

Is a Detector Only Good for Detection?

Quan Yuan

Sony Electronics Inc., San Jose, CA, USA

Quan.Yuan@am.sony.com

Stan Sclaroff

Computer Science Dept., Boston University

sclaroff@cs.bu.edu

Abstract

A common design of an object recognition system has two steps, a detection step followed by a foreground within-class classification step. For example, consider face detection by a boosted cascade of detectors followed by face ID recognition via one-vs-all (OVA) classifiers. Another example is human detection followed by pose recognition. Although the detection step can be quite fast, the foreground within-class classification process can be slow and becomes a bottleneck. In this work, we formulate a filter-and-refine scheme, where the binary outputs of the weak classifiers in a boosted detector are used to identify a small number of candidate foreground state hypotheses quickly via Hamming distance or weighted Hamming distance. The approach is evaluated in three applications: face recognition on the FRGC V2 data set, hand shape detection and parameter estimation on a hand data set and vehicle detection and view angle estimation on a multi-view vehicle data set. On all data sets, our approach has comparable accuracy and is at least five times faster than the brute force approach.

1. Introduction

We consider problems where foreground-background classification (e.g., face detection, human detection) and foreground within-class classification (e.g., face identification, body pose recognition) are both of interest. Many foreground state estimation methods [1, 4, 21] require localization of foreground objects as an essential preliminary step. For such methods, a complete system has two distinct subsystems: a detection subsystem and a foreground within-class classification subsystem.

While detection methods like boosted cascades [27, 33] are very fast, the subsequent foreground within-class classification process can be a performance bottleneck. Consider a face recognition system where each detected face is compared with hundreds or thousands of face IDs. Common methods that employ nearest neighbor [29] or large margin classifiers [7] can be slow. For instance, in our experiments, face ID via one-vs-all (OVA) SVM classifiers of

535 subjects takes more than two seconds per detected face. If we use this system to recognize terrorists at a train station, the detection stage could easily output dozens of faces per second during rush hours. A recognition speed of two seconds/face means a long waiting list of detected faces or dropping detected faces in a real time system.

In this paper, we devise a filter-and-refine strategy [3, 8] to alleviate this critical bottleneck. Our formulation can be employed when the foreground-background classifier subsystem is a boosted-cascade detector. For a given detector output, our method identifies a small number of plausible foreground state hypotheses (filter step). The within-class classification subsystem can then be applied only to evaluate a small set of candidate hypotheses to decide the foreground state (refine step). For instance, for the abovementioned face ID scenario, only a small subset of OVA classifiers would be invoked.

Our approach can be applied to object recognition schemes where multiple hypotheses are examined, for example, multi-class classification processes and nearest neighbor approaches. As demonstrated in the experiments, the proposed filter step can yield an order of magnitude reduction in the number of data base examples to be compared (for nearest neighbor approaches) or the number of within-class classifiers that are invoked (for multi-class classification approaches), with little or no impact on accuracy.

2. Related Work

Our work is related to fast multi-class classification strategies [18]. In [18], a multi-class classifier is constructed by combining binary classifiers in a directed acyclic graph. It employs the same number of binary classifiers as the OVA approach, but each binary classification is much simpler than OVA; therefore it runs faster. However, for n classes, the total number of binary classifiers to be trained is on the order of n^2 , which makes the method impractical for problems with large numbers of classes.

The filter-refine strategy has been used in detection and multi-class classification approaches, e.g., [27, 3]. In [27], a cascade detector is constructed to make object detection

much faster. Trivial background instances are rejected early in the cascade. However, a cascade structured filter step will not have the same advantage for within-class classification, because an input will go through all filter stages anyway in a within-class classification process. In [3], an embedding-based approach was proposed to speed up multiclass classification. Patterns and classes are mapped to vectors in such a way that patterns and their associated classes tend to get mapped close to each other. Thus, an efficient filter step can be employed in the embedded space to identify a small number of candidate classes. This approach can be applied to a variety of multiclass classification problems. However, extra training is needed to learn the embedding in [3], which usually implies a requirement for extra training data. Furthermore, the learned mapping functions need to be calculated using classifiers from the refine stage, which are usually slow in speed.

In another strategy [22], feature reuse has been proposed to make detection processes more efficient. It is shown that reusing features can improve the speed of cascade detectors by 25%. This work speeds up detection, but does not address a subsequent multiclass classification step. Reusing features has an obvious advantage of minimum extra calculations. In our work, we build the connection between detection and foreground within-class classification, which makes it possible to reuse features from detectors for foreground within-class classification.

In addition, there has been previous work [9, 13] that integrates detection and foreground classification, whereby the detection result also reveals the foreground state, *e.g.*, face view angle. The divide-and-conquer mechanism achieves great improvement in detection accuracy. However, to achieve accurate foreground state estimation, fine partitioning of the foreground space is needed; this implies the need for a sufficiently large amount of training data with foreground within-class state annotations to use in training a classifier for each foreground subclass, or a feature sharing approach [23] is necessary.

We should also mention that for foreground state recognition, regression based methods [1, 4] can also be used. Although our approach is not applicable to speeding up regression based methods, it can be applied to alternative methods like the nearest neighbor method, which can solve general foreground state estimation problems.

3. Our Approach

In our work, we assume that the detector is a boosted cascade detector [27, 28, 33]. In addition, we assume that there are foreground within-class classification strategies to rerank foreground state hypotheses at the refine step, *e.g.*, multiclass classifiers of face IDs, a database of annotated foreground examples for a nearest neighbor approach, etc. Our goal is to design a fast filter step to identify a small

number of foreground state hypotheses for a given input. The basic idea is to reuse weak classifier evaluations from the detector.

It may seem surprising that a boosted cascade detector's weak classifiers can also be helpful in foreground within-class classification. Detector training only optimizes accuracy in discriminating foreground vs. background. Yet, as we will soon see, the weak classifier outputs from a boosted cascade detector can be used to construct a Hamming distance that performs well as a filter step for foreground within-class classification. We first show how the cascade detector's weak classifiers are related to locality sensitive hashing (LSH) [10] functions, which enable approximate nearest neighbor search in the feature space. We then show how to construct a Hamming code using a subset of the cascade detector's weak classifier outputs that is optimized for foreground within-class classification.

3.1. From Random Binary Weak Classifiers to LSH

In the traditional Adaboost-based method [20], a strong binary classifier $H(x)$ is constructed as a weighted combination of weak classifiers that are selected from a pool of weak classifiers $h_i(x)$, with corresponding weights α_i :

$$H(x) = \sum_{i=1}^n \alpha_i h_i(x) \quad (1)$$

where $x \in X$ is a feature vector, and $h_i(x) \in \{-1, +1\}$ can be simple decision stumps [27] or linear classifiers [33]. In our approach, each h_i is assumed to be a domain bipartitioning classifier. Therefore, each h_i is equivalent to a hyperplane that divides the feature space into two regions and assigns the input x a binary value $+1$ or -1 , depending on which side of the hyperplane x locates.

We are going to show that those $h_i(x)$ that are random bipartitioning hyperplanes follow the definition of hashing functions in Locality Sensitive Hashing (LSH) [10]. Thus, they can be used to construct a Hamming distance that approximates nearest neighbor search in the Euclidean feature space. In LSH, a family $\mathcal{H} = \{h : X \rightarrow \pm 1\}$ of functions over X is called (r_1, r_2, p_1, p_2) -sensitive for a distance measure D_x , if for any $x_1, x_2 \in X$

- if $D_x(x_1, x_2) \leq r_1$, then $Pr(h(x_1) = h(x_2)) \geq p_1$,
- if $D_x(x_1, x_2) > r_2$, then $Pr(h(x_1) = h(x_2)) < p_2$.

For a locality-sensitive family \mathcal{H} to be useful, it must satisfy $r_1 < r_2$ and $p_1 > p_2$.

We use the following observation: in an Euclidean space, the probability p that two points x_1 and x_2 are separated by a random hyperplane increases monotonically with their Euclidean distance $d = D(x_1, x_2)$. Thus, we have $p = f(D(x_1, x_2))$, where f is a monotonically increasing function and its value is in the range $[0, 1]$.

If we define $h^*(x) = \pm 1$ according to which side of the random hyperplane x is located, we have,

$$\Pr(h^*(x_1) \neq h^*(x_2)) = f(D(x_1, x_2)). \quad (2)$$

Thus, for any $r_1 < r_2$, we have

- if $D(x_1, x_2) \leq r_1$, then $\Pr(h^*(x_1) = h^*(x_2)) = 1 - \Pr(h^*(x_1) \neq h^*(x_2)) \geq 1 - f(r_1)$,
- if $D(x_1, x_2) > r_2$, then $\Pr(h^*(x_1) = h^*(x_2)) = 1 - \Pr(h^*(x_1) \neq h^*(x_2)) < 1 - f(r_2)$.

Let $p_1 = 1 - f(r_1)$ and $p_2 = 1 - f(r_2)$, then we have (r_1, r_2, p_1, p_2) that satisfy $r_1 < r_2$ and $p_1 > p_2$. Therefore, $h^*(x)$ is a valid hashing function for LSH.

We define a binary string representation $B(x)$ as the collection of binary outputs of the weak classifiers:

$$B(x) = \{h_1(x), h_2(x), \dots, h_n(x)\}. \quad (3)$$

We define $D_r(x_1, x_2)$ as the Hamming distance between two binary strings $B(x_1)$ and $B(x_2)$, when h_k are random weak classifiers. Retrieval with D_r and a distance threshold d_H is a special case of LSH, that approximates the nearest neighbor search in the Euclidean feature space.

Although the weak classifiers collected for a detector are not purely random, it has been noticed [27, 24] that in a bootstrap training process of a cascade detector, the background training samples are more and more similar to the foreground samples, as the cascade stage goes deeper and deeper. The weak classifiers tend to have accuracies close to 50%, similar to random partitions. The Adaboost training process also makes the selected weak classifiers less correlated, because a weak classifier selected in an Adaboost iteration focuses more on training examples that cannot be correctly classified in previous iterations. We define D_c as the Hamming distance that uses weak classifiers from the detection stage. In our experiments, filter-refine with D_c achieves retrieval accuracy close to or even better than the Hamming distance D_r that is based on random partitions.

On the other hand, some h_k included in a cascade detector may not be useful for foreground within-class classification. We therefore propose optimization schemes that extract useful h_k from those in a detector for specific within-class classification tasks.

3.2. Optimized Hamming Distance Measure

In this section we propose boosting algorithms to optimize selections of h_k for a specific within-class classification task. The optimized distance measure is a Hamming distance, or a weighted Hamming distance, where each bit is weighted by a real value. Either of these two distance measures can be used in a fast filter step to eliminate implausible foreground state hypotheses quickly.

Intuitively, a good distance measure puts preferable neighboring objects closer to a query than unpreferable ones. For instance, consider continuous parameter estimation problems, like pose estimation [1, 4] or model alignment [15]. These problems can be defined as ranking problems when nearest neighbor approaches [2] or gradient descent methods [30] are applied. When a nearest neighbor approach is used, the parameter of a more preferable neighbor is closer to the true parameter of the query than a less preferable one. Whereas in discrete classification problems, like face recognition [17], the preferable neighbors of a query are those items that have the same class label.

To optimize a distance measure for ranking problems, previous work [2, 30] proposes using triples (q, a, b) as training examples, where q , a and b are samples from the foreground training set. In each triple, a is a more preferable neighbor to q than b . In training, a distance function is optimized to always put a closer to q than b . Another previous work [14] proposes using pairs (q, a) as training examples for discrete classification problems. Each pair (q, a) is assigned a label $+1$ or -1 to indicate whether a is from the same class as q or not. In training, a distance function is optimized to always put pairs of the same class closer than those of different classes. The method of [14] is only relevant to discrete classification. Therefore, we adopt training with triples in our solution, because it can be applied to both parameter estimation and discrete classification problems.

The inputs to our training approach are the following:

1. A training set $S = \{(q_1, a_1, b_1), \dots, (q_t, a_t, b_t)\}$ of t triples of foreground examples. q_i , a_i and b_i are all foreground examples. In each triple, a_i is a more preferable neighbor of q_i than b_i .
2. A set of binary functions $B = \{h_1, \dots, h_n\}$, where $h_k(x) \in \{-1, 1\}$.

Each h_k induces a distance measure

$$d_k(x, y) = |h_k(x) - h_k(y)|/2 \quad (4)$$

and a weak classifier f_k (Note f_k is defined on triples, different from h_k):

$$f_k(q_i, a_i, b_i) = d_k(q_i, b_i) - d_k(q_i, a_i), \quad (5)$$

where $d_k(x, y) \in \{0, 1\}$ and $f_k(q_i, a_i, b_i) \in \{-1, 0, +1\}$. Our goal in training is to find a strong classifier

$$F(q, a, b) = \sum \beta_j f_j(q, a, b), \quad (6)$$

such that $F(q, a, b) > 0$ for all triples (q, a, b) . If we define a new distance measure

$$D_w(x, y) = \sum \beta_j d_j(x, y), \quad (7)$$

and plug Eqn.(5) in Eqn.(6), we have

$$\begin{aligned}
 F(q, a, b) &= \sum \beta_j f_j(q, a, b) \\
 &= \sum \beta_j (d_j(q, b) - d_j(q, a)) \\
 &= D_w(q, b) - D_w(q, a) > 0. \quad (8)
 \end{aligned}$$

Eqn. (8) shows that a F that always assigns a positive value to a triple (q, a, b) implies a perfect D_w that always puts a more preferable neighbor a closer to q than b . Thus, we can obtain an optimized distance measure D_w for a specific foreground classification task. The training process to find optimal β_j and f_j in Eqn. (6) follows a standard Adaboost algorithm. The process stops when no more weak classifiers can be added to reduce the training error. If the same f_j are selected multiple times, their weights are summed to a single β_j to keep all f_j in F distinct.

There is a one-to-one correspondence between f_k and h_k . We call $\hat{B}(x)$ an optimized binary string representation,

$$\hat{B}(x) = \{h_j(x), \text{ where } f_j \text{ is selected for } F\} \subseteq B(x). \quad (9)$$

We call the distance D_w in Eqn. (7) an *optimized weighted Hamming distance*, since each dimension $h_j(x)$ is weighted by a real number β_j .

We are also able to obtain an optimized Hamming distance without real weights β_j . There are only two things that we need to modify in the training process. First, there is a new constraint that $\beta_j = 1$. In each iteration, we select an f_j that reduces the training error most, but fix its weight $\beta_j = 1$. Second, at the end of each boosting iteration, the selected weak classifier f_j is removed from the pool of all weak classifiers for following iterations. We denote this *optimized Hamming distance* as D_h .

The above distance optimization scheme considers only those weak classifiers that were included in the cascade detector. We could instead construct our optimized distance by selecting weak classifiers from the entire set that was available for training the detector. It would be expected that this distance measure might perform better in filter-and-refine retrieval, since distance construction is not limited only to those classifiers included in the detector. We define D_a to be the weighted Hamming distance obtained by selecting a subset from all weak classifiers.

In our experiments, the training process of D_a is very slow. The bottleneck is weak classifier selection in each iteration, as noted in [16, 31]. Speedup strategies [16, 33] that find the best weak classifier deterministically using statistics of training examples cannot be applied, since the same training example can be a in one triple, but b in another triple. On the face data set, we tried a fast feature selection strategy proposed in [31] that stores weak classifier responses of all training samples in a table, which are reused in each iteration. Furthermore, the feature set was reduced

to 1/10 of its original size by uniform sampling. The training process of D_a still runs for about eight hours, in contrast to 25 seconds if we only consider those weak classifiers that were included in a trained detector.

3.3. Implementation

We train a cascade detector of the foreground class by Adaboost. Then, an optimized binary string representation $\hat{B}(x)$ is obtained as described in the previous section.

A table T is constructed to store binary strings $\hat{B}(x)$ of foreground training examples. Each row corresponds to a unique binary string. If multiple foreground training samples have the same binary string, they are stored in the same row, along with the corresponding groundtruth annotations, e.g., face IDs.

During detection, if an input is accepted by the cascade detector, its binary pattern $\hat{B}(x)$ is compared with all the rows in table T by a fast distance measure (D_w or D_h) proposed in Section. 3.2. The foreground state hypotheses associated with top k nearest neighbors (or those within a distance threshold) of the input are passed to the refine step.

The following is a summary of the online stage for the example application of face detection and recognition:

1. **Detect:** x is input to the cascade detector, which uses a standard “sliding window” approach.
2. **Filter:** If x is detected as a face, $\hat{B}(x)$ is compared with all rows in table T by a proposed optimized distance measure (D_w or D_h). Candidate face IDs are those of top k nearest neighbors of x or those within a certain distance threshold from x . These are candidate face IDs for the refine step.
3. **Refine:** Apply OVA classifiers of the candidate face IDs on the corresponding feature representation of x . The face ID of the classifier that achieves the highest score is assigned to the input.

4. Experiments

We evaluate our method on three data sets: the FRGC version 2 data set [17], a hand image data set [32] and a vehicle data set [12]. The experiments are run on a 2.6GHz AMD Opteron 852 processor in Matlab. Approaches that are compared include: our methods (filter-refine using D_w and D_h), ClassMap [3], filter-refine with D_r , D_c and D_a , brute force approaches (OVA classifiers or nearest neighbor), and support vector regression.

4.1. Face Data Set

In this experiment, we use the same face data set as in [3], which contains all 2D face images in the FRGC version 2 data set. Example face images from this data set are shown in Fig. 1. 36,817 face images from 535 subjects (i.e.,



Figure 1. Example face images in the FRGC data set [17].

classes) are partitioned into three subsets, half for training, 1/4 for ClassMap embedding (which is not used in our approach), 1/4 for test. The 535 one-vs-all (OVA) face classifiers are trained using SVMs with RBF kernels as in [3].

We want to mention that nearest neighbor approaches [25, 29] that use similarity functions are also popular methods in practical face recognition systems. Nearest neighbor approaches are better choices than multi-class classifiers when few examples of a face ID are provided in the database. However, on this FRGC version 2 data set, sufficient training examples are provided for most of the face IDs. Thus, a nearest neighbor method will be slower due to a large number of database face images to compare with given an input. We therefore choose an OVA multi-class classification method as a baseline approach.

For comparison on the face data set, the most related works to speed up multi-class classification are DAGSVM [18] and ClassMap [3]. However, for DAGSVM the total number of binary classifiers is too large to train ($\frac{n(n-1)}{2}$ where n is the number of classes). Thus, we compare following seven approaches, brute force where all 535 OVA face ID classifiers are applied on an input face, filter-refine with ClassMap [3], filter-refine with D_w which is the optimized weighted Hamming distance, filter-refine with D_h which is the optimized unweighted Hamming distance, filter-refine with D_c which uses outputs of all weak classifiers in the detector cascade, filter-refine with D_a , which is trained with all possible weak classifiers, and filter-refine with D_r , which is the Hamming distance with random partitions on 50 trials.

The brute force approach takes two steps, face detection followed by face ID recognition. The other approaches take three steps, face detect, face ID filter and face ID refine. In the refine step, only those OVA classifiers for the remaining face IDs from the filter step are applied.

A cascade face detector is trained with 2,500 face images randomly sampled from the training subset. We use the same set of Haar wavelet features as in [27]. The final cascade detector has nine stages and 449 weak classifiers in total. It achieves a detection accuracy of 96% at a false positive rate of 10^{-5} on the test set.

The training set for distance optimization comprises 20,000 triples. For all boosting based methods, the boosting processes stops when the reduction of training error in an iteration is less than the threshold 10^{-4} . In training, 128

Table 1. Comparison of filter step with different distance measures. The filter time is the average per test example.

Distance measure	D_w	D_h	D_c	D_a	D_r
# weak classifiers	128	135	449	115	150
filter time (10^{-3} sec)	7	7	11	6	8
training time	25 s	30 s	N/A	8 h	N/A

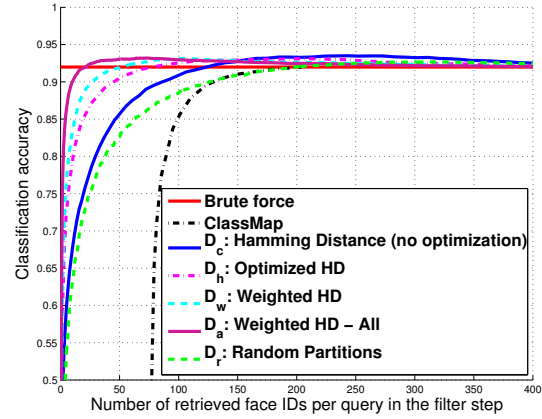


Figure 2. Recognition and retrieval accuracy on the face data set.

weak classifiers are selected for D_w , 135 for D_h and 115 for D_a . For fair comparison, we use 150 random weak classifiers for D_r in each trial. In the filter step, in which nearest neighbor retrieval is employed, each test example is compared with all 18,409 training examples. In Table. 1, the five different distance measures D_w , D_h , D_c , D_a and D_r are compared by number of weak classifiers used and the total time spent in the filter step for all 9,076 test examples. The brute force approach applies all 535 OVA classifiers on a test input. On average it takes 2.17 seconds to classify an input with 535 classifiers.

The graph in Fig. 2 shows the final face recognition results obtained on the face data set. The curve for D_r is the average over 50 trials. At the cost of 50 OVA classifier evaluations per query, filter-and-refine using D_h , D_w and D_a achieves accuracies of 90.5%, 91.8% and 93.0% respectively. In contrast, at the cost of 178 OVA classifier evaluations per query, the ClassMap method achieves an accuracy of 91.6%. The brute force approach that evaluates all OVA classifiers achieves an accuracy of 92.0%. In terms of speed, the methods D_w and D_a are 3.5 times faster than the ClassMap approach with better classification accuracies, and 10 times faster than the brute force approach.

The optimized Hamming distances D_w and D_h are consistently better than the Hamming distance based on random weak classifiers D_r . We also notice that about one third of the random weak classifiers only separate a very small portion of foreground examples from the rest, or do not partition the foreground class at all. This may partially explain why purely random partitions are not as good.

Interestingly, the proposed methods also achieve slightly better accuracies than the brute force approach. For in-

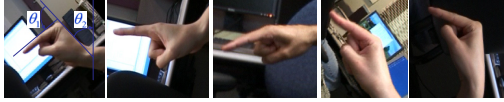


Figure 3. Examples from the hand data set [32].

stance, at the cost of 100 OVA classifier evaluations per query, filter-refine using D_h , D_w and D_a can achieve accuracies of 92.5%, 93.1% and 93.0% respectively. One plausible explanation is that a face misclassified via brute force can be avoided in our filter-and-refine steps if the OVA classifiers producing false alarms are not considered after the filter step. The same effect was also observed in [3].

Although D_a achieves better accuracy, it does not reuse weak classifiers from the detector, and as noted in Sec. 3.2, training is very slow. Moreover, training D_a is intractable when the potential weak classifiers are too many to enumerate, *e.g.*, linear discriminants in a high dimensional Euclidean space as in the following experiments.

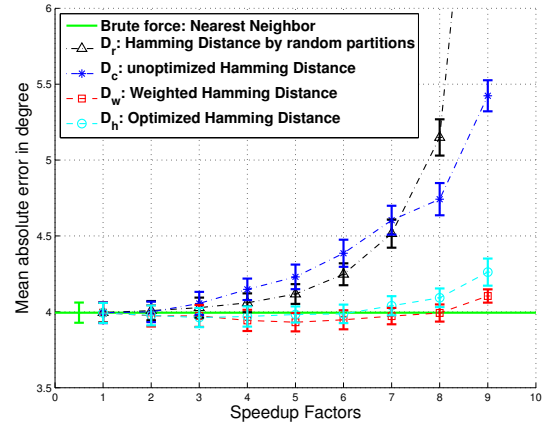
4.2. Hand Image Data Set

The second application is hand detection and hand shape estimation. We use a hand image data set [32] in which the hand shape is parameterized by two angles: θ_1 is the angle of the index finger with respect to the palm and θ_2 is in-plane orientation. $\theta_1, \theta_2 \in [0, 90]$. Example hand images are shown in Fig. 3. In the experiment setup of [32], 1,605 hand images are used for training and 925 for test.

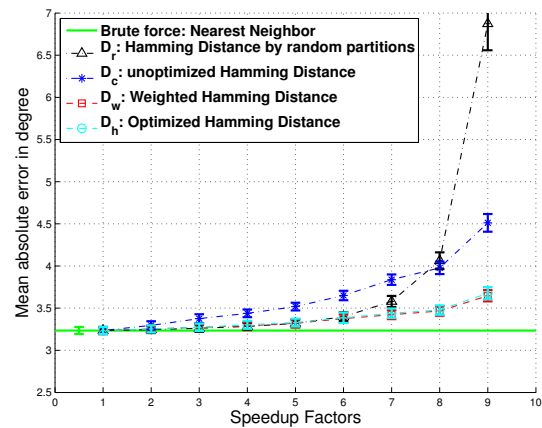
We adopted a two step process to recognize the hand shape. First, a boosted cascade is used to detect the hand. Then nearest neighbor retrieval with Euclidean distance in HOG feature space is used to recover two hand parameters. We use the same HOG features as in [32]. The detector is trained with linear discriminants as weak classifiers, as in [33]. The candidate weak linear discriminants are obtained on subsampled (30%) sets of HOG feature components at each iteration, via Fisher linear discriminant analysis [6].

We randomly partition the hand data (1605+925 examples) into training and test sets for 20 trials. In each trial we train a cascade detector and measure the performance of brute force nearest neighbor retrieval, filter-refine with D_w , D_h , D_c and D_r . All approaches are compared by their average accuracy at different speedup factors. Note ClassMap [3] is not included in this experiment; ClassMap is intended for multi-class classification and inappropriate for parameter estimation.

In each trial, we train two distance measures D_w and D_h , with 20,000 triples. Each training triple (q_i, a_i, b_i) is constructed such that b_i is farther away from q_i than a_i by Euclidean distance in (θ_1, θ_2) space. There is one more constraint that the parameter (θ_1, θ_2) of a_i is within 10 degrees difference from q in each dimension, since it is meaningless to maintain an order between a_i and b_i when they are



(a) Parameter estimation error of θ_1



(b) Parameter estimation error of θ_2

Figure 4. Results of parameter estimation on the hand data set.

both far from q_i . On average, 50 binary weak classifiers are selected for D_w and 53 for D_h . For fair comparison, we randomly sample 50 linear boundaries for D_r and 50 linear weak classifiers from the detector for D_c in each trial.

Fig. 4(a) and Fig. 4(b) show the comparison of parameter estimation errors. At a speedup factor of seven, the proposed approach using D_h obtains mean absolute errors (MAE) 4.0 and 3.4 for θ_1 and θ_2 , respectively. The brute force approach using nearest neighbor retrieval achieves MAEs 4.0 and 3.2 for θ_1 and θ_2 , respectively. The MAEs of D_w and D_h are consistently lower than D_r and D_c at speedup factors greater than two for θ_1 and at speedup factors greater than seven for θ_2 .

Recall that the basic assumption of this work is a two stage process, detection followed by foreground within-class classification. If the foreground within-class classification problem is continuous parameter estimation, a regression based method can be used. For the sake of comparison, we test support vector regression (SVR) [26] from SVMlight [11]. SVR exploits sparsity of the data, so it also has certain advantages in speed. The SVR models use the

Table 2. Mean absolute error (MAE) in degrees, and average filter+refine time spent on each test example, on the hand data set. “RBF” is radial basis function and “Poly2” is polynomial kernel of degree 2. D_w and D_r report MAEs at a speedup factor of seven.

Approach	MAE θ_1	MAE θ_2	Time ($10^{-4}sec$)
SVR-RBF, $\gamma = 0.5$	4.4 ± 0.13	3.4 ± 0.09	35 ± 1
SVR-Poly2	4.5 ± 0.10	3.5 ± 0.09	13 ± 4
SVR-linear	7.1 ± 0.15	5.0 ± 0.12	4 ± 1
Brute force NN	4.0 ± 0.13	3.23 ± 0.08	70 ± 0.2
Filter-Refine D_w	4.0 ± 0.11	3.4 ± 0.11	10 ± 1
Filter-Refine D_h	4.0 ± 0.13	3.4 ± 0.11	10 ± 1



Figure 5. Example images and masks in the data set from [12].

the same training and test sets as our method. The learning parameters (RBF kernel parameter γ , cost upper bound C) are both searched within the range $[10^{-3}, 100]$ via cross validation to find the best setting.

Table. 2 summarizes the performance of all approaches. Compared with the lowest error achieved by SVR, the proposed filter-refine method D_h reduces the error of θ_1 by 0.4 and obtains the same error of θ_2 , while maintaining a speed only slightly slower than SVR with a linear model.

4.3. Vehicle Image Data Set

We also test our method on a multi-view vehicle data set [12], which contains 1,297 vehicle images from the LabelMe [19] database. Each vehicle image has a binary segmentation mask converted from the LabelMe annotation polygon. In [12], the data is split into seven view point sub-categories, approximately 30 degrees apart. Because of vehicle symmetry, the labelled angles cover a half circle from approximately -30 to 180 degrees. For better comparison of view angle estimation accuracy, we manually labelled 472 out of all 1,297 vehicle images 5 degrees apart. We random partition the annotated vehicle images into a test set of 200 and a training set of 272 images in 10 trials. In each trial, a random sample of 700 images from the remaining 1,097 unlabelled vehicle images is added into the detector training set (but not for view angle estimation training).

HOG features are used in this experiment. Each vehicle image is normalized to 90 by 90, which is divided into 225 cells of size 6 by 6. Bins in each cell are normalized with the surrounding 3 by 3 cells using the 2-norm as in [5]. There are 2,025 features extracted from each image.

A cascade detector is trained in the same way as in the hand experiment in each trial, where linear discriminants are used as weak classifiers. On average the cascade detector has 480 weak classifiers in total.



Figure 6. Example results of view angle estimation by HOG feature matching. Test inputs are in the top row, and corresponding nearest neighbors from training images are in the bottom row. The rightmost three pairs are incorrect matches.

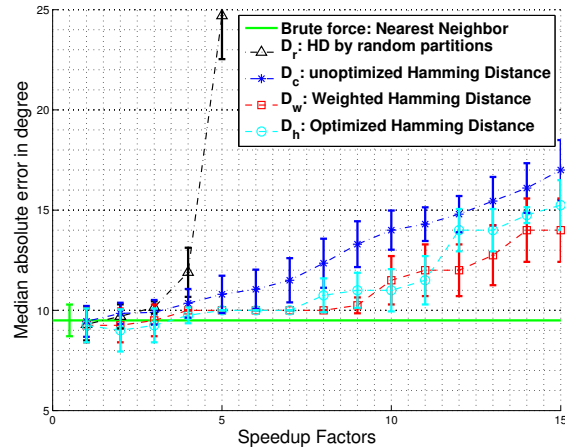


Figure 7. Comparison of different distance measures on car view angle estimation accuracy vs speedup factors.

To estimate the view angle of a detected vehicle, we use a simple nearest neighbor approach. The similarity measure is the dot product between HOG feature vectors of two examples. Vehicle masks of training examples are used to zero out feature components outside hypothetical foreground regions. The dot product is normalized by the number of actual vector components that are inside the mask. With this similarity measure, the view angle of the nearest annotated training example is assigned to the test input as an angle estimate. Example matching results are shown in Fig. 6.

In our approach, we add a filter step to speed up the view angle estimation process by selecting candidate training examples before HOG feature matching. D_w and D_h are trained with 5,000 triples of annotated training examples. Each triple (q, a, b) is constructed such that a is closer to q on the view angle axis than b , and a is within 10 degrees from q . During boosting based optimization in 10 trials, on average D_w added 44 weak classifiers and D_h added 47 weak classifiers. For fair comparison, D_r and D_c uses 45 weak classifiers in each trial.

Because there exists strong confusion between frontal and rear views of vehicles, there is a spike around 180 degrees in the distribution of absolute errors, which dominates mean of the absolute errors (MAE). For better understanding of the errors, we measure the median of absolute errors (Median-AE) at different speedup factors in each trial. In Fig. 7, distance measures D_w , D_h , D_c and D_r are compared with brute force nearest neighbor approach on aver-

Table 3. Median of absolute error (Median-AE) in degrees and the total filter+refine time spent on 200 test examples. D_w and D_r report Median-AEs at a speedup factor of ten.

Approach	Median-AE	Time ($10^{-2}sec$)
SVR-Poly2	25.7±2.7	11.0±1.6
SVR-RBF $\gamma = 0.01$	24.2±1.8	10.3±0.7
Brute force NN	9.5±0.8	12±0.3
Filter-refine D_w	11.5±1.2	1.2±0.1
Filter-refine D_h	11.0±1.1	1.2±0.1

age Median-AE vs speedup factors. Note the results are averages over 10 trials. The brute force approach achieves an average Median-AE of 9.50 degrees. The proposed filter-refine approach using D_w and D_h achieve average Median-AEs of 11.5 and 11.0 respectively, at a speedup factor of 10. In contrast, the filter-refine approach with D_r which uses random partitions achieve an average Median-AE of 25.0, at a speedup factor of 5.

We also test the SVR methods on view angle estimation. Unlike the HOG feature matching approach, regression methods (e.g., SVR) require that all inputs have the same feature dimensions. There is no straightforward way to apply image masks with regression methods. Consequently, the features from background regions outside the image masks are also included during training, which becomes a major disadvantage for regression methods on this data set. In Table. 3, we summarize the performance of SVR methods, in comparison with the proposed approaches. Filter-refine with D_w and D_h reduce Median-AE by half with a speedup factor of about nine over the SVR approaches.

5. Discussion

In the experiments, filter-refine using the optimized Hamming distances (D_w and D_h) constructed from weak classifier outputs of a boosted cascade detector does better than random partitions D_r . This seems to indicate that these weak classifiers, while explicitly chosen to optimize foreground-background discrimination, are also relevant for foreground within-class classification. In comparison with D_r , our formulation improves recognition accuracy by about 10% with a speedup factor of ten on the face data set, and reduces parameter estimation error by at least 15% at speedup factors larger than seven on the hand and vehicle data sets.

An interesting side-effect noticed in the experiments is that the foreground within-class classification accuracy can be improved over the brute force approach by including the filter step. One possible explanation is that those foreground state classifiers that produce the false positives are removed in the filter step, which is also noted in the filter step of [3].

6. Acknowledgments

This material is based upon work supported in part by U.S. National Science Foundation Grant IIS-0705749.

References

- [1] A. Agarwal and B. Triggs. 3D human pose from silhouettes by relevance vector regression. In *CVPR*, 2004.
- [2] V. Athitsos, J. Alon, S. Sclaroff, and G. Kollias. Boostmap: A method for efficient approximate similarity rankings. In *CVPR*, 2004.
- [3] V. Athitsos, A. Stefan, Q. Yuan, and S. Sclaroff. Classmap: Efficient multiclass recognition via embeddings. In *ICCV*, 2007.
- [4] A. Bissacco, M. Yang, and S. Soatto. Fast human pose estimation using appearance and motion via multi-dimensional boosting regression. In *CVPR*, 2007.
- [5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.
- [6] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179 – 188, 1936.
- [7] B. Heisele, P. Jo, and T. Poggio. Face recognition with Support Vector Machines: Global versus component-based approach. In *ICCV*, 2001.
- [8] G. Hjaltason and H. Samet. Properties of embedding methods for similarity searching in metric spaces. *PAMI*, 25(5):530 – 549, 2003.
- [9] C. Huang, H. Ai, Y. Li, and S. Lao. Vector boosting for rotation invariant multi-view face detection. In *ICCV*, 2005.
- [10] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, 1998.
- [11] T. Joachims. Making large-scale SVM learning practical. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.
- [12] B. Leibe, N. Cornelis, K. Cornelis, and L. V. Gool. Dynamic 3D scene analysis from a moving vehicle. In *CVPR*, 2007.
- [13] S. Li, L. Zhu, Z. Zhang, A. Blake, and H. Shum. Statistical learning of multi-view face detection. In *ECCV*, 2002.
- [14] S. Mahamud, M. Hebert, and J. Shi. Object recognition using boosted discriminants. In *CVPR*, 2001.
- [15] I. Matthews and S. Baker. Active appearance models revisited. *IJCV*, (2):135 – 164, 2004.
- [16] M. Pham and T. Cham. Fast training and selection of Haar-like features for face detection. In *ICCV*, 2007.
- [17] P. J. Phillips, P. J. Flynn, T. Scruggs, K. W. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min, and W. Worek. Overview of the face recognition grand challenge. In *CVPR*, 2005.
- [18] J. C. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification. In *NIPS*, pages 547–553, 1999.
- [19] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. LabelMe: A database and web-based tool for image annotation. Technical report, MIT, 2005.
- [20] R. Schapire, Y. Freund, P. Bartlett, and W. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *ICML*, 1997.
- [21] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *ICCV*, 2003.
- [22] J. Sochman and J. Matas. Inter-stage feature propagation in cascade building with Adaboost. In *ICPR*, 2004.
- [23] A. Torralba, K. Murphy, and W. Freeman. Sharing features: Efficient boosting procedures for multiclass object detection. In *CVPR*, 2004.
- [24] Z. Tu. Learning generative models via discriminative approaches. In *CVPR*, 2007.
- [25] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.
- [26] V. Vapnik, S. Golowich, and A. Smola. Support vector method for function approximation, regression estimation, and signal processing. In *NIPS*, 1997.
- [27] P. Viola and M. Jones. Robust real time object detection. *IJCV*, 57(2):137 – 154, 2004.
- [28] P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *IJCV*, volume 63, pages 153–161, 2005.
- [29] L. Wiskott, J. Fellous, N. Krger, and C. von der Malsburg. Face recognition by elastic bunch graph matching. *PAMI*, 19(7):775 – 779, 1997.
- [30] H. Wu, X. liu, and G. Doretto. Face alignment via boosted ranking models. In *CVPR*, 2008.
- [31] J. Wu, S. C. Brubaker, M. D. Mullin, and J. M. Rehg. Fast asymmetric learning for cascade face detection. *PAMI*, 30(3):369–382, 2008.
- [32] Q. Yuan, A. Thangali, V. Ablavsky, and S. Sclaroff. Multiplicative kernels: Object detection, segmentation and pose estimation. In *CVPR*, 2008.
- [33] Q. Zhu, S. Avidan, M. Yeh1, and K. Cheng. Fast human detection using a cascade of histograms of oriented gradients. In *CVPR*, 2006.