Dictionary-Free Categorization of Very Similar Objects via Stacked Evidence Trees

Gonzalo Martínez-Muñoz[1] Wei Zhang[1] Nadia Payet[1] Sinisa Todorovic[1]
Natalia Larios[2] Asako Yamamuro[1] David Lytle[1] Andrew Moldenke[1]
Eric Mortensen[3] Robert Paasch[1] Linda Shapiro[2] Thomas G. Dietterich[1]

[1]Oregon State University
Corvallis, OR 97331 USA

[2]University of Washington
Seattle, WA 98195 USA

[3]Lucidyne Technologies, Inc.
Corvallis, OR 97333 USA

Project Home Page: `http://web.engr.oregonstate.edu/~tgd/bugid/`

Abstract

Current work in object categorization discriminates among objects that typically possess gross differences which are readily apparent. However, many applications require making much finer distinctions. We address an insect categorization problem that is so challenging that even trained human experts cannot readily categorize images of insects considered in this paper. The state of the art that uses visual dictionaries, when applied to this problem, yields mediocre results (16.1% error). Three possible explanations for this are (a) the dictionaries are unsupervised, (b) the dictionaries lose the detailed information contained in each keypoint, and (c) these methods rely on hand-engineered decisions about dictionary size. This paper presents a novel, dictionary-free methodology. A random forest of trees is first trained to predict the class of an image based on individual keypoint descriptors. A unique aspect of these trees is that they do not make decisions *but instead merely record* evidence*—i.e., the number of descriptors from training examples of each category that reached each leaf of the tree. We provide a mathematical model showing that voting evidence is better than voting decisions. To categorize a new image, descriptors for all detected keypoints are "dropped" through the trees, and the evidence at each leaf is summed to obtain an overall evidence vector. This is then sent to a second-level classifier to make the categorization decision. We achieve excellent performance (6.4% error) on the 9-class STONEFLY9 data set. Also, our method achieves an average AUC of 0.921 on the PASCAL06 VOC, which places it fifth out of 21 methods reported in the literature and demonstrates that the method also works well for generic object categorization.*

1. Introduction

A central concern of recent research in computer vision has been generic object recognition, as exemplified by benchmark datasets, such as PASCAL [7]. Such research seeks to understand the universal ability of humans to recognize that previously-unseen objects belong to generic object classes. The object classes tend to be very distinct, with gross differences that are easy for humans to distinguish.

While this addresses important fundamental questions in computer vision, it does not solve any pressing application problem. In contrast, our research is focused on the problem of automating biodiversity studies to support biomonitoring (i.e., monitoring the health of ecosystems by monitoring the abundance and diversity of the species in the system) and fundamental ecological research (e.g., understanding population dynamics, mapping species distribution and habitat, measuring species' response to climate change, assessing the effectiveness of restoration and remediation efforts). Specifically, our goal is to develop general-purpose methods for categorizing images of arthropods (insects, spiders, etc.) to the level of family, genus, or species. This is an under-explored domain of categorization – that of classifying extremely challenging images that show objects exhibiting large intra-category variations and small inter-category differences, so that even trained human experts cannot easily categorize them, as illustrated in Fig. 1.

For the past several years, we have been collecting, photographing, and manually-classifying stonefly larvae to create a labeled database of images [10]. Stoneflies inhabit freshwater streams, and they are known to be a sensitive and robust indicator of stream health and water quality. While it is easy to collect specimens, a high degree of expertise is required to classify specimens according to species. This has severely limited their use in biomonitoring. An auto-

978-1-4244-3991-1/09/$25.00 ©2009 IEEE

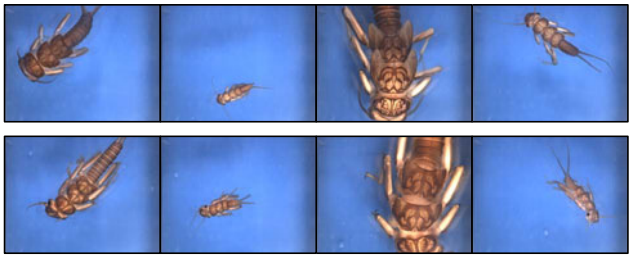

Figure 1. Sample images from our STONEFLY9 dataset. The top and bottom rows show two distinct stonefly species, difficult to classify even for trained human experts. In addition to small inter-class differences, STONEFLY9 presents additional challenges to object categorization: (1) The images show imperfect dorsal views of insects which may be only partially visible; (2) The geometric and photometric properties (e.g., size, color, and texture) change significantly with the insect's age; blob-like parts of the insects may contain specularities; multiple legs and antennae allow these highly articulated insects to appear in a wide range of poses.

mated system for photographing, classifying, and sorting specimens could greatly advance biomonitoring for water quality and basic ecological research.

In previous work [10], we applied the standard bag-of-keypoints dictionary approach [4, 16, 1, 8] to this problem. Our first dataset, STONEFLY2, consisted of images of two categories of stoneflies (known as Cal and Dor). We applied three different interest operators to identify keypoints and extracted SIFT descriptors from each keypoint region. These were then clustered (by fitting a Gaussian mixture model via EM) to create a separate visual dictionary for each combination of class and detector. Each image was then re-represented as a histogram of the number of occurrences of each dictionary entry (for each dictionary), and these histograms were concatenated to create a feature vector which was then fed to a classifier to perform the categorization. We applied various classifiers including boosted decision trees and boosted logistic model trees. On STONEFLY2, this bag-of-keypoints approach achieves an error rate of 20.3%. To assess this level of performance, we measured the ability of people (26 students and faculty from the zoology department) to perform the same task with the same images and discovered that they exhibited an error rate of 21.4%, which was statistically indistinguishable from our system. Subsequently, we have enlarged our database to include images of 9 stonefly categories to create the STONEFLY9 database. On this database, the bag-of-keypoints approach achieves an error rate of 16.1%, which is considered by our entomology collaborators to be mediocre.

Based on this work, we conclude that state-of-the-art generic object categorization methods are not sufficient to achieve high performance on difficult categorization problems such as STONEFLY9. We hypothesize that there are three reasons for the failure of these techniques. First, the dictionaries are constructed using purely unsupervised methods (although they are class-specific). As many researchers have pointed out, the regions of descriptor space that have high probability density—that is, the ones found by K-means or Gaussian mixture model clustering—are not necessarily the most useful for discriminating among categories. Second, when detected regions are mapped to dictionary entries, much information is lost. A 128-dimensional SIFT descriptor captures detailed information about the intensity texture of the detected region. Mapping this to one entry in a 2700-word dictionary (as we did for STONEFLY9) retains at most 12 bits of information (vs. 8×128=1024 bits for SIFT). Third, the dictionary approach requires manual tuning to select the number of clusters and the method for mapping descriptors to clusters (Euclidean distance, Mahalanobis distance, keyword counts, keyword probabilities, etc.).

Several researchers have developed quasi-supervised methods for creating visual dictionaries [14, 16, 17, 19]. Winn, et al. [16] first learn a very large number of clusters via K-means clustering and then merge clusters to maximize their discriminative power. Moosmann, et al. [14] build a forest of extremely randomized decision trees that attempt to classify the image directly from individual descriptor vectors. They then define one dictionary entry for each leaf in these trees. This method is discriminative, but it creates a very large dictionary which can be unweildy and lead to overfitting. Yang, et al. [17] construct a sequence of "visual bits" through a boosting algorithm. Each visual bit provides a boolean feature that can be input to a classifier. Zhang and Dietterich [19] post-process an unsupervised dictionary with relevant component analysis to improve the discriminative power of each cluster. While all of these methods improve on the dictionary construction process, they do not address the other two sources of the failure: the loss of information when mapping to the dictionary and the problem of manual engineering of the system.

In this paper, we pursue a simple and elegant approach that is able to address all three shortcomings of the bag-of-keypoints approach. Like Moosmann [16], the method constructs a randomized forest of trees that directly analyze the descriptor vectors. However, instead of using the trees to define a dictionary, we view the trees as a way of discriminatively structuring the evidence in the training set. Each leaf of each tree stores a histogram of the number of training examples from each category that reached that leaf. To classify a new image, the detected keypoint descriptors are dropped through all of these *evidence trees*. Each time a keypoint reaches a leaf, the evidence stored in that leaf is accumulated into an overall histogram. After normalization, this histogram is then passed to a second level ("stacked") classifier to make the final category prediction. In effect, each keypoint votes for the category of the object, but the

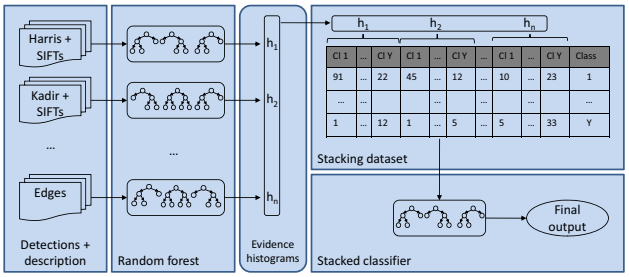

Figure 2. Block diagram of our approach: Descriptors of interest points and edges are directly input to a random forests without computing a visual dictionary. Each leaf node in the forest stores a histogram of the class labels of the training descriptors that reached this leaf. Descriptors for a new image are "dropped" through the forest until they reach a leaf node. The class histograms at all such leaves are summed to produce a single class histogram which is then provided as a feature vector to a boosted ensemble of decision trees that makes the final prediction.

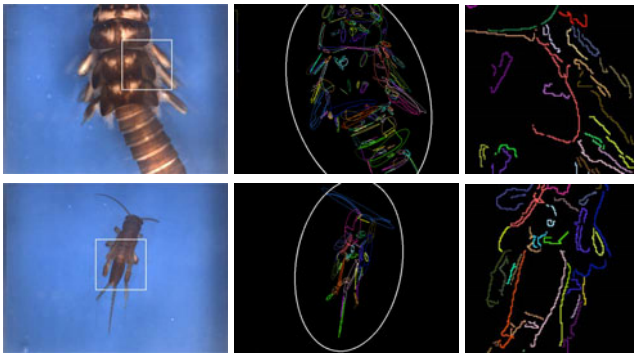

Figure 3. The top and bottom rows show different species from STONEFLY9, which are also different from those shown in Fig. 1: (left) A sample original image; (middle) Detected edges and ellipses fitted to pixels of each edge; and (right) Zoomed-in detail marked by a white box in (left). The large ellipse localizes the insect and provides information about the insect's main orientation.

vote is based on the corresponding evidence from the training set. We call this method "Voting Evidence".

We also studied a strategy whereby when a keypoint reaches a leaf of a tree, it places a single vote for the most frequent class in that leaf (according to the training set), which is the standard way that decision trees work. These votes are accumulated and passed to the stacked classifier. We call this method "Voting Decisions". We provide experimental evidence and a mathematical model showing that voting evidence is better than voting decisions.

An added benefit of the stacked classifier is that we can easily fuse different kinds of descriptors as well as arbitrary image-level information. In this paper, in addition to keypoints, we also extract contour features and build evidence trees for them. The evidence accumulated from these trees is provided to the stacked classifier as additional features.

Our voting evidence method achieves an error rate of 6.4% on STONEFLY9 using only keypoint descriptors. With the additional contour-based descriptors, the error rate improves to 5.6%. Our entomologists view this as extremely good accuracy—more than sufficient to provide a basis for a practical biomonitoring system.

This paper is organized as follows. Sec. 2 describes features and their associated descriptors. Sec. 3 explains details of the "Voting Evidence" and "Voting Decisions" methods and the stacked classifier. Sec. 4 presents experimental results on STONEFLY9 and PASCAL06; Sec. 5 presents a theoretical analysis of voting evidence and shows that under reasonable assumptions it gives lower error than voting decisions. Section 6 presents our concluding remarks.

2. Feature Extraction

This section explains the image features and descriptors we use.

Keypoints: On each image, we apply the following detectors to extract interest points: the Harris and Hessian detectors [13], the PCBR detector [5], and the Kadir-Brady salient region detector [9]. The PCBR detector, introduced in [5], detects stable watershed regions surrounded by curvilinear structures. These are particularly well-suited to the shapes found in our biological images. Our studies on STONEFLY4 showed that combining multiple detectors gave better results than any single detector [10]. Each detected keypoint is represented by the SIFT descriptor [12] for subsequent processing.

Edges are extracted by the Canny edge detector. Let E denote the set of detected edges, $E = \{e_1, \ldots, e_n, \ldots, e_N\}$. Intrinsic and spatial-layout properties of each edge, e_n, are then used to define its descriptor vector $\boldsymbol{x}_n$. Many of these properties are defined relative to the entire insect's orientation in the image. To identify the spatial extent and orientation of the insect, we make use of the fact that each image contains only one insect and that the background is nearly uniform. This means that most edges detected in the image belong to the insect. In particular, we expect that an ellipse, Λ_E, fitted to all pixels that belong to E, will localize the insect and provide useful information about the insect's head, tail, and main orientation. After estimating Λ_E, we compute the following: angle, ϕ_E, that the major principal axis of Λ_E subtends with the x axis of the image; length M_E of the intercept of the major principal axis with Λ_E; and length m_E of the intercept of the minor principal axis with Λ_E. The parameters (ϕ_E, M_E, m_E) of Λ_E are taken to represent the orientation, length, and width of the insect, as illustrated in Fig. 3. To specify intrinsic and spatial-layout properties of each edge, $e_n \in E$, we fit an ellipse, Λ_n to all pixels belonging to e_n and estimate the parameters (ϕ_n, M_n, m_n) in the same way as for E. For estimating the spatial layout properties, we define a neighborhood system among the de-

tected edges based on their Delaunay triangulation. To this end, each edge e_n is represented as point corresponding to the center (x_n, y_n) of the ellipse Λ_n. Neighbors of e_n are all those edges to which e_n is connected in the Delaunay triangulation.

For each edge e_n, we compute the following intrinsic and spatial layout properties: (1) Area coverage $\alpha_n = \frac{\text{area of } \Lambda_n}{\text{area of } \Lambda_E}$; (2) Orientation ϕ_n measured relative to ϕ_E, to achieve rotation-in-plane invariance; (3) Contrast γ_n estimated as the average gradient magnitude along e_n; (4) Signed curvature κ_n estimated on a piecewise linear estimate of e_n; (5) Displacement vector $\boldsymbol{d}_n$ between the center (x_E, y_E) of ellipse Λ_E and the center (x_n, y_n) of ellipse Λ_n; (6) Number of neighbors ν_n estimated from the Delaunay triangulation of all edges; (7) Mean μ_n and variance σ_n of strengths of interaction between e_n and its neighbors, where each interaction strength is inversely proportional to the length of the Delaunay arc connecting the two corresponding neighbors; (8) Context vector $\boldsymbol{\chi}_n$ computed as the standard 16×16 log-polar descriptor that counts the number of edge pixels in the neighborhood of e_n, where the descriptor is centered and scaled with respect to ellipse Λ_n.

In summary, the descriptor associated with each edge e_n contains the following properties $\boldsymbol{x}_n = \{\alpha_n, \phi_n, \gamma_n, \kappa_n, \boldsymbol{d}_n, \nu_n, \mu_n, \sigma_n, \boldsymbol{\chi}_n\}$.

3. Overall Classification Architecture

Fig. 2 shows the overall architecture of the system. We first describe the classification process and then the training process. The architecture is parameterized by a set of Q of (detector, descriptor) pairs $c = 1, \ldots, C$, where the detectors may detect keypoints, edges, etc. and the descriptors can be SIFT, filter banks, edge descriptor defined above, etc. To classify a new image I, we iterate over each (detector, descriptor) pair c in Q, apply the detector to I, and then represent each detection using the descriptor. This produces a bag of descriptor vectors $B_I^c = \{x_{I,1}^c, \ldots, x_{I,N_I}^c\}$.

For each combination c, let RF^c denote a previous-learned random forest of evidence trees. Each descriptor vector $x_{I,j}^c$ is "dropped through" each tree in RF^c until it reaches a leaf ℓ. That leaf ℓ stores a histogram h_ℓ^c such that $h_\ell^c[k]$ is the number of training examples from class k that reached the leaf during training. These histograms are summed over all trees to obtain h_j^c and over all j to obtain h_c. Each h_c is normalized (to sum to 1), and then the C histograms are concatenated to form the second level feature vector. This is then processed by the stacked classifier to assign a category to the image.

The learning process involves three steps: (a) learning the random forests, (b) constructing the second-level training set, and (c) learning the stacked classifier.

Learning a random forest. A random forest [2] is a set of decision trees all created from a single training set S. Each tree is constructed in the usual top-down way as in C4.5 [15], but with two modifications. First, the training data for each tree is obtained by taking a bootstrap sample [6] of S. (A bootstrap sample of size N is created by drawing N points uniformly at random with replacement from S.) Second, recall that each node η of a decision tree compares the value of a chosen attribute a (e.g., $x_{Ij}^c[a]$) against a chosen threshold θ_η and branches left if $x_{I,j}^c[a] \leq \theta_\eta$ and right otherwise. When growing a standard decision tree, all possible attributes a and all reasonable thresholds θ are considered and the (a, θ) combination with the highest discriminative power is chosen. In a random forest, at *each node* a random subset of size $1 + \lfloor \log A \rfloor$ is chosen (where A is the number of attributes; 128 for SIFT). Then the most discriminative combination (a, θ) is chosen by evaluating only this random subset. These two forms of randomization, bootstrap subsampling of the training data and random subsetting of the attributes at each node, act together to create a diverse ensemble of decision trees. In a standard random forest, a new instance is classified by a vote of the decisions of the individual trees.

We modify this random forest learning algorithm in several ways. First, to construct the training set S^c for RF^c, we start with each bag of descriptors B_I^c for each image I (with category label y_I) and create one training example for each descriptor vector: $(x_{I,j}^c, y_I)$. Then, we generate the bootstrap samples by drawing *images* with replacement from the set of training images. Second, we constrain the growth of the tree so that every leaf node contains at least 20 training examples. Finally, in each leaf ℓ, we store the histogram h_ℓ of the number of training examples belonging to each class.

For each detector/descriptor combination c, we train a random forest RF^c containing 100 evidence trees.

Creating the stacked training set. Once the initial random forests are built based on individual descriptor vectors, we then construct a second-level ("stacking") training set that contains one example per image. The stacking example for an image I is computed as follows. Because each tree is grown on a bootstrap sample, there is a set of so-called "out of bag" images that were *not* used to grow that tree because they were not members of the bootstrap sample. First, we initialize a histogram h_c for each detector/descriptor combination c. Then, for each image I, we take its descriptors and "drop" them through each tree for which I was "out of bag". Each time a descriptor reaches a leaf ℓ of random forest RF^c, we take the histogram h_ℓ and add it into a histogram h_c. Finally, for each c, we divide h_c by the sum of its components to normalize. This gives us one histogram for each detector/descriptor combination. These histograms are concatenated to form the feature vector for the stacking example. The class label of image I is then assigned to be the class label of this stacking example. Note that when we

concatenate the various normalized histograms h_c, we are fusing the information from both keypoint and edge features. Any other source of image-level information could be included in the stacking examples at this point.

Training the stacked classifier. We apply the Adaboost algorithm to train an ensemble of 200 decision trees.

This architecture addresses the three problems that plague the dictionary approach. First, because the evidence trees examine the individual attributes of the descriptor vectors, no information is lost in quantization. Second, the evidence trees are grown discriminatively, so there are no unsupervised steps. Third, the only parameters to be determined in the architecture are (a) the minimum number of training examples in each leaf node, (b) the number of trees in each random forest, and (c) the number of boosting iterations for the stacked classifier. Our experiments have shown that the results are insensitive to all of these parameters.

4. Experimental Results

We evaluated our categorization approach on the STONEFLY9 [10] and PASCAL06 datasets [7]. STONEFLY9 consists of 3826 images obtained by imaging 773 stonefly larvae specimens, examples of which are shown in Figs. 1 and 3. The dataset contains nine classes referred to as Cal, Dor, Hes, Iso, Mos, Pte, Swe, Yor and Zap. The experimental setup for STONEFLY9 consists of a stratified 3-fold cross validation using two folds for training and one for testing. Images from the same specimen are constrained to belong to the same fold. PASCAL06 consists of 5304 images of natural scenes that contain 10 object classes of interest. The object classes in PASCAL06, as in other popular benchmarks, differ from one another significantly.

For STONEFLY9, we apply three keypoint detectors (Hessian, Kadir-Brady, PCBR interest points) and the Canny edge detector. Keypoints are represented using SIFT descriptors, and edges are represented using the edge descriptor presented in Sec. 2. Four random forests are trained. For PASCAL06, we apply four detectors (Harris, Hessian, PCBR and regularly sampled image patches of sizes 24, 32, and 64 pixels) and represent each resulting keypoint using 3 detectors: SIFT, Color SIFT, and the filter bank descriptor employed by Winn et al. [16]. Twelve random forests are trained (one for each combination of detector and descriptor). When growing the random forests for the Hessian and Harris detections in PASCAL06, there are so many detections (over $1.6 \cdot 10^6$) that we cannot use all of them when growing any single tree. Hence, for each tree, we draw a random subsample of 40% (for Hessian) and 35% (for Harris) of the detections prior to growing each tree.

Each PASCAL06 image can contain several objects from the same category or from multiple categories. We treat each object in each image as a separate instance, and, when training the classifiers, we consider only those detections that fall exclusively within the bounding box for that object. Detections from regions where the bounding boxes overlap are discarded. In testing, we must of course use all detections in the image. To reduce the effect of background noise the stacked dataset is created by adding up for each class k only the N detections with the highest $h_j^c[k]$ values. The value of N was estimated for each class from the training set from: 25, 50, 100, 200, 400, 600 and all detections. For most classes, all the detections on the images were used. Note that unlike most work with PASCAL06, we train a single multi-class system rather than training a separate classifier for each category.

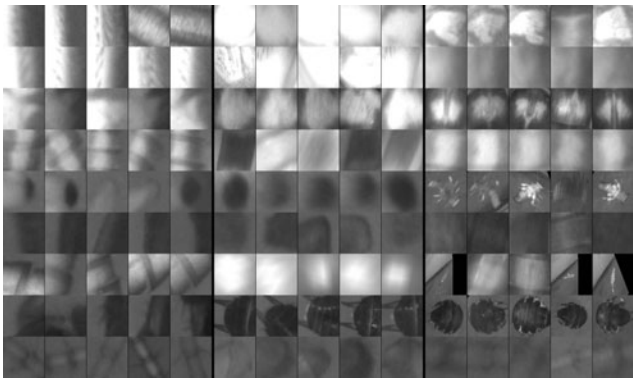

Figure 4. Examples of most discriminative patches for STONEFLY9. The rows correspond to the 9 classes in alphabetical order from the top. The first five columns are the most informative patches for Kadir; the next five for Harris; the last five for PCBR

4.1. Qualitative Results

Fig. 4 shows the most discriminative patches of the test STONEFLY9 images as selected by our classifier. The patches shown are those with the highest $h_j^c[k]$ values. There are patches of many different scales. The PCBR detector has selected whole-bug images for Mos and Yor. Our experts confirm that these two classes are best recognized by their overall shape. For Hes, PCBR has picked out an interesting 2-lobed area in the center of the back. Although not previously noticed by entomologists, subsequent examination of specimens reveals that this area is sufficient to discriminate among Cal, Hes, and Dor. Other informative patches found in this figure correspond to "hairs" (actually, gills) emerging from body joints, regions of the head, and various textures and spots.

4.2. Quantitative Results

Table 1 shows the results of applying both our stacked random forest classifier and a visual dictionary classifier to STONEFLY9. The dictionary approach employed K-means clustering to define a 100-element dictionary separately for each combination of detector/descriptor and class. Hence,

Table 1. Classification Error on STONEFLY9. Because keypoint dictionaries cannot incorporate edge features, no results are reported for configurations involving them.

Detector	Stacked R. Forest	Visual Dict.
Hessian	15.5±2.8	34.0±3.1
Kadir	11.1±1.1	23.9±1.3
PCBR	11.2±1.8	28.1±1.0
Edges	36.3±1.1	-
Hessian + Edges	11.4±1.9	-
Kadir + Edges	9.8±0.4	-
PCBR + Edges	9.2±1.8	-
Hessian + Kadir	8.1±1.5	19.0±2.5
Kadir + PCBR	7.8±1.6	19.2±1.7
PCBR + Hesaff	7.8±2.3	20.2±2.6
All keypoints	6.4±1.8	16.1±1.8
Edges + all keypoints	5.6±2.1	-

there were $3 \times 9 \times 100 = 2700$ words in the combined dictionary, and hence 2700 attributes in the feature vectors. SIFT descriptors were separately mapped to the nearest cluster center in each dictionary and accumulated into a histogram for classification by a boosted decision-tree classifier containing 200 trees—the same size as the second-level stacked classifier in our architecture.

On STONEFLY9, the error rate for stacked random forests is substantially below the error rate of the visual dictionaries. When only keypoints are used, our classification error is 6.4%, versus 16.1% for the visual dictionaries. Indeed, the performance of stacked random forests using only a single detector beats the performance of visual dictionaries when trained on keypoints from all three detectors.

The ability to fuse edges improves the stacked random forests. The effect is largest when edge information is fused with SIFTs from a single detector (e.g., Hessian alone has 15.5% error, whereas Hessian + edges has 9.8%). There is still a gain when edges are fused with SIFTs from all keypoints—the error rate of the stacked random forests incorporating all information is 5.6%.

The time required to train our system is much less than for the standard visual dictionary approach. The random forest for the PCBR detector ($\approx$ 400,000 detections) is trained in 127 min. (amd64 with 2.8 MHz CPU and 4GB memory). It takes only a few minutes to train the stacked classifier. Note that the number of attributes that the stacked classifier receives is small (only 36: 9 classes $\times$ 4 descriptors). In contrast, the time to build the final classifier for visual dictionaries on the three keypoint detectors is over 200 min. because of the large number of attributes (2700). Note that this time does not include the construction of the visual dictionary, which requires several hours.

Figure 5 shows the results for PASCAL06 [7, 18, 11]. Each line corresponds to one published method, and each column shows the (rescaled) AUC. Our method ranks 5th out of 21 (including ours). The best performing methods are either exploring the spatial distribution of the keypoints

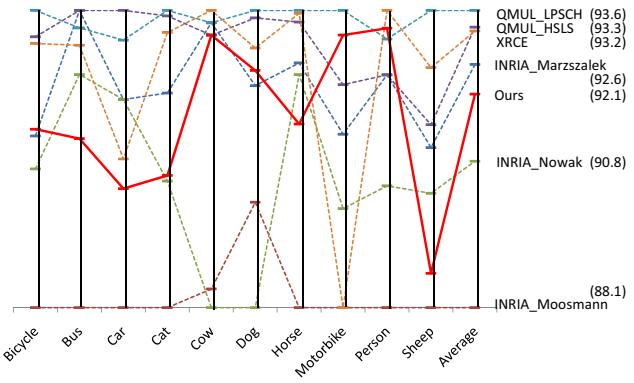

Figure 5. AUC on the PASCAL06 test set for the top 6 published method plus our method. For clarity, max/min AUC values have been rescaled separately for each task. Average AUC is shown between parentheses.

using spatial pyramid [11] or learning more complex image classifiers [18]. All of these methods learn a separate classifier for each category, which probably gives them an advantage over our results, which are generated from a single, multi-class classifier.

5. Mathematical Model

In standard random forests, a new instance is classified by voting the decisions of each tree in the forest. However, we found that voting evidence was more accurate than voting decisions. For example, the error rate on STONEFLY9 with Hessian, Kadir and PCBR is 16.4%, 12.0% and 12.3% respectively for voting decisions as compared with 15.5%, 11.1% and 11.2% for voting evidence. This was surprising, because the conventional wisdom has been that voting decisions works best for ensemble learning algorithms (i.e. bagging, boosting, etc). To understand why voting evidence leads to improved classification accuracy in object recognition, we developed the following mathematical model.

Let us consider the simple case where there are two categories of objects and each image contains exactly one instance of an object. In each image, suppose there are d detections, and hence, d SIFT descriptor vectors. We will assume that a fraction π of these are "informative" SIFT vectors in the sense that they are much more likely to have come from only one of the two classes. The remaining $(1 - \pi)d$ SIFT vectors are assumed to be uninformative, because they capture irrelevant regions in the background or foreground regions whose appearance is constant across the classes.

We will first analyze the case of a single decision tree. Suppose we have constructed a tree where every leaf has a true probabilistic margin of γ, by which we mean that if we consider only the informative SIFTs that reach a leaf, then fraction $\frac{1}{2} + \gamma$ of them belong to one class and $\frac{1}{2} - \gamma$

of them belong to the other class. In our training set, suppose C SIFT vectors reach each leaf. By assumption, πC of them are informative. If we consider only these informative SIFTs, what is the probability ϵ that the majority class of the training SIFTs does not match the true class of the leaf? The Chernoff bound [3] provides an approximate answer:

$$\epsilon \leq \exp[-2\pi C\gamma^2] \; . \tag{1}$$

Unfortunately, our training data also includes the $(1-\pi)C$ uninformative SIFTs. These have the effect of reducing the margin from γ to $\pi\gamma$.

Lemma 1. If the fraction of informative SIFTs is π, the fraction of uninformative SIFTs is $1-\pi$, and the margin of the informative SIFTs at a leaf is γ, then the effective margin of all of the SIFTS is $\pi\gamma$

Proof: Suppose the true class of a leaf is class 1. Let us compute the probability that a SIFT reaching that leaf is labeled with class 1. With probability π, the SIFT is informative, and with probability $\frac{1}{2}+\gamma$ it belongs to class 1. With probability $1-\pi$, a SIFT is uninformative, and we assume uninformative SIFTS have probability $1/2$ of being labeled as class 1. Consequently, the probability that a SIFT is labeled as class 1 is

$$\pi\left(\frac{1}{2}+\gamma\right)+(1-\pi)\,\frac{1}{2} \quad = \quad \frac{1}{2}+\pi\gamma \; .$$

Hence, the effective margin is $\pi\gamma$. □

Now let us consider classifying a new image using this decision tree τ by classifying each of the d SIFT vectors in the new image and taking the majority vote of these individual classification decisions.

Proposition 1. The error rate ϵ_{vd} (for "voted decisions") of classifying each SIFT vector separately and then taking the majority vote is bounded by

$$\epsilon_{vd} \leq \exp[-2d(\pi\gamma(1-2\epsilon))^2]. \tag{2}$$

Proof: Let $\epsilon = \exp[-2C(\gamma\pi)^2]$ be the probability that each leaf in the decision tree was incorrectly labeled during the training process. For concreteness, suppose the object in the image belongs to class 1 and let x be a SIFT vector selected uniformly at random from the d detections in the image. We will compute the probability that x will be predicted by the decision tree to belong to class 1.

With probability π, x is informative. With probability $\frac{1}{2}+\gamma$, x will reach a leaf node whose true majority class is class 1. With probability $1-\epsilon$, x will be correctly labeled as class 1. With probability $\frac{1}{2}-\gamma$, x will reach a leaf whose true majority class is class 2. With probability ϵ, that leaf will be incorrectly labeled, so x will be correctly predicted to belong to class 1. Finally, with probability $1-\pi$, x is uninformative, so it will be sent to a randomly-selected leaf and have probability $1/2$ of being labeled as class 1. Collecting these terms and simplifying, the probability that x is predicted to belong to class 1 is

$$\pi\left(\frac{1}{2}+\gamma\right)(1-\epsilon)+\pi\left(\frac{1}{2}-\gamma\right)\epsilon+(1-\pi)\frac{1}{2} = \frac{1}{2}+\pi\gamma(1-2\epsilon)$$

Hence, the effective margin of τ is $\pi\gamma(1-2\epsilon)$. We can treat each detection as an independent draw of a binomial random variable with this margin and apply the Chernoff bound to complete the proof. □

Suppose instead that we make our classification decision by summing the counts (the evidence) computed when the tree was constructed:

Proposition 2. The error rate ϵ_{ve} (for "voted evidence") of accumulating the leaf histograms for each SIFT vector and then taking the class with the highest count is bounded by

$$\epsilon_{ve} \leq \exp[-8dC(\gamma\pi)^4]. \tag{3}$$

Proof: Once again, we will start by computing the effective margin and sample size for a Bernoulli random variable, which we will call v. In this case, the value of v is generated by choosing a random SIFT vector x from the image, dropping it through the tree τ to find a leaf, choosing one of the training SIFTs stored at that leaf, and taking the class label of that training SIFT. Suppose the image belongs to class 1, what is the probability that v will be labeled as belonging to class 1?

With probability π, x is informative, so with probability $\frac{1}{2}+\gamma$ it will be routed to a class 1 leaf. There, with probability $\frac{1}{2}+\pi\gamma$ it will be labeled as belonging to class 1 (based on Lemma 1). With probability $\frac{1}{2}-\gamma$, x will be routed to a class 2 leaf, where it will be labeled as belonging to class 1 with probability $\frac{1}{2}-\pi\gamma$. With probability $1-\pi$, x is uninformative, so it will be routed to a leaf at random and be labeled as belonging to class 1 with probability $1/2$. Combining these quantities, we obtain $P[v=1] =$

$$\pi\left(\frac{1}{2}+\gamma\right)\left(\frac{1}{2}+\pi\gamma\right)+\pi\left(\frac{1}{2}-\gamma\right)\left(\frac{1}{2}-\pi\gamma\right)+\frac{1}{2}(1-\pi) \quad = \quad \frac{1}{2}+2(\gamma\pi)^2$$

Hence, the effective margin of v is $2(\gamma\pi)^2$, which is very small. By summing the count vectors for all of the d detections, the effect (ignoring sampling without replacement) is to take dC trials of this random variable. Applying the Chernoff bound completes the proof. □

It is difficult to compare analytically the bounds in equations 2 and 3. Figure 6 shows the relative performance of the two classification methods for $C=40$, $d=200$. The method of voting evidence performs much better than

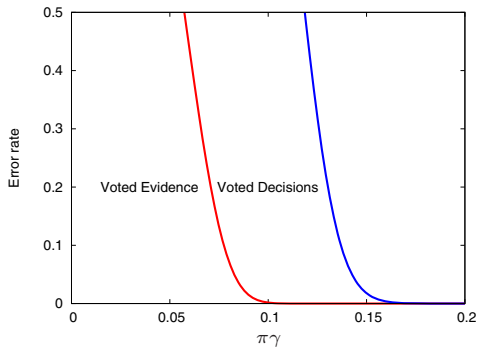

Figure 6. Comparison of error upper bounds for voting evidence vs voting decisions as a function of $\pi\gamma$ for $C = 40$ and $d = 200$.

the method of voting decisions. Notice that the model predicts that neither method would work well in the pure multiple-instance classification case where exactly one of the detections is informative. This would correspond to $\pi = 1/d = 0.005$ and $\pi\gamma = 0.002$ for an excellent $\gamma = 0.4$.

The above analysis was for a single decision tree. Our method grows a random forest of T trees. Assuming each tree is equally accurate, the model predicts that this will not change the margins of the two methods but only the effective sample size in each bound. Hence, the bound for voting the individual classification decisions will become

$$\epsilon_{vd} \leq \exp[-2dT(\pi\gamma(1-2\epsilon))^2]$$

and the bound for combining the counts will become

$$\epsilon_{ve} \leq \exp[-8dCT(\gamma\pi)^4].$$

The model predicts that combining counts will always outperform voting individual decisions. The model can be extended to more than 2 classes by applying Sanov's theorem in place of the Chernoff bounds.

6. Conclusion

We have presented an approach to categorizing highly articulated objects with large intra-category variations and small inter-category differences, where the objects may be only partially visible in images. These challenges have been addressed by: (1) avoiding unsupervised extraction of a visual dictionary, and (2) efficient combining of local texture and global shape properties of objects. The combination of disparate pieces of visual information is done using a two stage classifier. The first level is a random forest trained directly on the descriptors that learns class histograms. We have mathematically proved that this approach is better than using individual voting. The second level is a stacked classifier that combines the information coming from different sources. Experiments conducted on images of insects, as well as real-world images demonstrate validity and generality of our approach.

References

[1] A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. In *ICCV*, 2007.

[2] L. Breiman. Random forests. *Mach. Lrn.*, 45(1):5, 2001.

[3] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Math. Stat.*, 23:493–509, 1952.

[4] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *In Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.

[5] H. Deng, W. Zhang, E. Mortensen, T. Dietterich, and L. Shapiro. Principal curvature-based region detector for object recognition. In *CVPR2007*, pages 1–8, 2007.

[6] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, New York, NY, 1993.

[7] M. Everingham, A. Zisserman, C. K. I. Williams, and L. Van Gool. The PASCAL Visual Object Classes Challenge 2006 (VOC2006) Results. http://www.pascal-network.org/challenges/VOC/voc2006/results.pdf.

[8] F. Jurie and B. Triggs. Creating Efficient Codebooks for Visual Recognition. In *ICCV*, volume 1, 2005.

[9] T. Kadir and M. Brady. Scale, saliency and image description. *IJCV*, 45(2):83–105, 2001.

[10] N. Larios, H. Deng, W. Zhang, M. Sarpola, J. Yuen, R. Paasch, A. Moldenke, D. Lytle, S. Ruiz Correa, E. Mortensen, L. Shapiro, and T. Dietterich. Automated insect identification through concatenated histograms of local appearance features. *Machine Vision and Applications*, 19(2):105–123, 2008.

[11] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[12] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[13] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *ECCV*, pages 128–142, 2002.

[14] F. Moosmann, B. Triggs, and F. Jurie. Fast discriminative visual codebooks using randomized clustering forests. In *NIPS 19*, pages 985–992. MIT Press, Cambridge, MA, 2007.

[15] J. R. Quinlan. *C4.5: Programs for Empirical Learning*. Morgan Kaufmann, San Francisco, CA, 1993.

[16] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *ICCV*, volume 2, pages 1800–1807, 2005.

[17] L. Yang, R. Jin, R. Sukthankar, and F. Jurie. Unifying discriminative visual codebook generation with classifier training for object category recognition. In *CVPR*, 2008.

[18] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: An in-depth study. Technical Report RR-5737, INRIA Rhône-Alpes, Nov 2005.

[19] W. Zhang and T. Dietterich. Learning visual dictionaries and decision lists for object recognition. In *ICPR2008*, pages 1–4, 2008.