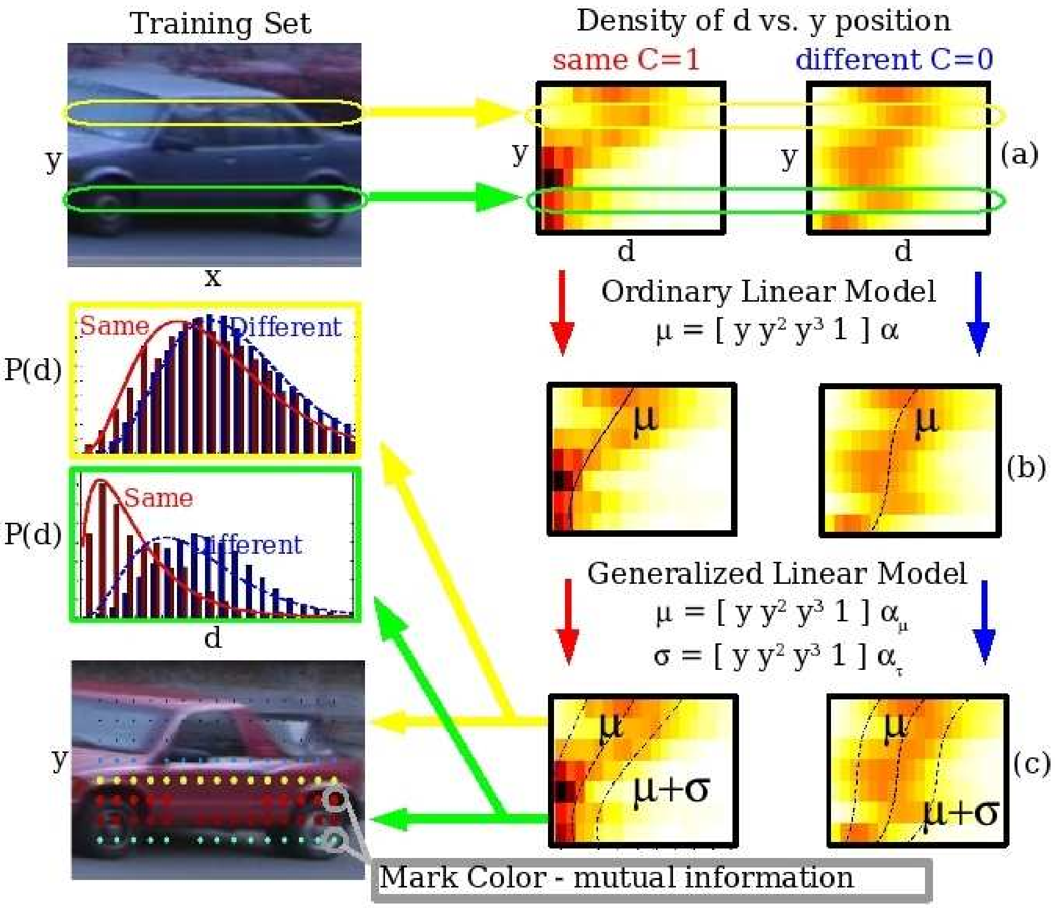

Figure 6: *Fitting a GLM to the gamma distribution.* We demonstrate our approach by fitting a gamma distribution, through the latent variables $\Theta = (\mu, \sigma)$, to the y position of the patches (in practice, we use the parameterization $\Theta = (\mu, \gamma)$). Here we allowed $\mu$ and $\sigma$ to be a 3rd degree polynomial function of y (i.e. $\mathbf{Z} = [\mathbf{y^3}, \mathbf{y^2}, \mathbf{y}, \mathbf{1}]^\mathbf{T}$). Each row of the images labeled **(a)** displays the empirical density of $d$ conditioned on the $y$ position of the left patch ($F^L$) for all bi-patches sampled from the training data (darker means higher density). There are 2 of these: one for bi-patches taken from matching vehicles (the pairs labeled "same"); the other from mismatched data ("different" pairs). **(b)** show the ordinary linear model fit, where the curve represents the mean. The outer curves in **(c)** show the $\pm\sigma$ (one standard deviation) range fit by the GLM. On the bottom left, the centers of patches from a model object are labeled with a dot whose size and color corresponds to the mutual information score $I(D;C)$. For 2 selected rows (each a range of y positions), the empirical distributions are displayed as a histogram. The gamma distributions as fit by the GLM are superimposed on the histograms. Notice that this model has learned that the top portion of the vehicles in the training set is not very informative, as the two distributions (the red and blue lines in the top histogram plot) are very similar ($D_j$ will have low mutual information with $C$). In contrast, the bottom area is much more informative.

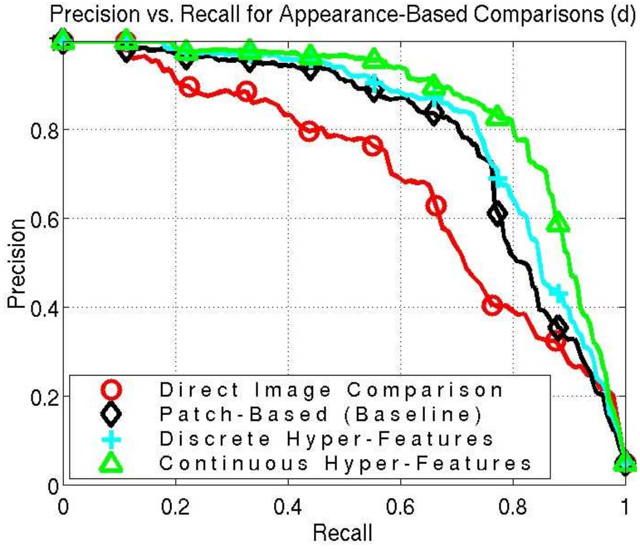

Figure 5: *Identification with Patches.* The bottom curve shows the precision vs. recall for non-patch based direct comparison of rectified images. (An ideal precision-recall curve would reach the top right corner.) The other curves show the performance of our algorithm on the *Cars 1* data set, using all fixed sized patches (25x25 pixels) sampled from a grid such that each patch overlaps its neighbors by 50%. Notice that all three patch based models outperform the direct method. The three top curves show results for various models of $d_j$: (1) no dependence on patch characteristics (Baseline), (2) non-parametric from Section 4.1 (Discrete), and (3) generalized linear with hyper-feature selection from Sections 4.2 and 4.3 (Continuous). The linear model significantly outperforms all of others. Compared to the baseline patch method it reduces the error in precision by close to 50% for most values of recall below 90% showing that conditioning the distributions on hyper-features boosts performance.

model patch $F_j^L$ (or derived quantities of it) explicit by writing $P(D_j|F_j^L, C)$ (in later chapters, we will often drop this term to enhance readability). When a training set of "same" $(C = 1)$ and "different" $(C = 0)$ images are available *for a specific model image*, estimating these distributions directly for each patch is straightforward. But how can we estimate the distribution $P(D_j|F_j^L, C = 1)$, where $F_j^L$ is a patch from a new model image, when we only have that *single positive example* of $F_j^L$? The intuitive answer: by finding analogous patches in the training set of labeled (same/different) image pairs. However, since the space of all possible patches (for a 25x25 patch, appearance and position is a point in $\Re^{25*25+2}$) is very large, the chance of having seen a very similar patch to $F_j^L$ in the training set is small. In the next 2 sections we present two approaches both of which rely on projecting $F_j^L$ into a much lower dimensional space by extracting meaningful features from its position and appearance (the *hyper-features*).

## 4.1. Discrete Hyper-Features

First we attempt a non-parametric approach, where we bin the hyper-features into a number of pre-specified axis-aligned bins. For example we might break the x coordinate of the position into four bins and the y into three and the contrast into two and then label each patch with its position in this 4-by-3-by-2 histogram (see *Discrete* curve in Figure 5). For each bin, we estimate $P(D_j|F_j^L, C = 1)$ and $P(D_j|F_j^L, C = 0)$ by computing the parameters $(\Theta_j^C)$ of the gamma distributions from all of the bi-patches $F_j$ whose left patch $F_j^L$ falls into that bin. More precisely, we use bi-patches from the "same" image pairs to estimate $\Theta_j^{C=1}$ and the "different" pairs to find $\Theta_j^{C=0}$.

Doing the same thing but modeling the distribution also non-parametrically (using a normalized histogram that also bins the value of $d_j$) produces very similar results when enough data is available in each bin and degrades when there are too many bins.

## 4.2. Continuous Hyper-Features

Similarly, when too many hyper-feature bins are introduced, the performance of the discrete model using parametric distributions also degrades. The problem is that the amount of data needed to populate the histograms grows exponentially with the number of dimensions. In order to add additional appearance-based hyper-features, such as mean intensity, oriented edge energy, etc., we moved to a smooth parametric model for the way the hyper-features influence the distribution.

Specifically, as before, we model the distributions $P(D_j|F_j^L, C = 1)$ and $P(D_j|F_j^L, C = 0)$ as gamma distributions $(\Gamma(\Theta^C))$ parameterized by the mean and shape parameter $\Theta = \{\mu, \gamma\}$ (see the left side of Figure 6 for examples of the gamma approximations to the empirical distributions). The smooth variation of $\theta$ with respect to the hyper-features can be modeled using a generalized linear model (GLM). Ordinary (least-squares) linear models assume that the data is normally distributed with constant variance. GLMs are extensions to ordinary linear models that can fit data which is not normally distributed and where the dispersion parameter also depends on the covariates (see [17] for more information on GLMs).

Our goal is to fit gamma distributions to $P(D_j|F_j^L, C = 1)$ and $P(D_j|F_j^L, C = 0)$ for various patches by maximizing the probability density of data under gamma distributions whose parameters are simple polynomial functions of the hyper-features. Consider a set $X_1, ..., X_k$ of hyper-features such as position, contrast, and brightness of a patch. Let

$\mathbf{Z} = [Z_1, ..., Z_l]^T$ be a vector of $l$ pre-chosen functions of those hyper-features, like squares, cubes, cross terms, or simply copies of the variables themselves. Then each bi-patch distance distribution has the form

$$P(d|X_1, X_2, ..., X_k, C) = \Gamma(d;\ \alpha_{\mathbf{C}}^{\mu} \cdot \mathbf{Z},\ \alpha_{\mathbf{C}}^{\gamma} \cdot \mathbf{Z}), \quad (7)$$

where the second and third arguments to $\Gamma()$ are mean and shape parameters. For our GLM, we use the identity link function for both $\mu$ and $\gamma$. While the identity is not the canonical link function for $\mu$, its advantage is that our ML optimization can be initialized by solving an ordinary least squares problem. We experimentally compared it to the canonical inverse link ($\mu = (\alpha_C^{\mu} * \mathbf{Z})^{-1}$), but observed no noticeable change in performance on our data set. Each $\alpha$ (there are four of these: $\alpha_{C=0}^{\mu}, \alpha_{C=0}^{\gamma}, \alpha_{C=1}^{\mu}, \alpha_{C=1}^{\gamma}$) is a vector of parameters of length $l$ that weights each hyper-feature monomial $Z_i$. The $\alpha$'s are adapted to maximize the joint data likelihood over all patches for $C = 1$ (using patches from the "same" image pairs) and $C = 0$ (from the "different" image pairs) within the training set. These ideas are illustrated in detail in Figure 6, where, for demonstration purposes, we let our covariates $\mathbf{Z} = [\mathbf{y^3}, \mathbf{y^2}, \mathbf{y}, \mathbf{1}]^{\mathbf{T}}$ be a polynomial function of the $y$ position.

### 4.3. Automatic Selection of Hyper-Features

In this section we describe the automatic determination of $\mathbf{Z}$. Recall that in our GLM model we assumed a linear relationship between $\mathbf{Z}$ and $\mu$. By ignoring the dispersion parameter, this allows us to use standard feature selection techniques, such as Least Angle Regression (LARS) [10], to choose a few (around 10) hyper-features from a large set of candidates. In order to use LARS (or most other feature selection methods) "out of the box", we use regression based on an $L2$ loss function. While this is not optimal for non-normal data, from experiments we have verified that it is a reasonable approximation for the feature selection step. LARS was then asked to choose the hyper-features $\mathbf{Z}$ from these candidates: (a) the x and y positions of $F^L$, (b) the intensity and contrast within $F^L$ and the average intensity of the entire object, (c) the average energy in each of the eight oriented filter channels, and (d) derived quantities from the above such as square, cubic, and cross terms as well as meaningful derived quantities such as the direction of the maximum edge energy. Once $\mathbf{Z}$ is set, we proceed as in Section 4.2.

Running an automatic feature selection technique on this large set of possible conditioning features gives us a principled method of reducing the complexity of our model. Reducing the complexity is important not only to speed up computation, but also to mitigate the risk of over-fitting to the training set. The top curve in Figure 5 shows results when $\mathbf{Z}$ includes the first 10 features found by LARS. Even with such a naive set of features to choose from, the performance of the system improves significantly.

## 5. Modeling Pairwise Relationships Between Patches

In Sections 3 and 4, we described our method for scoring a model image patch $F_j^L$ and its best match $F_j^R$ by modeling the distribution of their distance in appearance, $d_j$, conditioned on the match variable $C$. Furthermore, in Section 3.3, we described how to infer the saliency of the patch $F_j^L$ for matching based on these distributions. As we noted in that section, this works for picking the first patch, but is not optimal for picking subsequent patches: once we have already matched and recorded the score of the first patch, the amount of information gained from a nearby patch is likely to be small, because their scores are likely to be correlated. Intuitively, the next chosen patch would ideally be a highly salient patch whose information about $C$ is as independent as possible from the first patch. Similarly, the third patch should consider both the first and the second patches.

Let $F_{(k)}^L$ represent the $k$th patch picked for the cascade and let $F_{(1...n)}^L$ denote the first $n$ of these patches. Assume we have already picked patches $F_{(1...n)}^L$ and we wish to choose the next one, $F_{(n+1)}^L$, from the remaining set of $F_j^L$'s. We would like to pick the one that maximizes the *information gain* or the *conditional mutual information*:

$$I(D_{(n+1)}; C|D_{(1...n)}) = I(D_{(1...n+1)}; C) - I(D_{(1...n)}; C).$$

This quantity is difficult to estimate, due to the need to model the joint distribution of all $D_{(1...n)}$ patches. However, note that the information gain of a new feature is upper bounded by the information gain of that feature relative to any *single* feature that has already been chosen. That is,

$$I(D_{(n+1)}; C|D_{(1...n)}) \leq \min_{1 \leq i \leq n} I(D_{(n+1)}; C|D_{(i)}). \quad (8)$$

Thus, rather than maximizing the full information gain, Vidal-Naquet and Ullman proposed the following heuristic that maximizes this upper bound on the amount of "additional" information:

$$\arg\max_j \min_i I(D_j; C|D_{(i)}), \quad (9)$$

where $i$ varies over the already chosen patches, and $j$ varies over the remaining patches.

We use a related, but slightly different heuristic. When $D_j$ and $D_{(i)}$ are completely correlated (that is, $D_{(i)}$ predicts $D_j$) then $I(D_j; C|D_{(i)}) = 0$. However, even when $D_j$ and $D_{(i)}$ are completely independent given C, $I(D_j; C|D_{(i)})$ does not equal $I(D_j; C)$. This somewhat counterintuitive result is due to the fact that there is only a total of 1 bit of information in $C$, some of which has already been discovered by matching patch $F_j$. This property causes problems for the above pairwise approximation, as in some circumstances it might

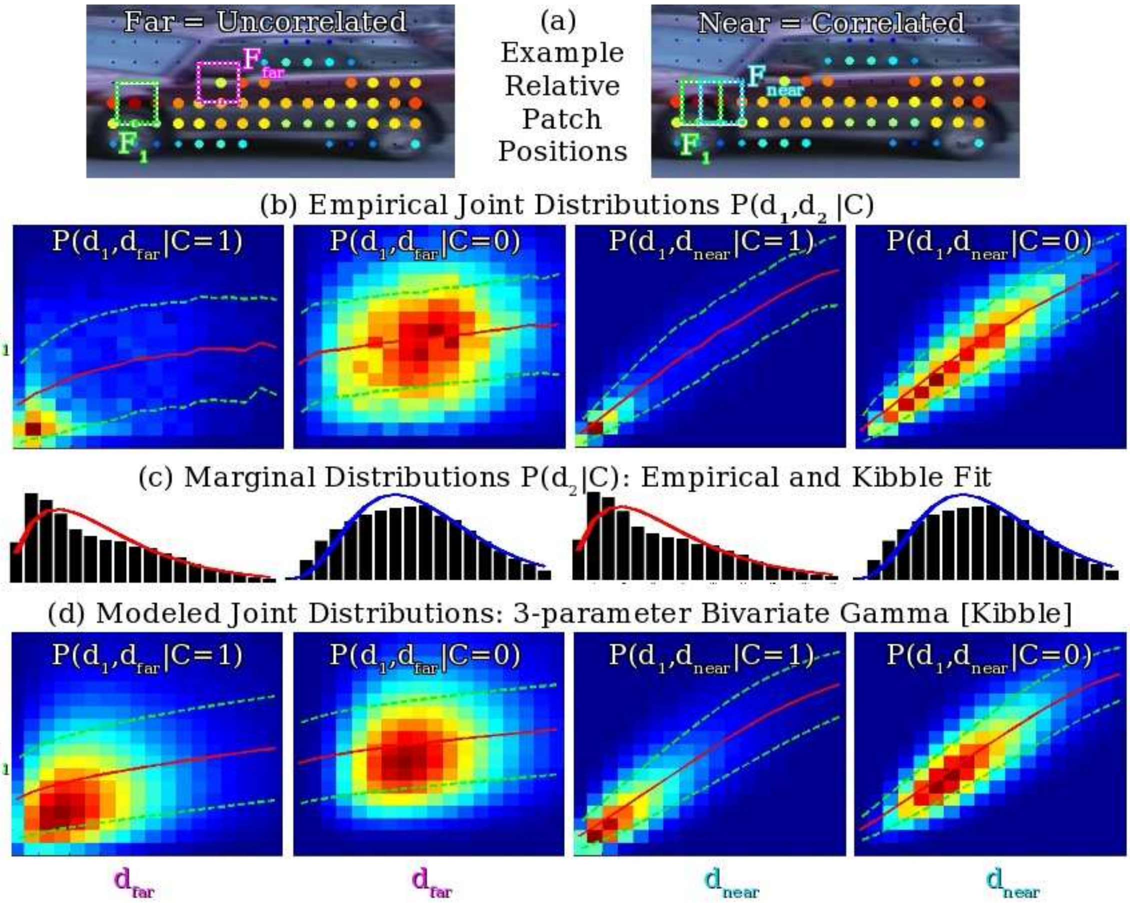

Figure 7: *Bivariate Gamma Distributions.* We demonstrate our technique by plotting the empirical and modeled joint densities of all patch pairs from the training set which are a fixed distance away from each other. On the **left** side, the two patches are far apart, thus they tend to be uncorrelated for both “same” ($C = 1$) and “different” ($C = 0$) pairs. This is evident from the empirical joint densities $d_1$ vs. $d_2$ (labeled $d_{far}$), computed by taking all pairs of “same” and “different” 25x25 pixel bi-patches from the training set that were more than 60 pixels apart. The great mismatch between the $P(d_1, d_{far}|C = 1)$ and $P(d_1, d_{far}|C = 0)$ distributions implies that the *joint mutual information* between $(d_1, d_{far})$ and $C$ is high. Furthermore, the mismatch in the joint distributions is *significantly larger* (as measured in bits) than the mismatch between the marginal conditional distributions shown below them in row (c). This means that the information gain, the joint mutual information less the marginal mutual information, is high. In contrast, the **right** side shows the case where the patches are very close (overlap 50% horizontally). Here $d_1$ vs. $d_2$ (labeled $d_{near}$) are very correlated. While there is still some disagreement between the joint distributions for $C = 0$ and $C = 1$, the information contained in this discrepancy (as measured in bits) is almost equal to the information contained in the discrepancy between the marginal distributions shown beneath them in row (c). That is, the joint distributions provide no additional information, or information gain, over the marginal distributions. Our parametric model for these joint densities are shown at the bottom (d). Notice that the modeled marginal distributions of $d_2$ (c) are gamma and are unaffected by the correlation parameter. The lines superimposed on the bivariate plots show the mean and variance of $d_1$ conditioned on $d_2$: notice that these are very similar for the empirical (b) and model (d) densities.

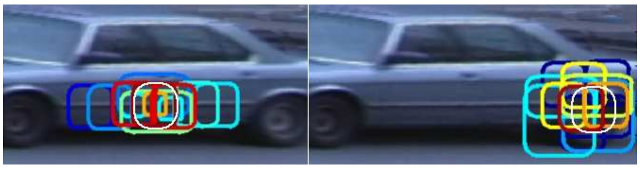

Figure 8: *Patch Correlations.* On each image, the patches most correlated with the white-circled patch are shown. Notice that in the left image, where the patch sits in an area with a highly visible horizontal structure, the most correlated patches all lie along the horizontal features. Contrast this with the right image, showing correlation of patches with a patch sitting on a wheel, where the most correlated patches are those that strictly overlap the white-circled patch.

lead to choosing a suboptimal next patch $F_{(i)}$ (a patch that is highly correlated with an uninformative patch might win out against another patch that is lightly correlated with a very informative one). Hence, in order to find the best next patch, we use a quantity related to $I(D_j; C|D_{(i)})$, but one which varies between 0 and $I(D_j; C)$ depending only on the correlation:

$$\arg\max_j \min_i I(D_j; C|D_{(i)}) \times \frac{I(D_j; C)}{I(D_j^*; C|D_{(i)})}. \tag{10}$$

Here $D_j^*$ is a random variable with the same marginal distribution as $D_j$ but is independent of $D_{(i)}$ when conditioned on $C$. This formulation also turns out to be easier to approximate within our framework (see Section 5.3).

## 5.1. Dependency Model

To compute (10), we need to estimate conditional mutual informations of the form

$$I(D_j; C|D_{(i)}) = I(D_j, D_{(i)}; C) - I(D_{(i)}; C).$$

In Section 3.3, we showed that we can determine the second term, $I(D_{(i)}; C)$, from the estimated gamma distributions for $P(D_{(i)}|C = 1)$ and $P(D_{(i)}|C = 0)$. Similarly, to calculate $I(D_j, D_{(i)}; C)$, we need to estimate the bivariate distributions $P(D_{(i)}, D_j|C = 1)$ and $P(D_{(i)}, D_j|C = 0)$.

Because there is relatively little data for each pair of patch locations, and because we want to evaluate the dependence of patches conditioned not only on location but on a variety of hyper-features, we again use a generalized linear model to gain statistical leverage, this time to model joint distributions of pairs of patch distances. The central goal in choosing a parameterization of the conditional joint distributions $P(D_{(i)}, D_j|C = 1)$ and $P(D_{(i)}, D_j|C = 0)$ is to choose a form for the distributions such that, when the parameters are estimated, the resulting computation of the joint mutual information is as accurate as possible. In order to achieve this, we adopt the following strategy for parametric estimates of the conditional joint distributions.

- We constrain each joint distribution to be an instance of Kibble's bivariate gamma distribution [15], a generalization of the one-dimensional gamma distribution that is constrained to have gamma distributions as marginals. A Kibble distribution has four parameters: $\mu_1, \mu_2, \gamma$, and $\rho$, with $0 < \rho < 1$. $\mu_1$ and $\mu_2$ are mean parameters for the marginals. $\gamma$ is a dispersion parameter for both marginals. $\rho$ is the correlation between $d_{(i)}$ and $d_j$, and varies from 0, indicating full independence of the marginals, to 1, in which the marginals are completely correlated (see Figure 7).
- We further constrain each distribution to have the same mean parameter for each marginal, i.e. $\mu_1 = \mu_2$ for each joint distribution. The shared mean parameter and the shared dispersion parameter $\gamma$ are set to the parameters of the marginal distribution $P(d_j|C = 0)$ and $P(d_j|C = 1)$ in the respective cases.
- Finally, we constrain the pair of distributions $P(D_{(i)}, D_j|C = 1)$ and $P(D_{(i)}, D_j|C = 0)$ to share the same correlation parameter $\rho$.

Thus we use Kibble's bivariate distribution with 3 parameters, which we write as $K(\mu, \gamma, \rho)$ (see Appendix B).

## 5.2. Predicting Patch Correlations from Hyper-Feature Differences

Given the above formulation, we have reduced the problem of finding the next best patch, $F^L_{(n+1)}$, to the problem of estimating the correlation parameter $\rho$ of Kibble's bivariate gamma distribution for any pair of patches $F^L_{(i)}$ (one of the $n$ patches already selected) and $F^L_j$ (a candidate for $F^L_{(n+1)}$). The intuition is that patches that are nearby and overlapping or that lie on the same underlying image features (for example the horizontal line on the side of the car in Figure 8) are likely to be highly correlated, whereas two patches that are of different sizes and far away from one another are likely to be less so.

We model $\rho$, the last parameter of $K(\mu_j^{C=1}, \gamma_j^{C=1}, \rho)$ and $K(\mu_j^{C=0}, \gamma_j^{C=0}, \rho)$, similarly to our GLM estimate of its other parameters (see Section 3.3): we let $\rho$ be a linear function of the *difference* of various hyper-features of the two patches, $F^L_{(i)}$ and $F^L_j$. Clear candidates for these covariates are the difference in position and size of the two patches, as well as some image-based features such as the difference in the amount of contrast within each patch. To ensure $0 < \rho < 1$, we use a *sigmoid* link function

$$\rho = (1 - \exp(\beta \cdot \mathbf{Y}))^{-1}, \tag{11}$$

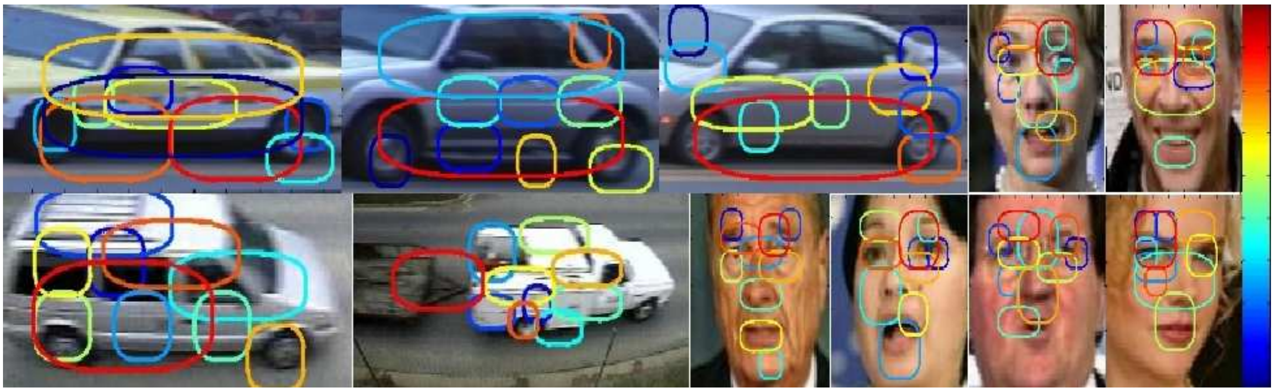

Figure 9: *The Ten Most Informative Patches.* The ten rectangles on each object show the top ten patches our classifier generator $\mathcal{H}_{id}$ selected for the classification cascade *for that object*. The face model seems to prefer features around the eyes, while the two car models (two data sets, top and bottom) tend to both like the side and wheels but differ in their interest in the roof region. Notice, however, that even within a category each cascade is unique, highlighting interesting appearance features for that object; this is because the patches are selected based on both position and appearance characteristics (hyper-features). The patches are color coded according to their cascade order, from most informative (red) to least (blue) (see color-bar on the right).

where $\mathbf{Y}$ is our vector of hyper-feature differences and $\beta$ is the GLM parameter vector.

Given a data set of patch pairs $F^L_{(i)}$ and $F^L_j$ and associated distances $d_{(i)}$ and $d_j$ (found by matching the "left" patches to a "right" image of the same or of a different object), we estimate the linear coefficients $\beta$. This is done by maximizing the likelihood of $K(\mu^{C=1}_j, \gamma^{C=1}_j, \rho)$ using data taken from image pairs that are known to be the "same"[2] and $K(\mu^{C=0}_j, \gamma^{C=0}_j, \rho)$ using data taken from "different" image pairs. Also similarly to Section 3.4, we choose the encoding of $\mathbf{Y}$ automatically, by the method of forward feature selection [13] over candidate hyper-feature difference variables. As anticipated, the top ranked variables encoded differences in position, size, contrast, and orientation energy. Our final model uses the top 10 variables.

### 5.3. Online Estimation of Patch Order

As we described in Section 5.1, we wish to select patches in a greedy fashion based on Eqn. 10. In the previous section, we have shown how to estimate $I(D_j; C|D_{(i)})$. Based on this, computing $I(D^*_j; C|D_{(i)})$ is straightforward: use the same Kibble densities as with $D_j$ but just set the correlation parameter $\rho = 0$.

Unfortunately, computing these quantities online is very expensive (notice that the formula for the Kibble distribution contains an infinite sum). However, we noticed that $k = \frac{I(D_j;C|D_{(i)})}{I(D^*_j;C|D_{(i)})}$, which varies from $0 < k < 1$, is well approximated by $k = (1 - \rho)$. Thus in practice, to find the next best patch, our algorithm finds the patch $j$ such that

$$\arg\max_j \min_i I(D_j; C) \times (1 - \rho_{j(i)}) \qquad (12)$$

where $\rho_{j(i)}$ is computed by Eqn. 11 from the hyper-feature differences between patch $F_j$ and $F_{(i)}$.

## 6. Building the Cascade

Now that we have a model for patch dependence, we can create a sequence of patches $F^L_j$ (see Section 3.3) that, when matched, collectively capture the maximum amount of information about the decision $C$ (same or different?). The sequence is ordered so that the first patch is the most informative, the second slightly less so and so on. The final step of creating a cascade is to define early stopping thresholds on the log likelihood ratio sum $R$ that can be applied after each patch in the sequence has been matched and its score added to $R$ (see Section 3.2).

We assume that we are given a global threshold $\lambda$ (see Section 3.2) that defines a global choice between selectivity and sensitivity. What remains is the definition of thresholds at each step, $\lambda^{C=1}_{(k)}$ and $\lambda^{C=0}_{(k)}$, which allow the system to accept (declare "same") if $R > \lambda^{C=1}_{(k)}$ or reject (declare "different") if $R \leq \lambda^{C=1}_{(k)}$, otherwise continue by matching patch $k + 1$. To learn these thresholds, we run $\mathcal{H}_{id}$ on the left images and the resulting classifier $\mathcal{C}_{id}$ on the right images of our training data set. This will produce a performance curve for each choice of $k$, the number of patches included in the classification score, including $k = m$, the sum for which $\lambda$ is defined. Our goal for the cascade is for it to make decisions as early as possible (tight thresholds) but, on the training set, never make a mistake on any pair which was correctly clas-

[2] $\mu^{C=1}_j$ and $\gamma^{C=1}_j$ are estimated from $F^L_j$ by the method of Section 3.4 and are fixed for this optimization.

sified using all $m$ patches and the threshold $\lambda$. These two constraints exactly define the thresholds $\lambda_{(k)}^{C=1}$ and $\lambda_{(k)}^{C=0}$:

1. For each "same" and "different" pair in the training set
   (a) generate the classifier $\mathcal{C}_{id}$ with a sequence of $m$ patches based on $I^L$
   (b) classify $I^R$ by evaluating

$$R = \sum_{j=1}^{m} \log \frac{P(D_j = d_j | C = 1)}{P(D_j = d_j | C = 0)} > \lambda$$

2. Let $I_{C=1}$ be the set of correctly classified "same" pairs (where label is "same" and $R > \lambda$). Set the rejection threshold $\lambda_{(k)}^{C=0}$ by

$$\lambda_{(k)}^{C=0} = \max_{I_{C=1}} \sum_{j=1}^{k} \log \frac{P(D_j = d_j | C = 1)}{P(D_j = d_j | C = 0)}$$

   That is, we want $\lambda_{(k)}^{C=0}$ to be the maximum without any "same" pairs that were correctly classified using all of the patches to be misclassified by this threshold.
3. Similarly define $I_{C=0}$, and set $\lambda_{(k)}^{C=1}$ using the $\min$.

# 7. Results and Conclusion

The goal of this work was to create an identification system that could be applied to different categories, where the algorithm would automatically learn (based on off-line training examples) how to select category-specific salient features from a new image. In this section, we demonstrate that after category training, our algorithm is in fact able take a single image of a novel object and solely based on it create a highly effective "same" vs. "different" classification cascade of image patches. Specifically, we wish to show that for visual identification each of the following leads to an improvement in performance in terms of accuracy and/or computational efficiency:

1. breaking the object up into patches (a.k.a parts, fragments), matching each one separately and combining the results,
2. differentiating patches by estimating a scoring and saliency function for each patch (based on its hyper-features),
3. modeling the dependency between patches to create a sequence of patches to be examined in order, and
4. applying early termination thresholds to the patch sequence to create the cascade.

We tested our algorithm on three different data sets: (1) cars from 2 cameras with significant pose differential, (2) faces from news photographs, and (3) cars from a wide-area tracking system with 33 cameras and 1000's of unique vehicles. Examples from these three data sets are shown in Figure 9, with the top 10 patches of the classification cascade. Notice that the sequence of patches for each object reflects both category knowledge (for cars, the $\mathcal{H}_{id}$ tends to select descriptive patches on the side with strong horizontal gradients and around the wheels, while for faces the eyes and eyebrows are preferred) and object specific characteristics (for example, note the focus on the unique trailer).

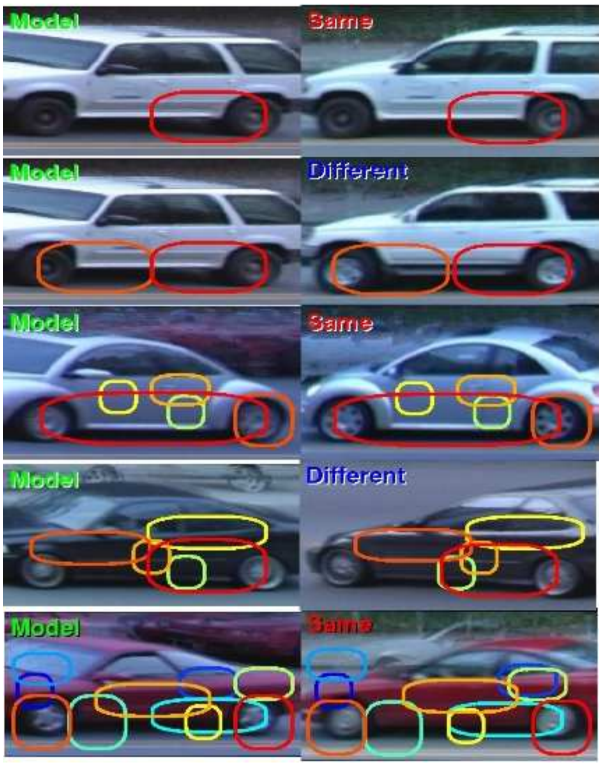

Figure 10: *Model-Test Car Image Pairs.* Each pair of images shows a model and a test image, which has been labeled as "same" or "different" (see upper left corner of test image) by our algorithm. The patches that were used in the cascade for that test image are indicated for each pair, where the order is color coded from red to blue. The first 3 rows show correct classification results, while the last 2 demonstrate errors. False-negative errors primarily occur with darker cars where the main source of features are the illumination artifacts that can vary greatly between the images. False-positive errors tend to involve very similar cars.

For each data set, a different automatic preprocessing step was applied to detect objects and approximately align them. After this, the same identification algorithm was applied to all three sets. For lack of space, we detail our experiments on data set 1, enumerate the results of data set 2, and only summarize our experience with data set 3. Qualitatively, our results on the three are consistent in showing that each of the above aspects of our system improves the performance, and that the overall system is both efficient and effective.

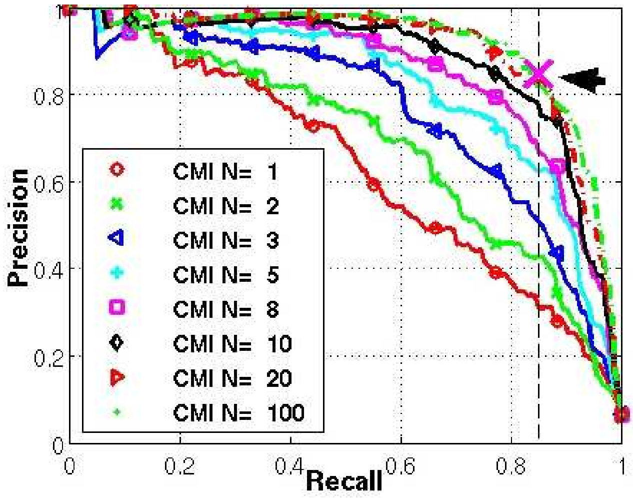

Figure 11: *Precision vs. Recall Using Different Numbers of Patches.* These are precision vs. recall curves for our full model. Each curve represents the performance tradeoff between precision and recall, when the system uses a fixed number of patches. The lowest curve uses only the single most informative patch, while the top curve uses up to 100 patches. The 85% recall rate, where the different models of Figure 12 are compared, is noted by a vertical black dashed line. A magenta X, at recall = 84.9 and precision = 84.8, marks the performance of the cascade model.

## 7.1. Cars 1

358 unique vehicles (179 training, 179 test) were extracted using a blob tracker from 1.5 hours of video from two cameras located one block apart. The pose of the cameras relative to the road (see Figure 1) was known from static camera calibration, and alignment included warping the sides of the vehicles to be approximately parallel to the image plane. Additionally, by detecting the wheels, we rescaled each each vehicle to be the same length (inter-wheel distance of 150 pixels). This last step actually hurts the performance of our system, as it throws away size as a cue (the camera calibration gives us a good estimate of actual size). However, we wanted to demonstrate the performance when such calibration information is not available (this is similar to our face data set, where each face has been normalized to a canonical size). Within training and testing sets, about 2685 pairs (true to false ratio of 1:15) of mismatched cars were formed from non-corresponding images, one from each camera. These included only those car pairs that were superficially similar in intensity and size. Using the best whole image comparison method we could find (normalized correlation on blurred filter outputs) on this set produces 14% false positives (29% precision) at a 15% miss rate (85% recall). Example correct and incorrect classification results using our cascade is shown in figure 10. This data set together with more example results are available from our web site.

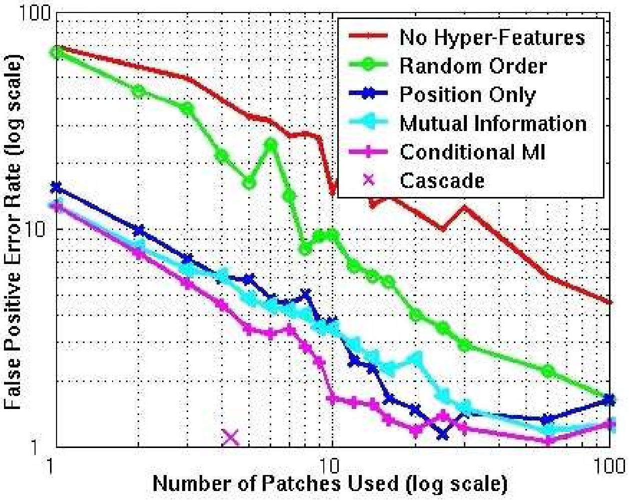

Figure 12: *Comparing Performance of Different Models.* The curves plot the performance of various models, as measured by the false-positive rate (fraction of different pairs labeled incorrectly as same), at a fixed recall rate of 85%. The $y$-axis shows the log error rate, while the $x$-axis plots the log number of patches the models were allowed to use (up to a max of 100). As the number of patches increases, the performance improves until a point, after which it levels off and, for the models that order patches according to information gain, even decreases (when non-informative patches begin to pollute the score). The (red) model that does *not use hyper-features* (i.e. uses the same distributions for all patches), performs very poorly compared to the hyper-feature versions, even when it is allowed to use 100 patches. The second curve from the top uses our hyper-feature model to score the patches, but *random selection* to pick the patch order. The *position only model* uses only position-based hyper-features for selecting patch order (i.e. it computes a fixed patch order for all cars). The light blue model sorts patches by *mutual information*, without considering dependencies. The last curve shows our full model based on selecting patches according to their *conditional mutual information*, using both positional and image-based hyper-features. Finally, the magenta X at 4.3 patches and 1.02% error shows the performance of the cascade model.

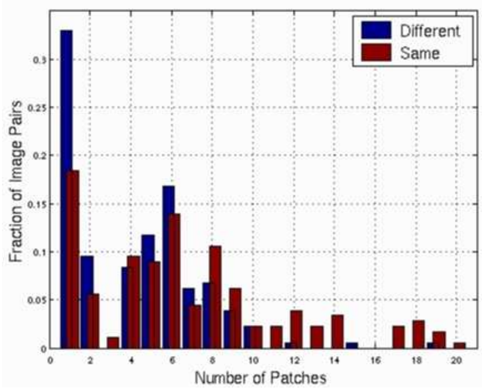

Figure 13: *How many patches does it take to make a decision?* This histogram shows the number of patches that were matched by the classification cascade before a decision could be made. On average, 4.2 patches were required to make a negative (declaring a difference) decision, and 6.7 patches to make a positive one.

Figure 12 compares several versions of our model by plotting the false-positive rate (y-axis) with a fixed miss rate of 15% (85% recall), for a fixed budget of patches (x-axis). The 85% recall point was selected based on Figure 11, by picking the equal error point given the 1 to 15 true-to-false ratio. The *Random Order* curve uses our hyper-feature model for scoring, but chooses the patches randomly. By comparing this curve to its neighbors, notice the performance gain associated with differentiating patches based on hyper-features both for scoring (*No Hyper-Features* vs. *Random Order*) and for patch selection (*Random Order* vs. *Mutual Information*). Comparing *Mutual Information* vs. *Conditional MI* shows that modeling patch dependence is important for choosing a small number of patches (see range 5-20) that together have high information content (Section 5). Comparing *Position Only* (which only uses positional hyper-features) vs. *Conditional MI* (which uses both positional and appearance hyper-features) shows that patch appearance characteristics are significant for both scoring and saliency estimation. Finally, the cascade performs (1.02% error, with mean of 4.3 patches used) as well as the full model and better than any of the others, even when these are given an unlimited computation budget.

Figure 11 shows another way to look at the performance of our full model given a fixed patch (computation) budget (the *Conditional MI* curve of Figure 12 represents the intersection of these curves with the 85% recall line). The cascade performance is also plotted here (follow the black arrow). The distribution of the number of patches it took to make a decision in the cascade model is plotted in Figure 13.

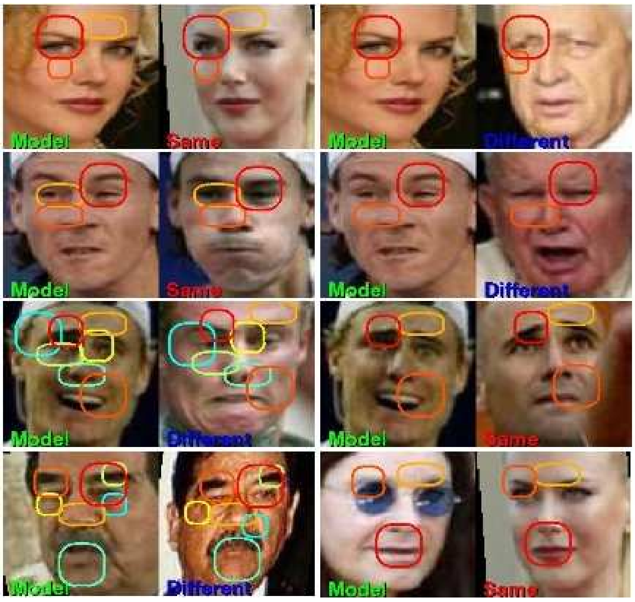

Figure 14: *Model-Test Face Image Pairs.* The first 2 rows of images show correct results, while the bottom 2 demonstrate errors. The large variations in pose, lighting, expression and image resolution make this data set very difficult. Our algorithm prefers eyes and seems to have learned that when the face is partially in profile, the eye that is more frontal is more informative (probably because it is more likely to be consistent). However, notice that the model for the person wearing sunglasses in the last row is the only one whose first patch in the cascade is not on the eye.

## 7.2. Faces

We used a subset of the "Faces in the News" data set described in [6], where the faces have been automatically detected from news photographs and registered by their algorithm. Our training and test sets each used 103 different people, with two images per person. This is an extremely difficult data set for any identification algorithm, as these face images were collected in a completely uncontrolled manner (news photographs). Table 1 summarizes our results for running the same algorithm as above on this set. Note the same pattern as above: the patch based system generally outperforms whole object systems (here we compare against state of the art PCA and LDA algorithms with face specific preprocessing using CSU's implementation [8]); estimating a scoring and saliency function through hyper-features greatly improves the performance of the patch based system; the cascades, using less than 6 patches on average, performs as well as always using the best 50 patches (performance actually declines above 50 patches). Refer to figure 14 for example classification results.

| Recall Rate | 60% | 70% | 80% | 90% |
|---|---|---|---|---|
| PCA + MahCosine | 82% | 73% | 62% | 59% |
| Filter + NormCor | 83% | 73% | 67% | 57% |
| No Hyper-Features | 86% | 73% | 68% | 62% |
| Random 10 Patches | 79% | 71% | 64% | 60% |
| Top 1 CMI Patch | 86% | 76% | 69% | 63% |
| Top 50 CMI Patches | 92% | 84% | 75% | 67% |
| **CMI Cascade** | **92%** | **84%** | **76%** | **66%** |

Table 1: *Precision vs. Recall for Faces.*
Each column denotes the precision associated with a given recall rate along the P-R curve. *PCA + MahCosine* and *Filter + NormCor* are whole face comparison techniques. *PCA + MahCosine* is the best curve produced by [8], which implements PCA and LDA algorithms with face-specific preprocessing. *Filter + NormCor* uses the same representation and comparison method as our patches, but applied to the whole face. The last four all use our patch based system with hyper-features. The last three use conditional mutual information based patch selection, where the number of patches allowed is set to 1, 50, and variable (cascade), respectively. These cascades use between 4 and 6 patches on average to make a decision.

### 7.3. Cars 2

We are helping to develop a wide-area car tracking system where this component must re-identify vehicles when they pass by a camera. Detection is performed by a blob tracker and the images are registered by aligning the centroid of the object mask (the cameras are located approximately perpendicular to the road). We tested our algorithm on a subset of data collected from 33 cameras and 1000's of unique vehicles, by learning an identifier generating function ($\mathcal{H}_i d$) for each camera pair (this way, the system incorporates the typical distortions that a vehicle undergoes between these cameras). Equal error rates for our classification cascade were 3-5% for near lane (vehicle length ~140 pixels) and 5-7% for far lane (~60 pixels), using 3-5 patches on average. Whole object comparison methods (we tested several different techniques) and using patches without hyper-features resulted in error rates that were 2 to 3 times as large. We estimate that an optimized implementation of our algorithm would be able to perform the vehicle identification component of this system (with up to 5 new vehicle reports per second, and 15 candidate ids per report) in real time on a single processor.

## Appendix A. Gamma Distribution

Gamma distributions are non-zero in the range $0 < x < \infty$ and have two degrees of freedom, most commonly parameterized as a shape parameter $\gamma$ and a scale parameter $\beta$. In this work, we typically use the parameters $\gamma$ and the mean $\mu$, where $\mu = \beta \times \gamma$. With this parameterization, the probability density function has the form

$$f(x;\mu,\gamma) = \frac{\gamma^{\gamma}(\frac{x}{\mu})^{(\gamma-1)}\exp(\frac{-x\,\gamma}{\mu})}{\mu\Gamma(\gamma)},$$

where $\Gamma()$ is the gamma function. For examples of gamma distributions, refer to Figures 3 and 6. In this paper we use the notation $\Gamma(\mu,\gamma)$ for the gamma distribution.

## Appendix B. Kibble's Bivariate Distribution

Kibble's bivariate gamma distribution is non-zero in the range $0 < x, y < \infty$ and has up to four degrees of freedom: the marginal parameters $\mu_x, \mu_y, \gamma$, and a correlation term $\rho$. Such a distribution has gamma marginals, where $\mu_x$ and $\gamma$ define the $x$ marginal and $\mu_y$ and $\gamma$ define the $y$ marginal. The parameter $\rho$ , which ranges $0 \leq \rho < 1$, is the correlation coefficient between the variables $x$ and $y$: when $\rho$ is small, $x$ and $y$ are close to independent; when $\rho$ is large, $x$ and $y$ are highly correlated. If we let $t_x = \frac{x\gamma}{\mu_x}$ and $t_y = \frac{y\gamma}{\mu_y}$, then this bivariate distribution has the form

$$\begin{aligned} f(x,y;\mu_x,\mu_y,\gamma,\rho) &= \frac{(t_x \times t_y)(\gamma-1)\exp(-\frac{t_x+t_y}{1-\rho})}{(1-\rho)^{\gamma}\Gamma(\gamma)} \\ &\times \sum_{j=0}^{\infty} \frac{\rho^j (t_x \times t_y)^j}{(1-\rho)^{2j}\Gamma(\,\gamma+j)j!}. \end{aligned}$$

The rate of convergence of the infinite series depends heavily on the $\rho$ parameter, where values of $\rho$ close to 1 converge much more slowly. Examples of Kibble's distribution can be found in Figure7(d). In this paper, we always set $\mu_x = \mu_y$, and thus denote Kibble's distribution as $K(\mu,\gamma,\rho)$.

# Acknowledgments

This work was supported by the DARPA CZTS project. We would like to thank Sarnoff Corporation for the wide area car tracking data set ("Cars 2"), especially Ying Sang and Harpreet Sawhney. From Berkeley, Ryan White provided the face data set and Hao Zhang an implementation for LARS.

# References

[1] Y. Amit and D. Geman. A computational model for visual selection. *Neural Computation*, 11(7), 1999.

[2] Y. Amit and M. Mascaro. An integrated network for invariant visual detection and recognition, tech. report no. 521. *Department of Statistics University of Chicago*, 2000.

[3] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projections. *IEEE Pattern Analysis and Machine Intelligence*, 19(7), 1997.

[4] S. Belongie, J. Malik, and J. Puzicha. Matching shapes. In *International Conference on Computer Vision*, 2001.

[5] A. Berg, T. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondence. *CVPR*, 2005.

[6] T. L. Berg, A. C. Berg, J. Edwards, M. Maire, R. White, Y. W. Teh, E. Learned-Miller, and D. A. Forsyth. Names and faces in the news. *CVPR*, 2004.

[7] V. Blanz, S. Romdhani, and T. Vetter. Face identification across different poses and illuminations with a 3d morphable model. *Proceedings of the 5th International Conference on Automatic Face and Gesture Recognition*, 2002.

[8] D. Bolme, R. Beveridge, M. Teixeira, and B. Draper. The csu face identification evaluation system: Its purpose, features and structure. *ICVS*, 2003.

[9] R. Diamond and S. Carey. Why faces are and are not special: An effect of expertise. *Journal of Experimental Psychology*, Gen(115):107–117, 1986.

[10] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

[11] L. Fei-Fei, R. Fergus, and P. Perona. A bayesian approach to unsupervised one-shot learning of object categories. In *International Conference on Computer Vision*, 2003.

[12] B. Heisele, T. Poggio, and M. Pontil. Face detection in still gray images, a.i. memo no. 521. *Massachusetts Institute of Technology Artificial Intelligence Lab*, May 2000.

[13] G. H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *International Conference on Machine Learning*, pages 121–129, 1994. Journal version in AIJ, available at http://citeseer.nj.nec.com/13663.html.

[14] T. Kadir and M. Brady. Scale, saliency and image description. *International Journal of Computer Vision*, 45(2):83–105, 2001.

[15] W. F. Kibble. A two-variate gamma type distribution. *Sankhya*, 5:137–150, 1941.

[16] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[17] P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Chapman and Hall, 1989.

[18] E. G. Miller, N. E. Matsakis, and P. A. Viola. Learning from one example through shared densities on transforms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages ?–?, 2000.

[19] G. Mori, S. Belongie, and J. Malik. Shape contexts enable efficient retrieval of similar shapes. *CVPR*, 2001.

[20] C. Schmid. Constructing models for content-based image retrieval. *CVPR*, 2001.

[21] H. Schneiderman and T. Kanade. A statistical approach to 3d object detection applied to faces and cars. *CVPR*, 2000.

[22] M. Tarr and I. Gauthier. FFA: A flexible fusiform area for subordinate-level visual processing automatized by expertise. *Nature Neuroscience*, 3(8):764–769, 2000.

[23] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cogntive Neuroscience*, 3(1):71–86, 1991.

[24] M. Vidal-Naquet and S. Ullman. Object recognition with informative features and linear classification. In *International Conference on Computer Vision*, 2003.

[25] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[26] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. *ECCV*, 2000.

[27] L. Wiskott, J. Fellous, N. Krger, and C. von der Malsburg. Face recognition by elastic bunch graph matching. *Proc. 7th Intern. Conf. on Computer Analysis of Images and Patterns*, 1997.

[28] W. Zhao, R. Chellappa, A. Rosenfeld, and P. Phillips. Face recognition: A literature survey. *ACM Computing Surveys*, 35(4):399–458, 2003.