Learning to Locate Informative Features for Visual Identification

DRAFT: IJCV special issue **www.eecs.berkeley.edu/Research/Projects/CS/vision/shape/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 is a specialized type of recognition in which the category (e.g. cars) is known and the goal is to recognize an object's exact identity (e.g. Bob's BMW). Two special challenges characterize object identification. First, inter-object variation is often small (many cars look alike) and may be dwarfed by illumination or pose changes. Second, there may be many different instances of the category but few or just one positive "training" examples per object instance. Because variation among object instances may be small, a solution must locate possibly subtle object-specific salient features, like a door handle, while avoiding distracting ones such as specular highlights. With just one training example per object instance, however, standard modeling and feature selection techniques cannot be used. We describe an on-line algorithm that takes one image from a known category and builds an efficient "same" versus "different" classification cascade by predicting the most discriminative features for that object instance. Our method not only estimates the saliency and scoring function for each candidate feature, but also models the dependency between features, building an ordered sequence of discriminative features specific to the given image. Learned stopping thresholds make the identifier 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. 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 photographed 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 (e.g. faces or cars) is known, 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. cars), as opposed to *categorization* where the algorithm must group together objects that belong to the same category but may be visually diverse [1, 12, 25, 32]. Identification is also distinct from *object localization*, where the goal is locating a specific object in scenes where distractors have little similarity to the target object [20].

These differences are more than semantic: the object identification problem poses different challenges than its

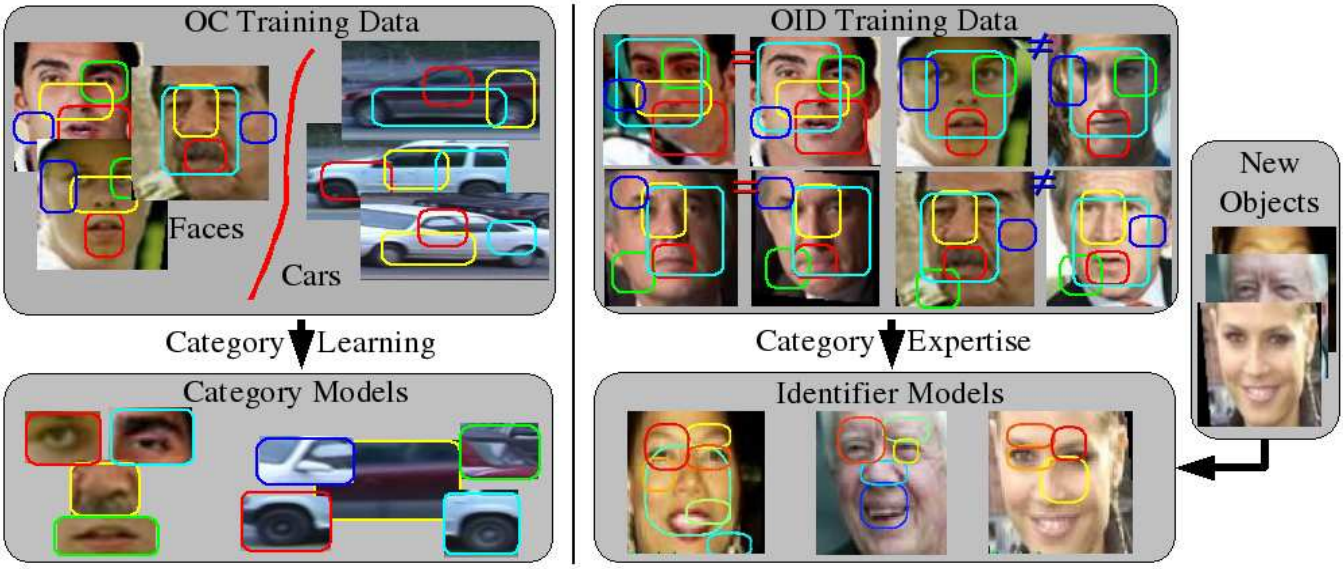

Figure 2: *Object Categorization (OC) versus Objection Identification (OID).* This figure highlights the different learning involved in categorization and identification. The training sets for object categorization, shown on the left side, typically contain many examples of each category (e.g. faces 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 object identification, on the other hand, contains pairs of images from a known category, with a label of "same" or "different" (denoted by $=$ and $\neq$ in the figure) for each pair. From these labeled pairs, the system must learn how to generate an object instance identifier given a single image of a new object (e.g., Mr. Carter) from the category. For these identifiers to work well, they should highlight distinctive regions of the object. That is, the identifiers should be different for each object.

coarser cousin, object categorization. Specifically, object *identification* problems are characterized by the following two properties.

1. The inter-instance variation is often small, and this variation is often dwarfed by illumination or pose changes (see Figure 1). For example, many cars look very similar, but the variability in appearance of a single vehicle, due to lighting for example, can be quite large.
2. There are many different instances of each category (many different individual cars), but few (in our case just one) positive "training" examples per object instance (e.g. only one image representing "Bob's BMW"). With only one example per instance, it is particularly challenging to build a classifier that identifies such an instance, which is precisely our goal.

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. [28] argued that the brain's fusiform face area does visual processing of categories for which expertise has been gained.

Categorization algorithms such as [1, 4, 31, 33] 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 object instances, 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.

1.1 The Three Steps of Object Identification

To clearly characterize the differences between object *categorization* and the main subject of this paper, object *identification*, we enumerate the key steps in each process. We

compare our object identification method with the traditional supervised learning paradigm of object categorization.

1.1.1 Object Categorization

In the simplest supervised learning framework for object *categorization*, the learner is supplied with two sets of examples: a set of positive examples that are in a category (like cars), and a set of negative examples that are not in the category. The standard procedure of categorization consists of two steps:

1. **Training** a classifier using examples labeled positive (in the category) and negative (not in the category), and

2. **Applying** the classifier to a new example to label it as positive or negative.

Theoretically, we could use the same scheme to do object *identification*. To recognize a particular individual, such as George Bush, we could collect sets of positive and negative examples, and use the traditional supervised learning method just described. As remarked previously, however, we would like to be able to identify an individual after seeing only a single picture. Traditional categorization schemes work very poorly when trained on only a single example. To address this lack of "training" data, we develop an entirely new scheme, based upon developing category expertise, and using this expertise to develop a customized identifier for each instance we wish to recognize.

1.1.2 Object Identification

In our new scheme, there are three steps rather than two. In the first step, performed off-line on training data, we develop expertise about the general category, such as faces. This is done by comparing the corresponding patches of many example pairs, some that match and some that do not. The goal is to analyze patch differences for each type of image pair, matching and non-matching, to understand under what conditions, we expect patches to be similar or different.

The expectation of the degree of differences between corresponding patches can depend upon many factors. Patches that are not covering the face should not match well even if it is the same person, while patches from the eye area are likely to match well if the images are of the same person, but not for different people. On the other hand, patches from the cheek area may match well even when the images are not of the same person (due to the frequent lack of texture in this region). Finally, forehead images from the same person are likely to match if there is no hair in the patch, but may not match well if there is hair, since hair is highly variable from appearance to appearance. These intuitions translate into a "scoring function" that relates the appearance similarity of individual matched patches to an indication of equivalence of the whole face.

In addition to modeling the appearance differences among patches conditioned on the type, location and appearance of the patches, we can also estimate the **expected utility** or **discriminativeness** of a single patch from this analysis. We rate the discriminativeness of a patch by considering whether the expected differences with a corresponding patch depend heavily on whether the faces match or not. For example, since a pair of corresponding patches which do not cover a portion of the face are expected to have large differences, *irrespective of whether the faces match or not*, such patches have low expected utility. On the other hand, patches near the eye region are expected to have small differences when the faces match, but to have larger differences when the faces do not match, and hence are expected to have high utility. In summary, the first step of our procedure produces models of patch differences conditioned on a variety of variables, and also allows us to assess the expected utility of a given patch based upon the patch difference models.

In the second step, which occurs at "test time", we use this expertise to build an identifier (more specifically an identification *cascade*) for an object instance given a single image of that object. For each patch in the given image, we select a specific model of patch appearance differences from the global model of patch appearance differences defined in the first step. Then, using these models of patch appearance differences for each patch, we analyze the discriminativeness of each patch in the given image. Finally, we sort the patches in order from most discriminative to least discriminative.

In the third step, we use the object specific identifier to decide whether other instances of the category are the same as or different than the given instance. This is done by comparing patches in the given image (the "model" image) to corresponding patches in another image (a "database" or "test" image). The most discriminative patches (as ordered in the second step) are used first and followed, if still necessary, by those with less discriminative power.

In summary, we define the steps in object identification as:

1. **Learning a global model of patch**[1] **differences**, and as a result, a model of patch discriminativeness,

2. **Building an identification cascade for a specific object instance** by selecting, from the global model, object-specific models of patch differences and sorting the patches by discriminativeness, and

[1] To simplify the exposition, we have described the process of learning category expertise in terms of learning a model of patch discriminativeness, which is how the identifiers in this paper were built. However, it is straightforward to generalize this general scheme to encode category information in a way other than modeling patches–for example, by modeling the distributions of colors in images.

3. **Applying the identification cascade** to novel images to assess whether they are the same or different as the specific object instance for which the cascade was built.

We note that the last step of object categorization and the last step of our scheme for object identification are essentially the same. In the case of object categorization, we apply a categorizer to a new example to decide whether it represents a category of interest. In the case of object identification, we apply an identifier to a new example to decide whether it is the same object as the single given training example. Other than this last step, however, the two paradigms are quite different.

While the traditional object categorization scheme encodes information at only one level, the level of the object category, the identification scheme encodes two types of information. In the first step of object identification, we encode information about the entire category. In the second stage, we encode information about the specific object instance. **It is the use of the category expertise learned in the first step that enables us to build effective identifiers from just a single example of each object.** Without the category level information, it would be impossible to tell how to weight various areas of the image from only a single example. Our key contribution is a novel method for encoding the category information so that we can use it effectively to define an identifier for a new object instance.

1.1.3 Hyper-Features

As stated above, in step 1 of our object identification process, we analyze corresponding patches in image pairs to develop a global model of which types of patches are likely to be useful and which are not, and for the patches that are useful, to know how to score a correspondence. This model needs to generalize to whole space of possible patches so that at test time we can predict the utility and build a scoring function for patches of new instances (e.g. new car models) that were not in the training set.

Given that we register objects before analyzing patches, it should not be surprising that patches from certain parts of the image tend to be more informative than patches from other parts of an image. For example, when faces are registered to a canonical position by centering them in an image, the patches in the upper corners of the image tend to represent background, and hence are unlikely to be useful for discrimination. Hence, the spatial location of a patch is useful in predicting its future utility in discrimination, even when we do not yet know the patch we might be comparing it against. This type of "conditioning" on spatial location to predict utility is common in computer vision.

In this paper, however, we introduce a novel method for predicting the utility of image patches that goes beyond merely conditioning on spatial location. In particular, we also use *appearance* features of a patch to predict its utility. For example, a patch of someone's cheek that is uniformly colored is not likely to be very discriminative since there is a strong possibility the corresponding patch on another different person's face would be indistinguishable, causing this patch to be of no value in discrimination. However, if the patch on someone's cheek shows a distinctive mole or scar, its predicted utility dramatically increases, since this feature would be unlikely to be repeated unless we are comparing against the same person.

Conditioning on visual features to predict utility, in addition to spatial location, gives our patch discriminativeness models much more power. We call these features on which we condition *hyper-features*, and their use to model image differences is the main contribution of this work.

The remainder of the paper is organized as follows. Section 2 discusses previous work in a number of areas. Section 3 summarizes the three stages of our algorithm: learning class expertise in training, building an identification cascade for a specific example, and running the identifier. 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 experiments on multiple car and face data sets.

2. Comparison to Previous Work

In this section, we highlight relevant previous papers and describe how our method differs or improves on them.

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) [4, 6, 10, 15, 18, 20, 24, 31, 32, 33, 34]. 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 classifier then combines this evidence, optionally using the positional configuration of the detected parts as an additional cue, to determine the presence or absence of the object.

Due to the constraints of object identification described in the introduction, our system differs from previous work in a fundamental way. In the above systems, a model consisting of informative parts and (optionally) their relationships

is learned from a set of object and background examples. This "feature selection" step is fundamental to these methods and is possible because statistics such as the frequency of appearance of a particular feature (e.g., a quantized SIFT vector) can be *directly* computed for the positive and negative examples of a class. Thus these systems rely on underlying feature selection (and weighting) techniques such as Conditional Mutual Information [13] or AdaBoost [14]. In our setting this is not possible because only one example of a particular category instance (which plays the role of a "class" in our setting) will be presented, which is not enough to directly estimate the discriminativeness of any feature. Our main contribution is overcoming this fundamental barrier by learning a *model for the space of all possible features* tuned to a particular category (e.g., cars) that then allows us to pick the discriminative features for the given category instance (e.g., Bob' BMW).

A minor additional difference compared to many of the the above techniques is the choice of part representation. Popular encodings such as SIFT [20], which are designed to be invariant to significant distortions, are too coarse for our needs – they often destroy the very information that distinguishes one instance from another. Thus we use a more dense representation of multiple filter channels. However, we stress that this is not fundamental to our method, and any part encoding and comparison metric could be used within our learning framework.

2.2. Interclass Transfer

Because of the lack of training data for a particular instance, the general framework of most object recognition systems, that of selecting informative features using a set of object/non-object examples, is impossible to directly apply to our setting. In view of this difficulty, given a new category instance (e.g., Bob's BMW), how can we pick good features, and how can we combine them to build an object instance identifier?

One possible solution is to try to pick universally good features, such as corners [18, 20], for detecting salient points. However, such features are not category specific: we expect to use different kinds of image features when distinguishing Bob's car from other cars versus when we are distinguishing Bob's dog from another dog.

Another possibility is to build generative models for each class including such characteristics as the typical illuminations, likely deformations, and 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, good features could be explicitly coded into the algorithm [34]. However, this tends to be complicated and time consuming, and must be done individually for a particular category (see Section 2.4 below for examples). Additionally, one might hope that given a good statistical model and a large training data set, an algorithm would actually be better at finding informative features.

A better option is to attempt to transfer models from previous classification tasks of the same category (*interclass transfer*). Thrun, in [29], introduces such an interclass transfer (also referred to as lifelong learning or learning-to-learn) framework, in which a series of similar learning tasks are presented, where each subsequent task uses the model that was built for the previous tasks. More recently [22], distributions over parameters of a similarity transformation learned from one group of classes (letters) are used to model other classes (digits) for which only a single example is provided. In other work [12], priors for a fixed-degree constellation model 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 all of these works, the set of hidden variables (the features used by [29], the transformations in [22], or the parameters of the constellation model in [12]) are predefined and the generalization from other categories can be thought of as learning priors for these fixed sets of variables. In contrast, we wish to learn how to identify any number of good features that are present in the single "training" example for an object instance that can then be assembled into a binary classification cascade. This forces us to learn a model for the space of all possible features (in our case, image patches).

2.3. Pairwise Constraints

The machine learning literature provides a different perspective on our interclass transfer problem. Our problem can be thought of as a standard learning task where the input is a pair of images, and the output is a "same" vs. "different" label. The task is then to learn a "distance metric" between images, specifically by choosing and weighting relevant features.

Recent work on equivalence constraints such as Relevant Component Analysis [27] and others [26, 35] show how to optimize a distance metric over the input space of features that maps the "same" pairs close to one another while keeping "different" ones apart. In our setting, the transformations that we would be interested in are subset selection and weighting (although our technique does more then weight each feature). These methods, however, assume that each example is described by the same predefined set of features, and that the comparison function is a specific distance metric over these features (e.g., Euclidean).

In our case, the "features" we are trying to use are subpatches of one image, compared to the best corresponding location in the other image. Thus our feature space is very high dimensional, and the comparison method is not a simple distance metric (notice, for example, that it is not symmetric due to the local search of best corresponding patch). Even if this space of features were discretized, it would be impossi-

ble to enumerate all possible such features, and most would never appear within the training set. These differences make our algorithm very different from other pairwise constraint techniques.

A core observation of this paper is that it is not necessary to enumerate all possible features. Instead, we can model the space of the features in a way that allows us to estimate the informativeness of novel features that the algorithm was not directly trained on (informative means that when this patch is compared to the best corresponding patch in a test image, the appearance similarity gives us information about the test image being the "same" or "different"). Thus we model this space of features (in our case, each feature is defined by the size, position and appearance of the image patch) using a smooth function (actually a pair of them, one based on the matching to the "same" cars and one based on "different" pairs). Then, given a new instance, the algorithm can select the most informative patches among the ones that are actually present in that image. Furthermore, our pair of functions gives us a way to convert the patch matching distance to a score for each selected patch (this is similar to but has more degrees of freedom then a linear feature weight).

Here our features are image patches. We note, however, that our technique could be used in any setting where RCA is used when the features can be embedded into a continuous space. This has the potential advantage of exploiting the relationship between the features that the above techniques have no access to.

2.4. 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 [36].

Techniques such as Eigenfaces [30] (PCA), Fisherfaces [2] (LDA), and Bayesian face recognition [23], like our method, start with a general face modeling step, and later make a same/difference determination for a new face. Bayesian face recognition, which won the 1996 FERET competition, explicitly uses "same" and "different" equivalence constraints similar to the techniques described in Section 2.3. These are all "holistic" techniques 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 LDA and Bayesian techniques) and find a lower dimensional subspace that, it is hoped, is more conducive to identification. This is analogous to step 1 in our procedure. 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 [34], where hand selected fiducial points are matched within a graph that defines their relative positions, and the method of Blanz and Vetter [7], which maps images onto a 3D morphable face model.

We now turn to a more detailed description of our method.

3. Algorithm Overview

In this section, we outline the basic components of our system. After discussing initial preprocessing and alignment, we describe the three stages of our algorithm: global modeling of patch differences, building an identification cascade from one example, and application of the identification cascade. For clarity of exposition, we describe these stages in *reverse* order.

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 (in our data sets 10-20% of image size) around the expected location. No foreground-background segmentation is required, as the system learns which features within the images (both in terms of position and appearance) are useful for identification - thus patches that are off of the object are rejected by the learning algorithm. The specific detection and alignment methods used for our various data sets are described in Section 7. For example, for the Cars 2 data set, objects were aligned based only on the centroid of a simple background subtraction based blob detector.

3.2. Applying the Object Instance Identifier

We now describe the object instance identifier, which is the final step in our three step identification system. We start by introducing some notation.

We assume that at test time, we are given a single image of an object instance, known as the *model image*. The goal will be to compare it to a large set of database images, known as *test images*. For each pair of images (the model image and one of the test images) we wish to make a determination about whether the images represent the same specific object, or two different objects. The variable C will represent this match/mismatch variable, with $C = 1$ denoting that the two images are of the same object (i.e., they match) and $C = 0$ denoting that the two images are of different objects (i.e., they do not match).

For the purposes of illustration, we will often present pairs of images side by side, where the image on the **left** will be the model image and the image on the **right** will be one of the test images (Figures 3, 4, 10, 14). Thus we use I^L to refer to the model image and I^R to refer to the current test image. Thus, the identifier for a particular object instance decides if a test image (or "right" image) I^R is the same as ($C = 1$) or different than ($C = 0$) the model image (or "left" image) I^L it was trained for.

3.2.1. Patches

Our strategy is to break up the whole image comparison problem into the comparison of patches [31, 33]. The m (possibly overlapping) patches in the left image will be denoted F_j^L, with $1 \leq j \leq m$. The corresponding patches in the right image are denoted F_j^R.

Although the exact choice of features, their encoding and the comparison metric are not crucial to our technique, 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 large and small details of objects.

For our experiments, our patch sizes ranges from 12x12 pixels to the size of the full image, and are not constrained to be square. To compute the patch features, 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 (for our experiments, all of our patches, except the smallest ones taken from the lowest level of the pyramid, contained between 500 and 750 pixels). 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. The patch F_j^L is defined by its appearance encoding, position (x, y) and size (w, h).

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 evaluate the normalized correlation and compute

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

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

As the two images to be compared have been processed to be in rough alignment, we need only 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 [20] and shape contexts [3]. 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 insensitive to small differences, did not perform well. Specifically, using SIFT features as described in [20] (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 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 described elsewhere [33].

3.2.3. Likelihood Ratio Score

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

$$R = \frac{P(C=1|I^L, I^R)}{P(C=0|I^L, I^R)} \tag{2}$$

$$= \frac{P(I^L, I^R|C=1)P(C=1)}{P(I^L, I^R|C=0)P(C=0)} > \lambda. \tag{3}$$

where λ is chosen to balance the cost of the two types of decision errors. The prior probability of C is assumed to be known.[2] 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

$$P(C|I^L, I^R) \approx P(C|F_1, ..., F_m) \tag{4}$$

$$\propto P(F_1, ..., F_m|C). \tag{5}$$

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,

$$R = \frac{P(I^L, I^R|C=1)}{P(I^L, I^R|C=0)} \tag{6}$$

$$\approx \frac{P(F_1, ..., F_m|C=1)}{P(F_1, ..., F_m|C=0)} \tag{7}$$

$$= \prod_{j=1}^{m} \frac{P(F_j|C=1)}{P(F_j|C=0)}. \tag{8}$$

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

In practice, we compute the logarithm of this likelihood ratio, where each patch contributes an additive term. 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. Thus, to convert d_j to a score, the object instance identifier must consist of probability distribution functions $P(D_j|C = 1)$ and $P(D_j|C = 0)$ for each patch in the model image. These functions encode our expectations about how well we expect a patch in the test image to match a particular patch (j) in the model image, depending upon whether or not the images themselves represent the same object instance.

The object instance identifier computes the log likelihood ratio by evaluating these functions for each d_j obtained by comparing the model image patches to test image patches. (A comment on notation: d_j refers to the specific measured distance for a given model image patch and the corresponding test image patch, 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:

$$R = \sum_{j=1}^{m} \log \frac{P(D_j = d_j|C = 1)}{P(D_j = d_j|C = 0)}. \quad (9)$$

To compute this quantity, 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 $\theta_j^{C=0}$ for each patch and matching condition are defined as part of the object instance identifier. How we set these parameters using a single image is discussed in Section 3.3.

3.2.4. Making a Decision

The object instance identifier described above compared a fixed number of patches (m), computed the score R by Eq. 9, and compared it to a threshold λ. $R > \lambda$ means that I^L and I^R are declared to be the same. Otherwise they are declared different. In Section 6, we define a more efficient object instance identifier by building a *cascade* from the sequence of patches. This is done by applying *early termination thresholds* $\lambda_k^{C=1}$ (for early match detection) or $\lambda_k^{C=0}$ (for early mismatch detection) after the first k patches have been compared. These thresholds may allow the object instance identifier to stop and declare a result after comparing only k patches.

3.2.5. Summary of the Object Instance Identifier

To summarize, the object instance identifier 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 set 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. Generating an Object Instance Identifier

Stepping back one step in the process, we now describe how an object instance identifier is built, or "generated", from a single image of an object instance. Obviously, an identifier must be generated before it can be used to identify.

The identifier generator 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. Estimating $P(D_j|C)$

Estimating $P(D_j|C = 0)$ and $P(D_j|C = 1)$ means estimating parameters for two gamma distributions for each patch, such as the ones shown in Figure 3. Conceptually, we want $\theta_j^{C=0}$ and $\theta_j^{C=1}$ 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 estimated in the initial training phase (step 1), and how they are estimated is discussed at length below.

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 do not 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:

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

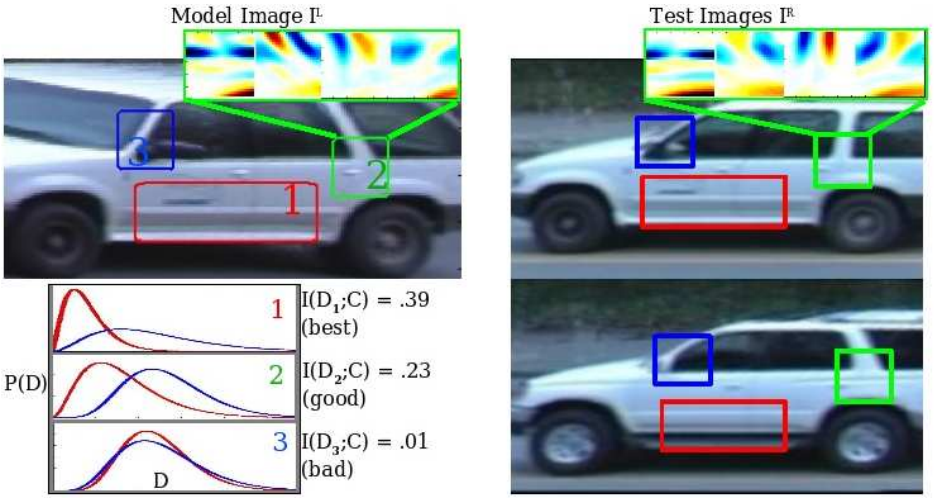

Figure 3: *The Object Instance Identifier.* On the left, a model image I^L is shown with an identifier composed of three patches (these would not be the actual top three patches selected by our system. To build the identifier, the three patches were analyzed to select three "same" and "different" distributions from the global model of patch differences. The red curve in each plot represents the "same" distribution, while the blue represents the "different" distribution. Our patch encoding using oriented filter channels is shown for patch 2. The object instance identifier matches the patches to the test images, computes the log likelihood ratio score for each using the estimated distributions, and makes a same versus 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 matching patch in the other "different" image (bottom); matching patch 3 should be much less informative, as both matching test image patches look highly dissimilar to the corresponding model patch. The superiority of patch 1 was inferred correctly by our system based on the position and appearance of these patches in the model image only, as shown by the mutual informations $I(D_j|C)$ for each patch, which are functions only of the model image patches.

Here $H()$ is Shannon entropy. Notice 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 ever 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 than 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 an object instance identifier from a single image.

3.4. Off-Line Training

Finally, we complete our reverse-order discussion by describing the first major step of our system, learning about a given category (e.g., cars) from training data. This procedure is done only once, and happens prior to testing.

The off-line training procedure defines the two functions $Q^{C=1}$ and $Q^{C=0}$ that estimate the parameters of the gamma distributions $P(D_j|C = 1)$ and $P(D_j|C = 0)$ from the position and appearance of a model image patch F_j^L. Additionally, it builds a dependency model among patches and defines the early termination cascade thresholds $\lambda_k^{C=1}$ and $\lambda_k^{C=0}$.

This off-line training starts with 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. Each patch is compared to its

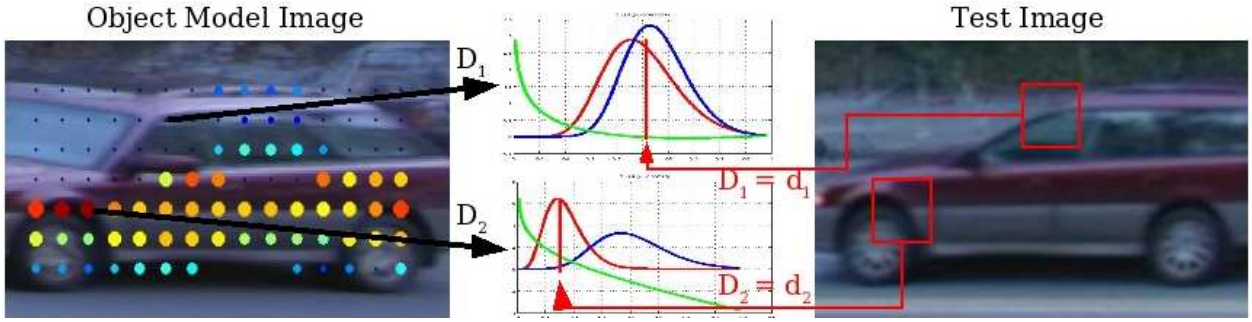

Figure 4: *Estimating the Distributions and Informativeness of Patches.* The identifier generator takes an object model image (left), samples patches from it, estimates the same and different distributions and mutual information score for each patch, and selects the sequence of patches to use for identification. The gamma distributions (middle) 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.

corresponding patch in the right image. The correspondence is defined by finding the best matching patch over a small search area around the same location in the second image. Once the corresponding patch is found in the right image, the resulting value of d_j is recorded and associated with the original patch from the left image.

The functions $Q^{C=0}$ and $Q^{C=1}$ will ultimately be polynomial functions of the hyper-features, that is, the location and appearance features of each patch. These polynomials are estimated in a maximum likelihood framework using a generalized linear model. In short, the functions $Q^{C=1}$ and $Q^{C=0}$ are optimized to produce gamma distributions which maximize the likelihoods $P(d_j|C)$ of the patch difference data from training. The details of this estimation are discussed in the following section.

4. Hyper-Features and Generalized Linear Models

In this section, we describe in detail how to estimate, from training data, the functions $Q^{C=0}$ and $Q^{C=1}$ that map the position and appearance of a model image patch F_j^L to the parameters θ_j^C of the gamma distributions for $P(D_j|C)$.

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. 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[3] is very large, the chance of having seen a patch very similar to F_j^L in the training set is small. In the next two subsections 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, i.e., the *hyper-features*.

4.1. Discrete Hyper-Features

First we explore a binning approach, where we place 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, the $y-$coordinate into three bins, and an appearance feature of the patch, such as contrast, into two bins. We would then label each patch with its position in this 4-by-3-by-2 histogram. 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 (θ_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 $\theta_j^{C=0}$.[4]

Figure 5 compares the performance of various models on the *Cars 1* data set, which is described in Section 7.1. Here, for simplicity of comparison, we use no patch selection (105 patches are sampled at fixed, equally spaced locations) and patch sizes are fixed to 25x25. The two bottom curves are

[3] For a 25x25 patch, appearance plus position (including size) is a point in $\Re^{25*25+4}$.

[4] Using the same binning of hyper-features, but modeling the resulting conditional distributions as normalized histograms, rather than gamma distributions, produces very similar results when enough data is available in each bin.

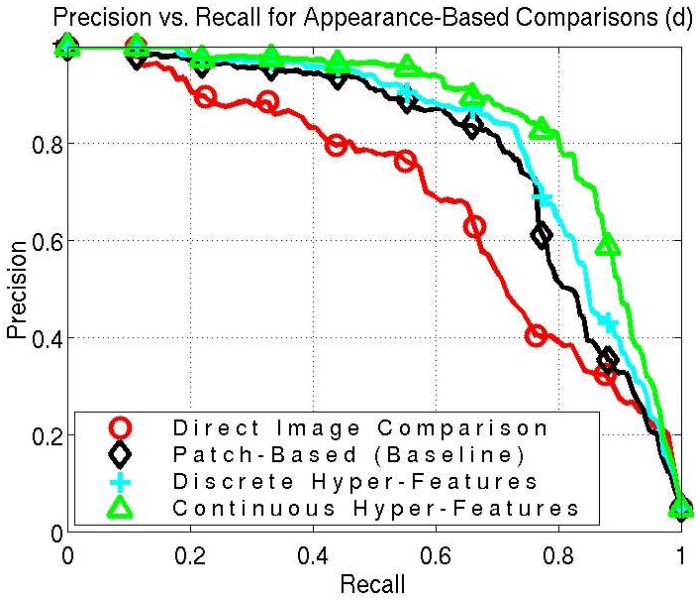

Figure 5: *Identification with Patches.* The bottom curve shows the precision vs. recall for non-patch based direct comparison of rectified images (the most accurate technique we found was to match a fixed subrectange of the image by searching for the best normalized correlation of the 4 filter channels). 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) using hyper-features in discrete bins Section 4.1 (Discrete), and (3) using a generalized linear model 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. Note: this figure differs from results presented later in that no patch selection was performed.

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.

The cyan "Discrete" curve in Figure 5 shows the performance improvement from conditioning on discrete hyper-features.

4.2. Continuous Hyper-Features

When too many hyper-feature bins are introduced, the performance of the discrete model 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 polynomial model to describe how hyper-features influence the choice of gamma distribution parameters.

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 θ with respect to the hyper-features can be modeled using a generalized linear model (GLM). Ordinary (least-squares) linear models assume that the data for each conditional distribution 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 [21] 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 monomials of those hyper-features, like squares, cubes, cross terms, or simply copies of the variables themselves. Then each bi-patch

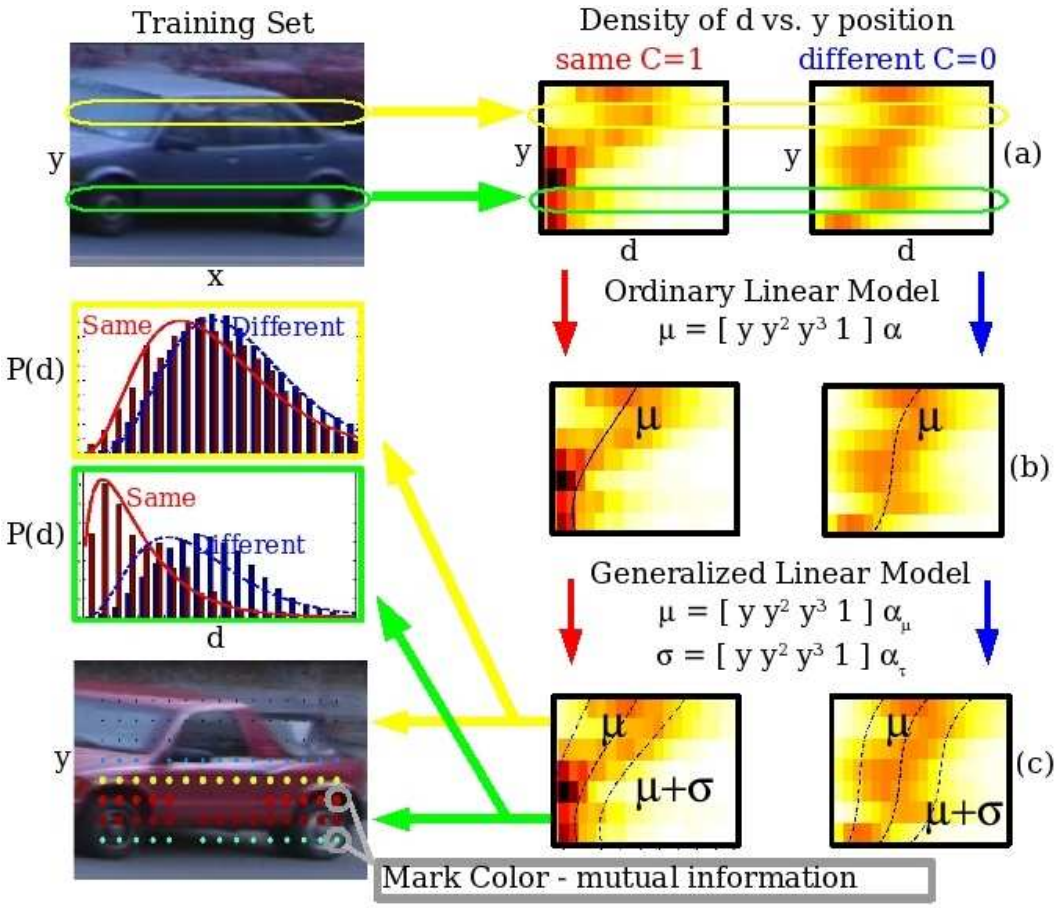

Figure 6: *Fitting a generalized linear model to the gamma distribution.* We demonstrate our approach by fitting a gamma distribution, through the latent variables $\theta = (\mu, \sigma)$, to the y position of the patches (in practice, we use the parameterization $\theta = (\mu, \gamma)$). Here we allowed μ and σ 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 two of these: one for bi-patches taken from matching vehicles (the pairs labeled "same"); the other from mismatched data ("different" pairs). Row **(b)** shows 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 two selected rows, each representing a particular y position, 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. That is, D_j will have low mutual information with C. In contrast, the bottom area is much more informative.

distance distribution has the form

$$P(d|X_1, X_2, ..., X_k, C) = \Gamma(d;\ \alpha^{\mu}_{\mathbf{C}} \cdot \mathbf{Z},\ \alpha^{\gamma}_{\mathbf{C}} \cdot \mathbf{Z}), \quad (10)$$

where the second and third arguments to $\Gamma()$ are mean and shape parameters. Note that both the mean and shape parameters are linear functions of the hyper-feature monomials $\mathbf{Z}$, which is what makes this model a generalized *linear* model.

For our GLM, we use the identity link function[5] for both μ and γ. While the identity is not the canonical link function for μ, 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 ($\mu = (\alpha^{\mu}_{C} * \mathbf{Z})^{-1}$), but observed no noticeable change in performance on our data set. Each α (there are four of these: $\alpha^{\mu}_{C=0}, \alpha^{\gamma}_{C=0}, \alpha^{\mu}_{C=1}, \alpha^{\gamma}_{C=1}$) is a vector of parameters of length l that weights each hyper-feature monomial Z_i. The α's are adapted to maximize the joint data likelihood over all patches for $C = 1$ (using patches from the "same" image pairs) and for $C = 0$ (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

While it is certainly possible to select the basic hyper-features X and their monomials Z manually, we make additional improvements to our system by considering larger potential sets of hyper-feature monomials and using feature selection techniques to select only those that are most useful.

Recall that in our GLM model we assumed a linear relationship between $\mathbf{Z}$ and μ. By ignoring the dispersion parameter, this allows us to use standard feature selection techniques, such as Least Angle Regression (LARS) [11], 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.

To use LARS for feature selection, we start with a large set of candidate hyper-feature monomials: (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 four 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. LARS is used to selected a subset of these, which act as the final set of hyper-features Z. Once $\mathbf{Z}$ is set, we proceed as in Section 4.2.

[5] "Link function" and "canonical link function" are terms related to generalized linear models. The reader should refer to GLM references for discussions of these terms [21].

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. We believe that further improvement in our results is possible by designing more sophisticated hyper-features.

5. Modeling Pairwise Relationships Between Patches

In Sections 3 and 4, we described our method for scoring a model image patch F^L_j and its best match F^R_j by modeling the distribution of their difference 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^L_j 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 $F^L_{(k)}$ represent the kth patch picked for the cascade and let $F^L_{(1...n)}$ denote the first n of these patches. Assume we have already picked patches $F^L_{(1...n)}$ and we wish to choose the next one, $F^L_{(n+1)}$, from the remaining set of F^L_j'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,

$$I(D_{(n+1)}; C|D_{(1...n)}) \leq \min_{1 \leq i \leq n} I(D_{(n+1)}; C|D_{(i)}). \quad (11)$$

Thus, rather than maximizing the full information gain, Vidal-Naquet and Ullman [31] (see [13] for a comparison to other feature selection techniques) proposed the following heuristic that maximizes this upper bound on the amount of "additional" information:

$$\arg\max_j \min_i I(D_j; C|D_{(i)}), \quad (12)$$

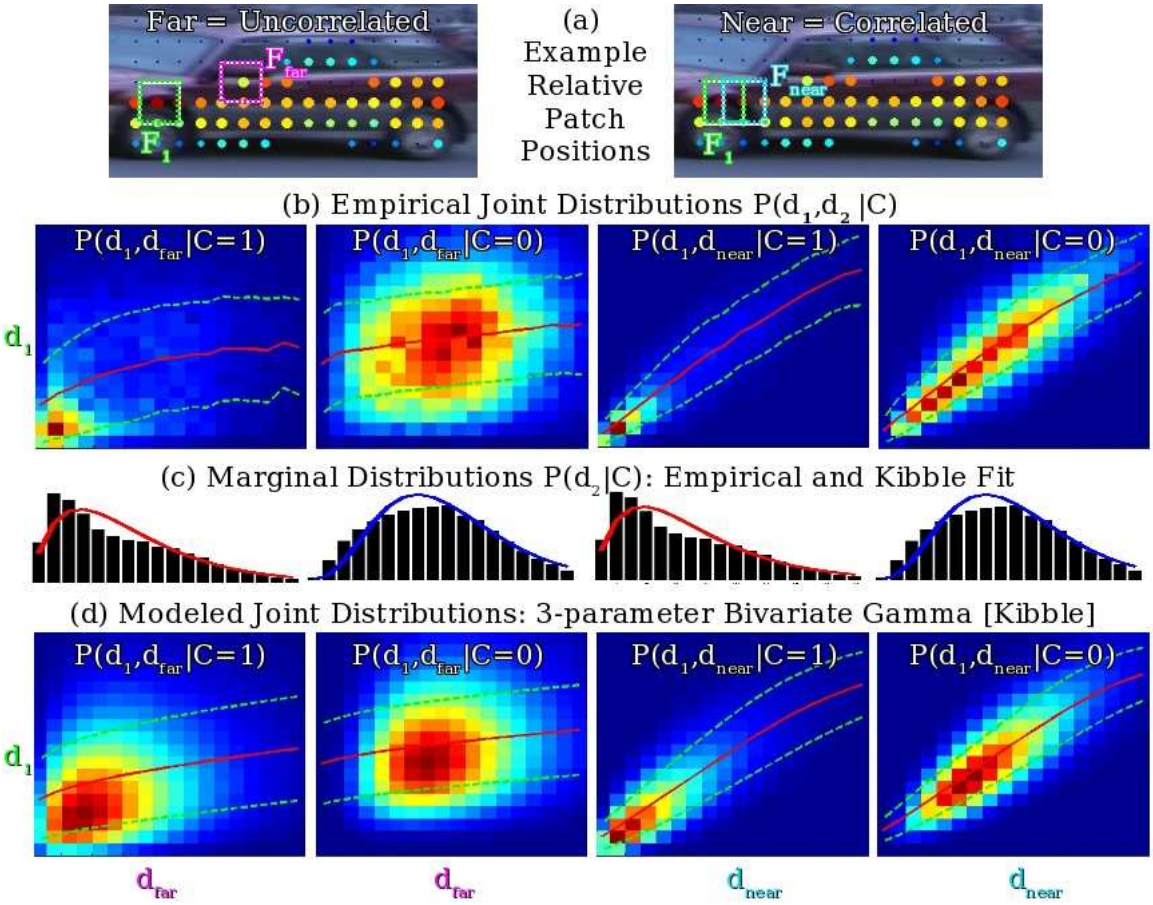

Figure 7: *Bivariate Gamma Distributions.* We demonstrate our technique by plotting the empirical and modeled joint densities of all patch pairs from the training set which are a fixed distance away from each other. On the **left** side, the two patches are far apart, thus they tend to be uncorrelated 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.

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 lead to choosing a suboptimal next patch $F_{(i)}$. In particular, 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:

$$\arg\max_j \min_i I(D_j; C|D_{(i)}) \times \frac{I(D_j; C)}{I(D_j^*; C|D_{(i)})}. \qquad (13)$$

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 (13), 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 also appearance-based hyper-features, we again use a generalized linear model to gain statistical leverage, this time to model the 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 [19], a generalization of the one-dimensional gamma distribution that is constrained to have gamma distributions as marginals. A Kibble distribution has four parameters: μ_1, μ_2, γ, and ρ, with $0 < \rho < 1$. μ_1 and μ_2 are mean parameters for the marginals. γ is a dispersion parameter for both marginals. ρ 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 γ 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 ρ.

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 ρ 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.

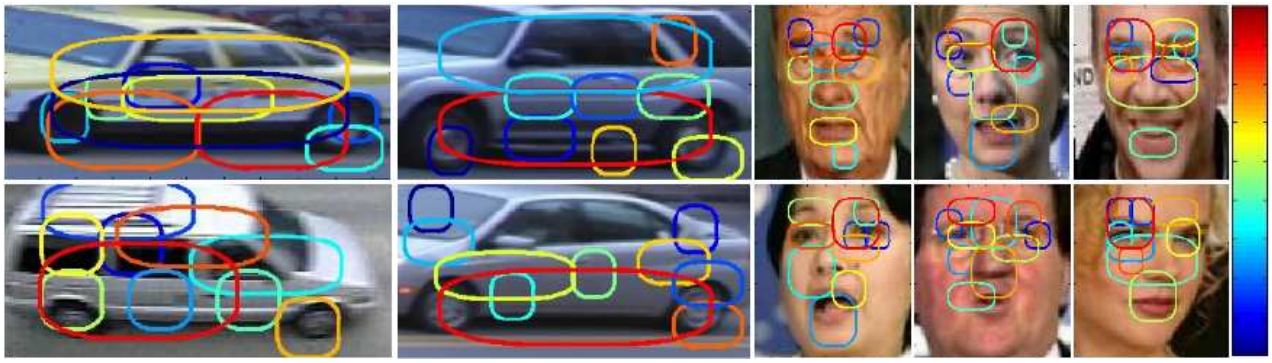

Figure 9: *The Ten Most Informative Patches.* The ten rectangles on each object show the top ten patches our identifier generator selected for the identification cascade *for that object*. The face model seems to prefer features around the eyes, while the two car models tend to both like the side and 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).

We model ρ, 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 ρ be a linear function of the *difference* of various hyper-features of the two patches, $F_{(i)}^L$ and F_j^L. 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

$$\rho = (1 - \exp(\beta \cdot \mathbf{Y}))^{-1}, \tag{14}$$

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

Given a data set of patch pairs $F_{(i)}^L$ and F_j^L 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 β. This is done by maximizing the likelihood of $K(\mu_j^{C=1}, \gamma_j^{C=1}, \rho)$ using data taken from image pairs that are known to be the "same"[6] and $K(\mu_j^{C=0}, \gamma_j^{C=0}, \rho)$ 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 [17] 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. 13. 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

$$\arg\max_j \min_i I(D_j; C) \times (1 - \rho_{j(i)}) \tag{15}$$

where $\rho_{j(i)}$ is computed by Eqn. 14 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_j^L (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 λ (see Section 3.2) that defines a global choice between selectivity and sensitivity. What remains is the definition of thresholds at each step, $\lambda_{(k)}^{C=1}$ and $\lambda_{(k)}^{C=0}$, which allow the system to accept (declare "same") if $R > \lambda_{(k)}^{C=1}$ or reject (declare "different") if $R \leq \lambda_{(k)}^{C=1}$. If neither of these conditions is met,

[6] $\mu_j^{C=1}$ and $\gamma_j^{C=1}$ are estimated from F_j^L by the method of Section 3.4 and are fixed for this optimization.

the system should continue by comparing the $k+1$th patches of each image.

To learn these thresholds, we generate identifiers on the left training images and run the resulting identifier comparing against 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 λ is defined. Our goal for the cascade is to make decisions as early as possible but to avoid increasing the error on the training set. 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 an identifier 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 as large as possible without misclassifying any additional "same" pairs over the base identifier which uses all patches.
3. Similarly define $I_{C=0}$, and set $\lambda_{(k)}^{C=1}$ using the $\min$.

7. Results

The goal of this work is 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, matching each one separately and combining the results,
2. differentiating patches by estimating a scoring and saliency function for each patch (based on its hyperfeatures),
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.

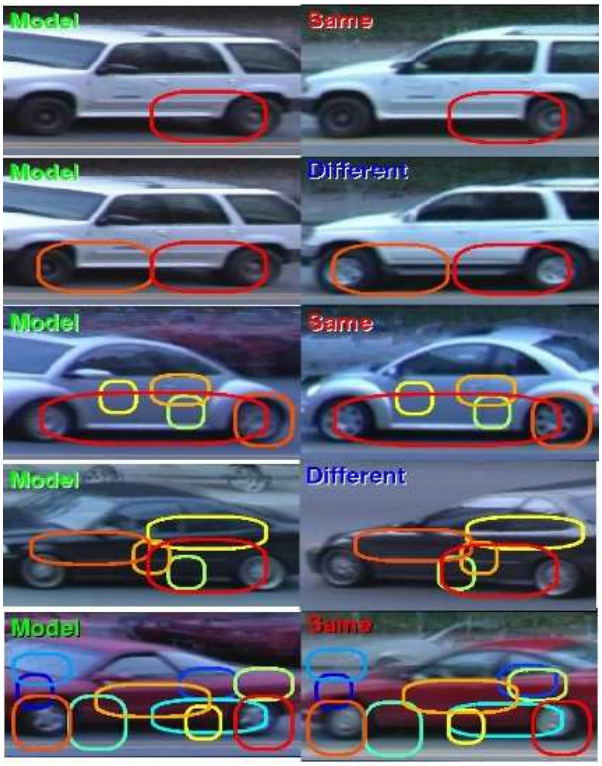

Figure 10: *Model-Test Car Image Pairs.* Each pair of images shows a model and a test image, which has been labeled as "same" or "different" (see upper left corner of test image) by our algorithm. The patches that were used in the cascade for that test image are indicated for each pair, where the order 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.

We tested our algorithm on three different data sets: (1) cars from two 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 system 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 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

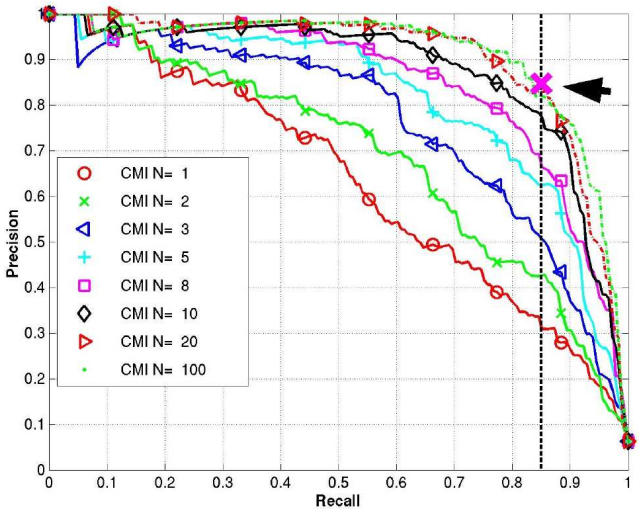

Figure 11: *Precision vs. Recall Using Different Numbers of Patches (Cars 1).* These are precision vs. recall curves for our full model. Each curve represents the performance tradeoff between precision and recall, when the system uses a fixed number of patches. The lowest curve uses only the single most 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.

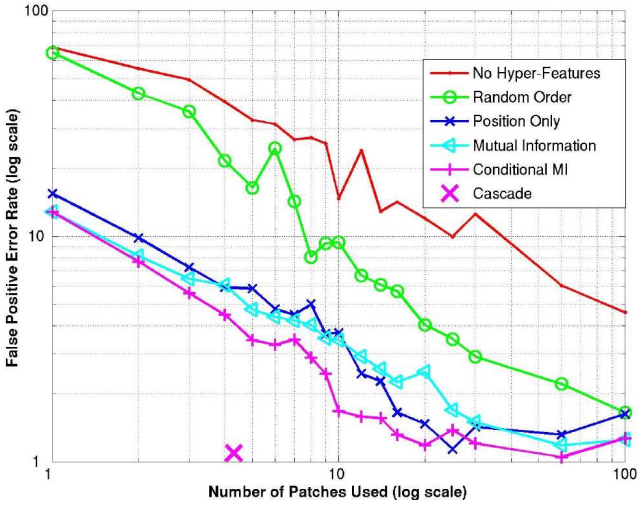

Figure 12: *Comparing Performance of Different Models (Cars 1).* The curves plot the performance of various models, as measured by the false-positive rate (fraction of different pairs labeled incorrectly as same), at a fixed recall rate of 85%. The y-axis shows the log error rate, while the x-axis plots 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 (including size: x, y, w, h) 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.

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.

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 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

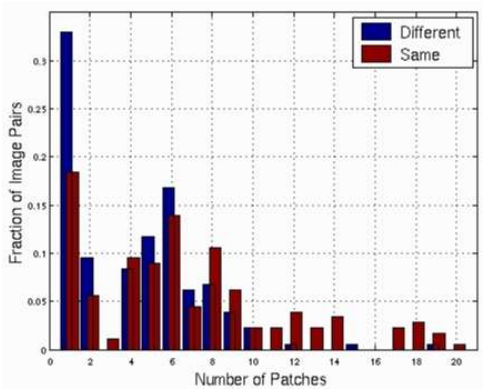

Figure 13: *How many patches does it take to make a decision?* This histogram shows the number of patches that were matched by the classification cascade before a decision could be made. On average, 4.2 patches were required to make a negative (declaring a difference) decision, and 6.7 patches to make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

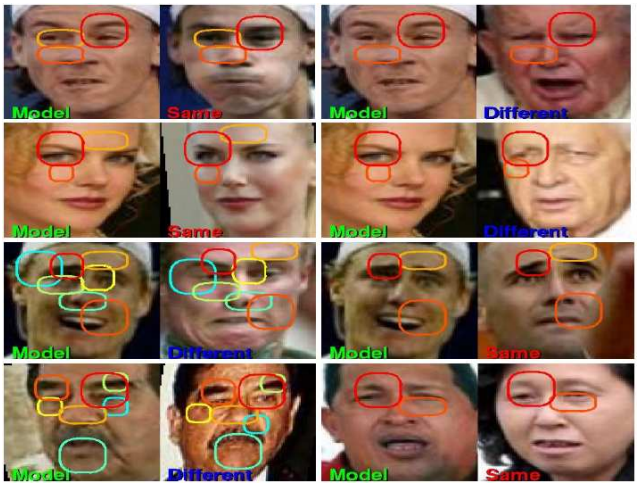

Figure 14: *Model-Test Face Image Pairs.* The first two rows of images show correct results, while the bottom two demonstrate errors. The large variations in pose, lighting, expression and image resolution make this data set very difficult. Our algorithm prefers eyes and seems to have learned that when the face is partially in profile, the eye that is more frontal is more informative (probably because it is more likely to be consistent).

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 are shown in Figure 10. This data set together with more example results are available from our web site.

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 precision-recall 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 those 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.

7.2. Faces

We used a subset of the "Faces in the News" data set described in [5], 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

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.

always using the best 50 patches (performance actually declines above 50 patches). Refer to Figure 14 for example classification results. As part of an extention of this current paper [16], we have also compared this algorithm to Bayesian face recognition [23], which won the 1996 FERET face identification competition, and found our algorithm to perform significantly better on this difficult data set. Our more recent work further improves on the results reported here by training a discriminative model on top of the hyper-features.

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 for each camera pair. In this fashion, 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 $\sim$140 pixels) and 5-7% for far lane ($\sim$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.

7.4. Algorithm Complexity

This algorithm was designed to be able to perform real-time object identification. The most computationally expensive part is the off-line training, as many patches must be sampled and matched using normalized correlation (3-800,000 in our experiments above). For the on-line step, our observation is that a model will be built only once for each category instance (step 2 of our algorithm), but that model will then be applied (matched) many times to incoming images (step 3). Thus we choose to pay the one-time cost of scanning over all candidate patches to build the most efficient classification cascade possible.

Evaluating the informativeness of all patches within an image is, however, not as computationally daunting as it sounds: computing all of our hyper-features for a patch can be performed using integral images, making their computation time independent of patch size. Given the vector of hyper-features for a patch, computing the "same" and "different" gamma distributions used by the information measure involves computing 2 dot products (one for each degree of freedom of the distribution). Finally, the mutual information measure is computed using a table-lookup based on the gamma parameters.

The most expensive on-line step (by far) is matching the patches of the cascade in step 3 by searching for the most similar patch in terms of normalized correlation. Therefore the on-line running time of our algorithm is directly a function of the average number of patches that must be matched before a decision can be made, and the size of the area that needs be searched for the best matching patch. Given the care with which we pick the patches of our cascade, the average number of patches is typically less then 5 (see above). The search area, on the other hand, depends not on our algorithm but the accuracy of object detection method used.

Our current implementation is in Matlab. We estimate that an optimized implementation of our algorithm would be able to perform the vehicle identification component of the system described above with up to five new vehicle reports per second, and 15 candidate ids per report, in real time on a single processor.

8 Conclusion

We have described a new object identification system that is general in that it can be applied to different categories of objects. We have shown that the key step is to teach the system to be able to pick informative features for new objects within the given category given an off-line labeled category training set. Our main contribution is a novel learning algorithm that

Figure 15: *Selecting Unusual Features.* For each image, the first (most informative) patch of the cascade is displayed.

actually models the space of possible features for the category, allowing it to select and score features that it has has never seen before, based on a single new example.

In the introduction, we have argued that our goal was to build an algorithm that has the potential to recognize the usefulness of a mole on a person's cheek for identification, even when it had never seen another person with such a mole. The hope is that the system would be able to generalize from having seen other similar facial blemishes in other locations (that is, in our case, image patches with mole-like appearance characteristics), and recognize that such patches, wherever they are located, make good features.

While we have no faces with obvious moles in our data set, Figure 15 shows two example results that make us hopeful that we are on the right track. In both, the algorithm has picked a very atypical patch as its top most informative feature, due to an unusual image characteristic. The left image contains the only person in the the our data set who is wearing sunglasses and it is the only image for which the algorithm has decided *not* to pick a patch near the eyes as its top feature. The right example shows a truck towing a flatbed trailer, where the unique connecting area is chosen as the best feature. We can only hypothesize how the algorithm made these seemingly correct yet unusual choices.

In both cases the appearance of the features dominated the decision: the homogeneous area of the sunglasses replaced the usually informative eye features, while the elongated lines of the trailer are reminiscent of the type of features that are found to be informative elsewhere. While accuracy of these unusual choices are quantitatively very difficult to measure, we believe that the overall performance of our algorithm is due to this ability to pick the right patches for most objects.

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 γ and a scale parameter β. In this work, we typically use the parameters γ and the mean μ, 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 μ_x, μ_y, γ, and a correlation term ρ. Such a distribution has gamma marginals, where μ_x and γ define the x marginal and μ_y and γ define the y marginal. The parameter ρ , which ranges $0 \leq \rho < 1$, is the correlation coefficient between the variables x and y: when ρ is small, x and y are close to independent; when ρ 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 ρ parameter, where values of ρ 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. Vidit Jain at UMass Amherst provided comparison to Bayesian face recognition. Author Learned-Miller was partially supported by NSF CAREER Award IIS-0546666.

References

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

[2] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using

class specific linear projections. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[3] S. Belongie, J. Malik, and J. Puzicha. Matching shapes. In *International Conference on Computer Vision*, pages 454–463, 2001.

[4] A. Berg, T. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondence. *CVPR*, pages 26–33, 2005.

[5] 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*, 2:848–854, 2004.

[6] E. J. Bernstein and Y. Amit. Part-based statistical models for object classification and detection. In *IEEE Computer Vision and Pattern Recognition*, pages 734–740, 2005.

[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*, pages 202–207, 2002.

[8] D. Bolme, R. Beveridge, M. Teixeira, and B. Draper. The CSU face identification evaluation system: Its purpose, features and structure. *ICVS*, pages 128–138, 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] G. Dork and C. Schmid. Object class recognition using discriminative local features. Technical Report RR-5497, INRIA Rhone-Alpes, 2005.

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

[12] 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*, volume 2, pages 1134–1141, 2003.

[13] F. Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research*, 5:1531–1555, 2004.

[14] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *13th International Conference on Machine Learning*, pages 148–156, 1996.

[15] B. Heisele, T. Poggio, and M. Pontil. Face detection in still gray images. *Massachusetts Institute of Technology Artificial Intelligence Lab*, A.I. Memo No. 521, May 2000.

[16] V. Jain, A. Ferencz, and E. Learned-Miller. Discriminative training of hyper-feature models for object identification. In *British Machine Vision Conference*, volume 1, pages 357–366, 2006.

[17] 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.

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

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

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

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

[22] 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 464–471, 2000.

[23] B. Moghaddam, T. Jebara, and A. Pentland. Bayesian face recognition. *Pattern Recognition*, 33:1771–1782, November 2000.

[24] G. Mori, S. Belongie, and J. Malik. Shape contexts enable efficient retrieval of similar shapes. *CVPR*, pages 723–730, 2001.

[25] H. Schneiderman and T. Kanade. A statistical approach to 3d object detection applied to faces and cars. *CVPR*, pages 1746–1759, 2000.

[26] N. Shental, A. Bar-Hillel, T. Hertz, and D. Weinshall. Computing Gaussian mixture models with EM using equivalence constraints. *NIPS*, 2003.

[27] N. Shental, T. Hertz, D. Weinshall, and M. Pavel. Adjustment learning and relevant component analysis. *ECCV*, 2002.

[28] 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.

[29] S. Thrun. *Explanation-based neural network learning: A lifelong learning approach*. Kluwer, 1996.

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

[31] M. Vidal-Naquet and S. Ullman. Object recognition with informative features and linear classification. In *International Conference on Computer Vision*, pages 281–288, 2003.

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

[33] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. *ECCV*, 1:18–32, 2000.

[34] 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*, 19(7):775–779, 1997.

[35] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. *Advances in Neural Information Processing Systems*, 2002.

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