Incremental Multiple Kernel Learning for Object Recognition

Aniruddha Kembhavi[†,*], Behjat Siddiquie[†], Roland Miezianko[§], Scott McCloskey[§], Larry S. Davis[†]

† University of Maryland, College Park § Honeywell Labs

Abstract

A good training dataset, representative of the test images expected in a given application, is critical for ensuring good performance of a visual categorization system. Obtaining task specific datasets of visual categories is, however, far more tedious than obtaining a generic dataset of the same classes. We propose an Incremental Multiple Kernel Learning (IMKL) approach to object recognition that initializes on a generic training database and then tunes itself to the classification task at hand. Our system simultaneously updates the training dataset as well as the weights used to combine multiple information sources. We demonstrate our system on a vehicle classification problem in a video stream overlooking a traffic intersection. Our system updates itself with images of vehicles in poses more commonly observed in the scene, as well as with image patches of the background, leading to an increase in performance. A considerable change in the kernel combination weights is observed as the system gathers scene specific training data over time. The system is also seen to adapt itself to the illumination change in the scene as day transitions to night.

1. Introduction

The problem of visual category recognition has received considerable interest over the past few years. The most common approach consists of three major components: interest point detection, interest region description and classification. A recent focus has been on improving region descriptors. This has led to a number of powerful descriptors being proposed such as Histograms of Oriented Gradients [8], Geometric Blur [3] and Pyramidal Histogram of Visual Words [6]. While each of these descriptors provides good classification accuracies for different object classification tasks, combining information from such multiple sources has been shown to be more reliable [5, 24, 19]. Varma et al. [19] proposed combining multiple descriptors using Multiple Kernel Learning (MKL) and showed impressive results on varied object classification tasks.

Using such a set of powerful descriptors, along with a nonlinear classifier such as a Support Vector Machine (SVM), can lead to a boost in classification performance.

*Corresponding author: anikem@umd.edu

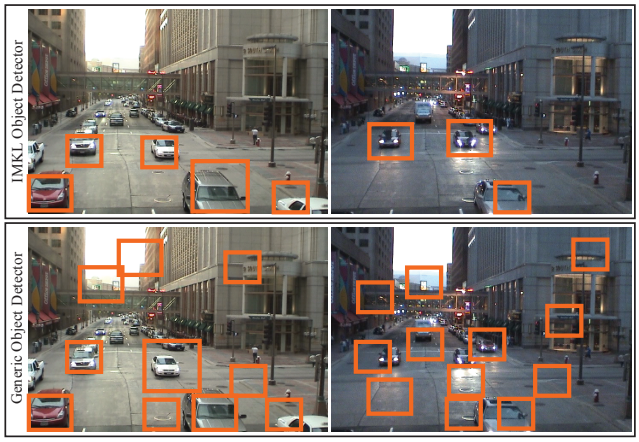


Figure 1. Sample result frames showing varying illumination conditions. Our incremental framework (IMKL) tunes itself to the scene by updating itself with images of objects in commonly observed poses and images of the varying background. Thus, it outperforms a static detector built on a generic training set.

But it is equally important to have a good set of training images, representative of the test images that are expected in the given application. Collecting large number of images and forming a generic training dataset for commonly seen objects is relatively easy using an internet search engine such as Google. Furthermore for many standard objects such as cars, training datasets are already available, such as the UIUC Car Database [1]. However, obtaining a representative training database for a given application is not as straightforward, as it requires a fair amount of manual labor.

Consider a camera at a traffic intersection detecting and classifying vehicles such as shown in Figure 1. First, the location of the camera in this scene and typical paths traversed by the vehicles, restricts the observed poses. Second, the camera position restricts the images representing the negative class (in our case, the background images) for this classification task. Third, images corresponding to vehicles as well as background also change over time, due to factors such as illumination changes and shadows cast by the nearby buildings. Obtaining such scene specific exam-

ples of the object classes and the background class would clearly benefit the visual classifier, but would require a tedious manual annotation procedure.

Our Incremental Multiple Kernel Learning (IMKL) approach uses an easily obtained generic training database as input, and then tunes itself to the classification task at hand. It simultaneously updates the training examples to tailor them towards the objects in the scene. It also updates the weights that determine the optimal combination of different information sources[1], while allowing different combinations to be chosen for different object classes. Finally, it tunes the classifier to the updated training dataset. As the scene changes over time, a feedback loop updates our training dataset with detections from all object classes. The incremental procedure is then invoked to update the kernel combination weights as well as the classifier. Our final system is obtained by combining the outputs of this online classifier with the high probability outputs of the original offline classifier trained on the generic training database. This enables us to tune the classifier to the given scene, while reducing the number of misclassifications on rarely seen objects. We can also remove images from our training database over time. This is useful when dealing with gradual illumination changes, for example.

We first describe the MKL formulation of Bach et al. [15], known as SimpleMKL, which we use to obtain a classifier for the initial training database. SimpleMKL carries out this optimization in an SVM framework to simultaneously learn the SVM model parameters as well as kernel combination weights. Our incremental procedure for MKL is an exact online solution that allows us to update the Lagrangian multipliers of the training points, as well as the kernel combination weights, one new point at a time. The central idea is to add a new data point to the solution and update its Lagrangian multiplier while maintaining the Karush-Kuhn-Tucker conditions on all the current data points in the solution. We derive our IMKL procedure in Section 3.

We demonstrate our visual categorization framework on the task of vehicle detection and classification. The dataset we use consists of video sequences collected from a camera overlooking a traffic intersection. We initialize our training database with a set of images collected from Google and update it incrementally to improve the classification performance over time. The dataset also shows a significant change in illumination conditions in the scene as day transitions into night. Our system is able to update itself over time to handle this transition. We compare our algorithm with OPTIMOL [13], an incremental model learning approach, recently proposed for the task of automatic object dataset collection.

2. Related Work

Early works on object recognition used global features such as color or texture histograms [14]. However these features were not robust to view-point changes, clutter and occlusion. Over the years, more sophisticated approaches such as *part-based* [9] and *bag-of-features* [16] methods have become more popular.

Increased interest in object recognition has resulted in new feature descriptors and a multitude of classifiers. Inspired by the pyramidal feature matching approach of [12], Bosch et al. proposed two new region descriptors - the Pyramidal Histogram of Oriented Gradients (PHOG) and Pyramidal Histogram of Visual Words (PHOW) [6]. These features were then used with Random Forests as a multiway classifier [5]. Zhang et al. used the Geometric Blur (GB) feature [3] and proposed using a discriminative nearest neighbor classification for object recognition [23]. Wu et al. [21] used edgelet features to capture the local shape of objects and were able to simultaneously detect and segment objects of a known category.

Zhang et al. [24] combined multiple descriptors and obtained improved results for texture classification and object recognition. They provided equal weights to each descriptor. Similarly, Bosch et al. [5] linearly combined the PHOG and PHOW descriptors to obtain improved performance. The linear combination weights were, however, obtained by a brute force search using a validation dataset. Since the number of features was small, their search space had few dimensions, thus making the brute force computationally feasible. Wu et al. [22] combined multiple heterogeneous features for object detection by using cascade structured detectors in a boosting framework. Features were combined using their classification powers and computational cost.

Lanckriet et al. [11] introduced the MKL procedure to learn a set of linear combination weights, while using multiple sources of information with a kernel method, such as an SVM. Their problem formulation, however, resulted in a convex but non-smooth minimization problem. Bach et al. [2] considered a smoothed version of the problem. Their Sequential Minimal Optimization (SMO) algorithm was significantly more efficient than the previous formulation in [11]. Sonnenburg et al. [17] reformulated the problem as a semi-infinite linear program and solved it efficiently by recycling the standard fast SVM implementations. Their algorithm worked for hundreds of thousands of examples or hundreds of kernels. Rakotomamonjy et al. [15] formulated the problem using a 2-norm regularization formulation to a smooth and convex optimization problem. Their method provided the additional advantage of encouraging sparse kernel combinations. Varma et al. [19] combined multiple features using MKL and showed a considerable increase in the performance of their visual classifier.

A number of unsupervised, online learning algorithms have been used for computer vision applications. Li et al. [13] used a non-parametric graphical model in an incremental approach for automatic dataset collection from the Internet (OPTIMOL). Their iterative framework simultaneously learns object category models and collects object category datasets. We compare our IMKL method with OP-

[1] In this paper each information source refers to a kernel matrix.

TIMOL in Section 5. Boosting techniques for incremental learning have also been popular. Javed et al. [10] used *co-training* to label incoming data and used it to update a boosted classifier. Co-training [4] is a method for training a pair of learners, given that the two algorithms use different *views* of the data. The two classifiers are used to provide additional informative labeled examples to one another, which improves the overall performance. Wu et al. [20] extended the online boosting algorithm and proposed an online framework for cascade structured detectors. An automatic labeler called the *oracle*, with a high precision rate, provided samples to update the online object detector. In order to prevent the boosting algorithm from overfitting noisy data (provided by the *oracle*), they employed two noise resistant strategies from variants of the Adaboost algorithm designed to be robust to outliers. Our initial object classifier, built from a generic training dataset, is tuned similar to this *oracle*. Our work builds on MKL and fits well into the SVM framework. It also provides the useful property of being able to adapt kernel weights over time in addition to updating the training database.

3. An Incremental Solution

3.1. The Multiple Kernel Learning Problem

Kernel based learning methods have proven to be an extremely effective discriminative approach to classification as well as regression problems. Given multiple sources of information, one might calculate multiple basis kernels, one for each source. In such cases, the resultant kernel is often computed as a convex combination of the basis kernels,

$$\Phi(x_i, x_j) = \sum_{k=1}^{K} d_k \Phi_k(x_i, x_j), \quad \sum_{k=1}^{K} d_k = 1 \quad , d_k \geq 0 \qquad (1)$$

where x_i are the data points, $\Phi_k(x_i, x_j)$ is the k^{th} kernel and d_k are the weights given to each information source (kernel). Learning the classifier model parameters and the kernel combination weights in a single optimization problem is known as the Multiple Kernel Learning problem [11]. There have been a number of formulations for the MKL problem, as noted in Section 2. Our incremental approach builds on the MKL formulation of [15], known as SimpleMKL. This formulation enables the kernel combination weights to be learnt within the SVM framework. The optimization equation is given by,

$$\begin{aligned} \min \quad & \sum_k \frac{1}{d_k} w_k w_k^T + C \sum_i \xi_i \\ \text{such that} \quad & y_i \sum_k \phi_k(x_i) + y_i b \geq 1 - \xi_i \ \ \forall i \\ & \xi_i \geq 0 \ \ \forall i, \ \ d_k \geq 0 \, \forall k, \ \ \sum_k d_k = 1 \end{aligned} \qquad (2)$$

where b is the bias, ξ_i is the slack afforded to each data point and C is the regularization parameter. The solution to the above MKL formulation is based on a gradient descent on the SVM objective value. An iterative method alternates between determining the SVM model parameters using a standard SVM solver and determining the kernel combination weights using a projected gradient descent method.

3.2. Karush-Kuhn-Tucker Conditions

The support vectors returned by the training algorithm of an SVM generally represent a small fraction of all the training examples, but are able to summarize the decision boundary between the classes very well. Thus, one way to increment an SVM is to retain only the support vectors, to reduce the computational load required at every successive training step [18]. The same approach could be used for the MKL problem. However, this gives only approximate results.

The first exact online approach to train SVMs was proposed by Cauwenberghs et al. [7]. New data points are presented to the SVM one at a time. The new data point is added to the solution while ensuring that the Karush-Kuhn-Tucker (KKT) conditions are retained on all the previous data points. Our proposed approach to IMKL is inspired by this work.

The key idea behind the Incremental SVM is that the SVM optimization problem is convex. Thus, the KKT conditions are not only *necessary* but also *sufficient*. Thus, maintaining the KKT conditions on all old points, as well as the new point, indicates that a new solution has been obtained. The optimization problem given by the SimpleMKL framework in Equation 2 is also convex, making it suitable for our purposes.

The KKT conditions for our problem are derived from the Lagrangian function corresponding to Equation 2,

$$\begin{aligned} L = \frac{1}{2} \sum_k \frac{w_k w_k}{d_k} + C \sum_i \xi_i - \sum_i \nu_i \xi_i - \mu_k d_k - \\ \sum_i \alpha_i (y_i w_k \phi_k(x_i) + y_i b - 1 + \xi_i) - \lambda (\sum_k d_k - 1) \end{aligned} \qquad (3)$$

where α_i is the Lagrange multiplier corresponding to the first constraint in Equation 2, ν_i and μ_k are the Lagrange multipliers associated with the non-negativity constraints on ξ_i and d_k respectively, while λ corresponds to the Lagrange multiplier of the l_1-norm equality constraint on d.

The optimal solution of the multiple kernel system in Equation 2 occurs at the saddle point of Equation 3. The saddle point is obtained by differentiating the Lagrangian equation with respect to the primal variables (w_k, d_k, ξ_i, b) and the dual variables (α_i, ν_i, μ_k). A small amount of algebraic manipulations yields the KKT conditions given below,

$$\begin{aligned} g_i = \sum_j \sum_k d_k \alpha_j Q_{ij}^k + y_i b - 1 = 0, \quad \sum_i \alpha_i y_i = 0 \\ \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j Q_{ij}^k + \mu_k - \lambda = 0, \quad \mu_k d_k = 0, \quad \sum_k d_k = 1 \end{aligned} \qquad (4)$$

where $Q_{ij}^k = y_i \Phi_k(x_i, x_j) y_j$.

Note that $g_i = y_i f(x_i) - 1$, where $f(x_i)$ is the solution of the multiple kernel SVM given by,

$$f(x_{new}) = \sum_j \sum_k d_k \alpha_j y_j \Phi_k(x_j, x_{new}) + b \qquad (5)$$

3.3. Algorithm

Consider a set of data instances $(x_1, x_2, \ldots, x_n)$ with corresponding class labels $(y_1, y_2, \ldots, y_n)$. Let $\Phi_k(x_i, x_j)$ be the set of K kernels. The MKL solution for the given data is obtained by SimpleMKL and it thus satisfies the KKT conditions in Equation 4. The data points are divided into three disjoint sets based on their Lagrange multipliers ($\alpha_i's$): set L containing the set of points lying on the correct side of the margin vectors ($\alpha_i = 0$), set S containing the support vectors ($0 < \alpha_i < C$) and set E containing the points lying on the wrong side of the margins ($\alpha_i = C$). We also divide the kernels into two sets: set D_+ containing kernels with positive weights and set D_0 with kernels having zero weight. These sets are illustrated in Figure 2.

When a new point x_q is added to the solution, we need to calculate its Lagrange multiplier α_q ($0 \leq \alpha_q \leq C$) such that the KKT conditions are satisfied once again. We begin with a value $\alpha_q = 0$ and keep increasing it until we reach the updated solution. Every time we increment α_q, the remaining Lagrangian multipliers, the kernel weights and the bias must be changed to maintain the constraints in Equation 4. These changes are given by the differential form of the constraints,

$$\sum_j \alpha_j \sum_k \Delta d_k Q_{ij}^k + \sum_j \Delta\alpha_j \sum_k Q_{ij}^k + \sum_j \sum_k \Delta d_k \Delta\alpha_j Q_{ij}^k + y_i \Delta b = 0, \ \forall i \in S, \forall j \in \{S, E, L, q\}$$

$$\sum_i \sum_j \Delta\alpha_i \alpha_j Q_{ij}^k + \frac{1}{2} \sum_i \sum_j \Delta\alpha_i \Delta\alpha_j Q_{ij}^k \tag{6}$$

$$+\Delta\mu_k - \Delta\lambda = 0, \qquad \forall k \in K$$

$$\sum_i \alpha_i y_i = 0 \forall i \in \{S, E, L, q\}, \quad \sum_k \Delta d_k = 0$$

$$\Delta\mu_k d_k + \mu_k \Delta d_k + \Delta\mu_k \Delta d_k = 0, \qquad \forall k \in K$$

For a given step size $\Delta\alpha_q$, Equation 6 is a set of $(num_S + 2K + 2)$ equations in $(num_S + 2K + 2)$ unknowns. Here, num_S is the number of points in set S and K is the number of kernels. The unknown variables are: $\{\Delta\alpha_1 \ldots, \Delta\alpha_{num_S}, \Delta d_1, \ldots, \Delta d_K, \Delta\mu_1, \ldots, \Delta\mu_K, \Delta b, \Delta\lambda\}$. These non-linear equations can be solved using a standard non-linear equation solving package. Since an addition of a new point may not alter the system significantly, a good initial solution for all the unknowns in Equation 6 is 0.

The above differential equations only hold when $\Delta\alpha_q$ is small enough to ensure that there is no change in set membership for either the points or the kernels. Thus, when set membership changes, the differential equations are updated and the process is repeated. The conditions for a change in the set membership are described in Figure 2.

The algorithm is terminated when any of the following conditions occur.

- $g_q > 0$ at $\alpha_q = 0$: x_q is a correctly classified point. Added to set L.
- $g_q = 0$ before $\alpha_q = C$: x_q is a support vector. Added to set S.

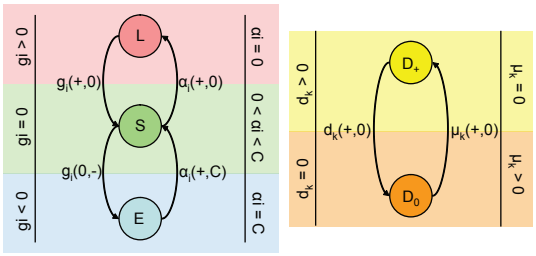


Figure 2. Categorization of the data points and kernels. The image on the left shows the values of the Lagrange multipliers ($\alpha's$) and the output of the system ($g's$) for each of the sets: L, S and E. It also shows the conditions that are checked to detect a set transition. (Notation: $g_i(+,0)$ denotes the value of g_i changing from a positive value to 0.) The image on the right shows the two kernel sets, the corresponding values of the weights ($d's$) and their Lagrange multipliers ($\mu's$) and the set change conditions.

- $\alpha_q = C$ and $g_q < 0$: x_q is on the wrong side of the margin. Added to set E.

A similar procedure can be used for removing data points from the classifier (decremental unlearning).

The number of computations required by the IMKL algorithm depends on the computations to solve the non-linear system and the number of steps taken to reach the final value of $\Delta\alpha_q$. In our experiments, we have observed that setting the initial solution of the non-linear solver to a zero vector, reduces the computational cost significantly. The number of steps taken to reach the final solution is lower bounded by the number of set changes that are required to arrive at the final solution. We use a large step size at every time instant and backtrack our solution if we observe a set change for the given step size. The IMKL algorithm can also be sped up by ignoring the higher order terms in Equation 6 to obtain linear equations. However this provides only an approximate solution.

Consider the two class classification problem shown in Figure 3. A new point q, marked in red, is added to the system, and it initially gets misclassified. As the Lagrange multiplier α_q is incremented upwards from a value of 0, the distance between the new point and the margin reduces, while some of the other points change set membership. At the same time, the kernel combination weights also change.

4. Object Recognition Framework

A training database, representative of the expected test points, is an essential component of any classification system. In a practical object recognition framework, a good training database is one that contains images of the expected objects in their more likely poses and illumination conditions. It must also contain a representative set of images in the negative set, which, in an object recognition framework, is usually the background. Obtaining such a set of good training examples can often be a tedious process. On the other hand, it is easier to obtain a generic training dataset of images of the expected object classes. Our object detector

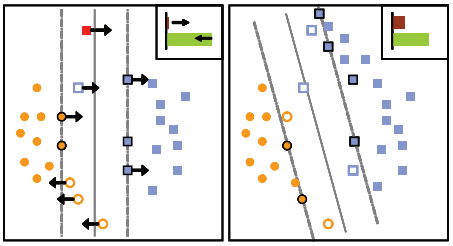


Figure 3. A 2-class classification example. Points in class 1 are shown in orange and points in class 2 are shown in blue. Points in set S are marked with a black border. Points in set L are solid colored while points in set E are not filled with color. Kernel 1 (weight shown by the brown bar) captures the similarity between the y-coordinates of the points, while Kernel 2 (green bar) captures the similarity between the x-coordinates. The left figure shows the effect of adding a new point (shown in red) on the original points and the weights. A change in set membership is observed for some points. The figure on the right shows the final classifier after adding 7 new points close to the first new point.

is initialized on a generic training dataset and tunes itself towards the objects and background in the scene.

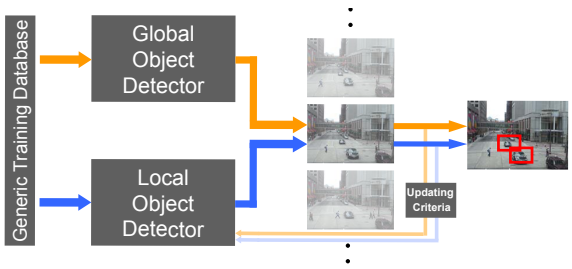


Figure 4. Object recognition framework.

Figure 4 provides an overview of our visual categorization framework. Training images from a generic training dataset are used to train an initial object detector which we call the *global* detector. The *global* detector is not updated at any time and serves as a generic object classifier. The generic training dataset is also used to train a *local* object detector, which runs in an online mode throughout the duration of analysis. Incoming images from a video stream are scanned using overlapping windows and each window is classified into one of the classes by both the detectors. The classification results returned by the global detector keep updating the training image sets of the local object detector.

The updating criterion differs for the foreground classes (buses and cars) and the background class. The image windows that are classified by the *global* detector as belonging to one of the foreground classes are thresholded so as to retain only very high confidence detections. Such windows are considered reliable detections and used to update the foreground training sets of the *local* object detector. Since the purpose of the local object detector is to train on typically observed appearances and poses, updating it with high confidence samples works well. The high precision of the *global* detector comes at the cost of a lower recall. Updating the *local* detector with false positives can lead to a significant drop in the performance of the system, and the probability threshold is set sufficiently high to minimize this.

On the other hand, for the background class, such an updating criterion leads to the addition of a large number of image patches from a single portion of the scene. This is because background patches with very similar appearances repeat over several frames. Thus if a patch gets classified with a very high probability of belonging to the negative set, several similar images also get added to the *local* training set. Ideally, one would like the entire scene to be well represented in the background class of the local detector. Thus, we first threshold image windows classified by the *global* detector as belonging to the background class. Then, for every image patch passing this initial criterion, we evaluate its positional entropy with respect to the distribution of the positions of all image patches currently in the *local* background training set. This is given by,

$$H(I) = - \sum_{w \in \{BG_{local}\}} p(w_{(x,y)}|I_{(x,y)}) \log p(w_{(x,y)}|I_{(x,y)}) \tag{7}$$

where w represents an image patch in the current background set, I represents the new image patch and (x, y) represent the co-ordinates of an image patch in the scene. Image patches passing the initial background threshold, as well as having a high entropy with respect to the current local training set, form good candidates to improve the diversity of the *local* background set and are added to it. Over time, the object classes get updated with images of objects in their typical observed appearances and poses and the background class gets updated with image patches from different parts of the scene. Figure 5 demonstrates the image patches in the *local* background set which has been updated using both criteria. Using the entropy criteria in addition to a probability threshold, samples the entire scene well. Li et al. [13] used a similar criteria to update their dataset. While their entropy is calculated in the feature space, our measure is calculated in the image co-ordinate space.

The *local* detector fits itself towards image patches observed in the recent past, improving its performance. However, it also has the tendency of misclassifying objects that are atypical in the scene, due to overfitting on the observed data. The more generically trained *global* detector helps classify such atypical objects. The outputs of both detectors are combined to obtain the final detections. The resultant object detections are used to update the *local* detector.

In order to fit the local detector towards a dynamically changing scene, it is also important to discard image patches from the local training dataset. For every image patch added to the local set, we retain a timestamp indicating the frame it was obtained from. We use this to discard training samples based on the length of their stay in the training set. Thus the classifier adapts itself towards changing illumination conditions, particularly when day transitions to night.

Our IMKL algorithm described in Section 3 is used to update the *Local* classifier with new training images. This

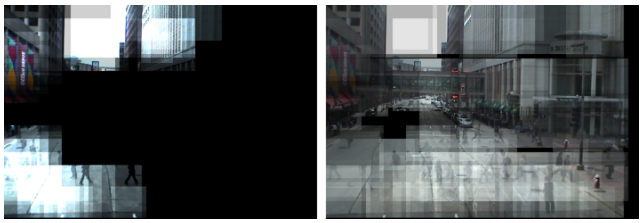


Figure 5. Representation of the *local* negative training set using two sampling methods to update the training set. For this display, all image patches in the set are added together at the appropriate locations in the scene. Thus brighter regions corresponds to more patches in that portion of the scene, black regions indicate that no image patches represent that portion of the scene. (Left) High probability criteria - Only certain portions of the scene are represented. (Right) High probability + high entropy criteria - Most portions of the scene are represented equally.

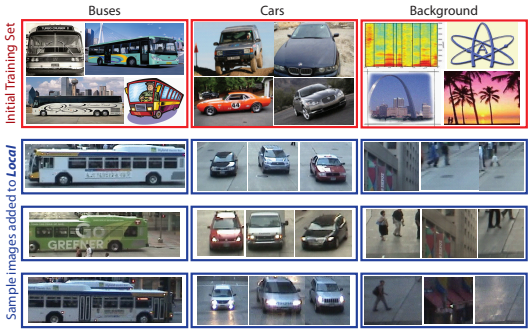


Figure 6. Snapshots of the training set at 4 time instants. Top row shows the initial training set. The next 3 rows show sample images added to *local* over time. The illumination change is noticeable at each time instant. The dataset gets updated with many objects in similar poses and representative background patches.

also results in an update of the kernel combination weights based on the training data. We use multiple 1-Vs-All classifiers for our purpose of multi-class classification. This enables us to compute a separate set of kernel combination weights, one for each object class. In Section 5 we show an example of the evolution of these kernel weights over time.

5. Experiments

We test the performance of our system on the task of object detection on videos taken from a traffic dataset. This dataset consists of 11 challenging videos (480 x 704 pixels at 15 frames/second), of a busy intersection, taken from a traffic surveillance camera. The total number of frames is more than 120,000. Our task is to detect two classes of objects, cars and buses. We have ground truth marked for every tenth frame in this dataset.

Due to the camera location and traffic restrictions in the scene, cars in the video typically have a frontal view, while buses typically appear in a profile view. Other views are also observed, but they are less common. The *car* category includes cars of varying sizes as well as SUV's and trucks. With a few exceptions, buses have a similar appearance, since most of them are public transportation buses. The dataset consists of videos captured at different times of the day, resulting in a variety of illumination conditions as shown in Figure 1, including street-lights at night. For videos captured during the transition of day to night, the appearances of the vehicles also change (most prominently, vehicles in the dark have their headlights turned on).

5.1. Kernel Matrices

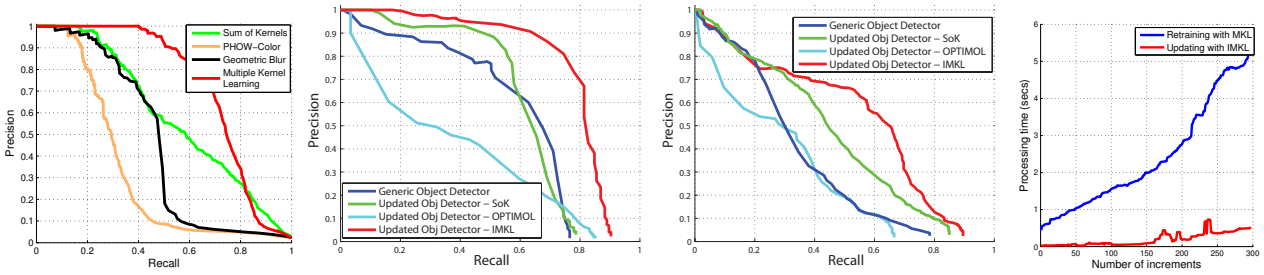
We use 5 kinds of features in our system, giving rise to a total of 17 kernel matrices. The first feature used is the Pyramidal Histogram of Oriented Gradients (PHOG-180) [6] to represent local shape. This consists of HOG features calculated over increasingly finer spatial grids. The orientations are calculated over the interval $[0, 180]$.We set the number of levels of the pyramid to 4. HOG features calculated for grids within the same level of the pyramid are concatenated to form a long feature vector, but feature vectors calculated at different levels are treated independently. Our IMKL algorithm automatically weights each level of the pyramid based on the training dataset. Histogram intersection is used as the similarity metric for all features in this paper. The first feature gives rise to 4 kernels, one for each level of the pyramid. The second feature is the PHOG-360. It only differs from PHOG-180 in that orientations are calculated over the interval $[0, 360]$. This also gives rise to 4 kernels.

The third feature, PHOW-Gray [6], encodes appearance. SIFT features are densely sampled at 10 pixel intervals in each direction and quantized to a 300 visual words vocabulary. Histograms of visual words are calculated over an increasing number of grids at each pyramidal level. We use 3 levels. The fourth feature is PHOW-Color. The only difference from PHOW-Gray is that it is calculated on the 3 channels of the HSV image. These give rise to 6 kernels.

The fifth feature is Geometric Blur (GB) [3], which captures shape information of the objects and also accounts for the geometric distortion between images. The un-quantized GB feature was used with an expensive correspondence based distance metric in [23]. However, in order to speed-up computations, we quantized the GB feature to a set of 300 visual words. We then calculated histograms of GB words in the same pyramidal framework to enforce some measure of spatial constraints. We used a 3 level pyramid. Thus we obtained a total of 17 kernel matrices.

5.2. Analysis

Evaluation of MKL. We first evaluate the power of using multiple kernels and using MKL to determine kernel weights for the given classification task. For this purpose, we created a validation dataset consisting of images of buses, cars and background extracted from the ground truth as well as the initial training set (obtained from Google). We then individually evaluated each kernel as well as the combination of kernels using a Sum of Kernels (SoK) approach (such as in [12]) and an MKL approach for both object classes over the validation set. The SoK approach assigns equal weights to all kernels. SoK has been known to provide good results when kernels are carefully chosen for the given data, but its performance degrades in the presence



(a) Evaluation of MKL (b) Precision-Recall for buses (c) Precision-Recall for cars (d) Efficiency of IMKL

Figure 7. (a) shows the evaluation of the individual kernels, combination using SoK and combination using MKL. MKL outperforms all other schemes. The best performing individual kernel is GB. (b) and (c) show the Precision-Recall curves for the *bus* and *car* classes respectively. Using our incremental object detector consistently increases performance in both cases. (d) compares the processing time of our incremental approach to retraining the MKL system at every step using all available images.

of noisy kernels. In our experiments, the MKL approach performs better than all other methods where as the SoK approach comes in second, outperforming both GB (the best performing individual feature) and the popular SIFT feature. Figure 7(a) shows the results for the *Buses* class.

Local dataset snapshots. We now demonstrate results of our IMKL approach on the video dataset. Starting from a generic training dataset, our IMKL algorithm simultaneously updates the training dataset as well as the kernel combination weights. Figure 6 shows snapshots of the training database at different time instants for one video.

Kernel weights over time. Figure 8 demonstrates the change of kernel combination weights over time. For this experiment, we chose a video where the scene is bright in the beginning but gets very dark by the end. We do not display kernel weights 1 to 8, since they do not show considerable change over time. Time 1 refers to the initial training dataset obtained from Google. Between times 1 and 2, we do not update the foreground classes to study the effect of updating only the background training set. This also causes a non-trivial change of weights (Time 2). After time 2, we update all object classes. Between times 2 and 3, the scene is bright. In this period, the detector tunes itself towards objects of specific poses and background patches. Beyond time 3, the scene gets darker. Here, PHOW-Color weights show a considerable drop (kernels 12-14), since color information in the video deteriorates, while PHOW-Gray kernels get higher weights. GB at fine spatial resolution (kernel 17) gets high weights with decreasing illumination, indicating added importance to positional information (such as importance given to the position of vehicle headlights).

Performance evaluation. Figures 7(b) and 7(c) show the performance of our system for the *bus* and *car* classes respectively, averaged over all videos in the dataset. We compare our IMKL object detector with 3 other detectors. Our baseline detector (which we call the *Generic* detector), represents an object detector built offline using only the generic training dataset and is not updated over time. It uses all 17 kernels and MKL to obtain the kernel weights. Our second comparison is to an object detector built on the generic

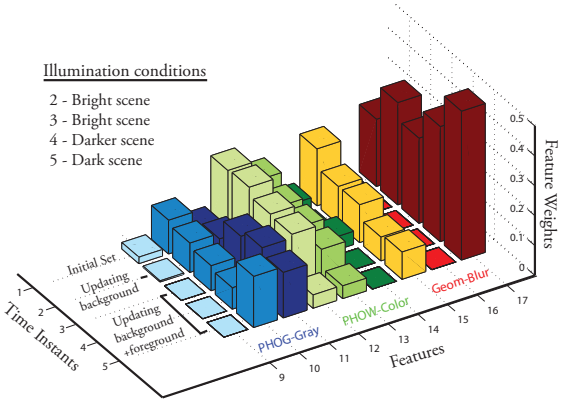


Figure 8. Kernel combination weights sampled at multiple time instants. Results shown for *Buses* class. *(see text for details)*

Google dataset and updated over time, but using SoK (equal kernel weights). Since these kernel weights are fixed over time, an incremental SVM approach suffices as the classifier. Our third comparison is to OPTIMOL [13], an incremental model learning approach, recently proposed for automatic object dataset collection.[2] The OPTIMOL algorithm is run independently of the IMKL system with a single change. In [13], Li et. al use SIFT as their feature descriptor. But given the superior performance of GB in our validation set (Figure 7(a)), we use histograms of GB based visual words as our feature descriptor for OPTIMOL.

Our IMKL approach outperforms the other 3 methods, especially at high recalls. Figure 10 provides some more insight into the results. This plot shows the performance of the various methods over time for one of the videos in our dataset for which illumination changes. The images at the bottom show a sample frame within the specified time interval. OPTIMOL starts off slowly but as it gets updated, it catches up with the rest of the object detectors. As the scene gets darker, however, its performance deteriorates. OPTIMOL uses GB, and even our IMKL approach begins to reduce the importance given to this kernel when the scene be-

[2] We obtained code for OPTIMOL from the authors.

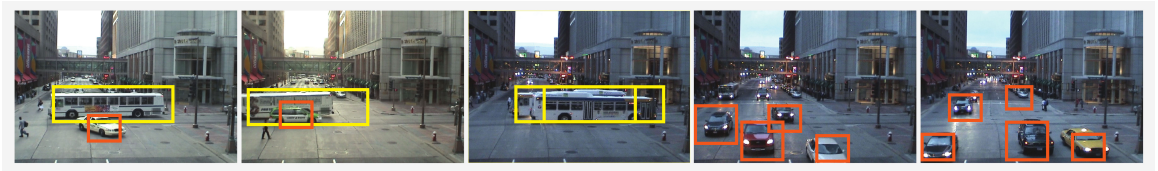


Figure 9. Sample results from a video sequence showing the ability of our system to adapt to gradual illumination changes.

comes dark. We also noticed a low overall performance of OPTIMOL (on a subset of the data) while using other kernels such as PHOW-Gray and PHOW-COLOR. This is because no single kernel has been able to provide consistently good results in all scene conditions. Using multiple kernels with fixed weights (SoK) was also sub-optimal. Our IMKL approach provided the best results because it was able to dynamically change kernel weights based on the current object and scene characteristics. IMKL's performance decreases at times 4 and 6 since the scene changes, but recovers at instants 5 and 7, once it updates itself sufficiently.

Figure 9 shows sample results. Overall, we detect buses more reliably than cars. We are unable to consistently detect cars smaller than 60x60 pixels, which is the case for cars approaching from a distance, giving rise to a number of false negatives. Finally, Figure 7(d) illustrates the computational efficiency of the IMKL algorithm as compared to retraining the entire system using SimpleMKL.

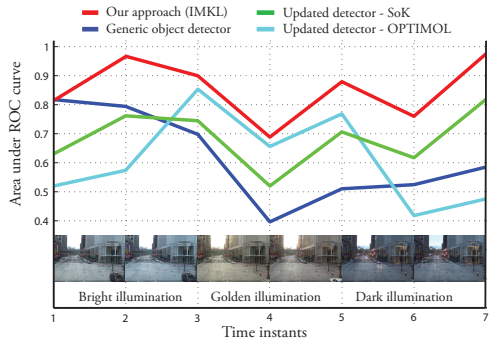


Figure 10. Performance comparison of object detectors over time for a single video for *Buses* class. *(see text for details)*

6. Conclusion

We have proposed an Incremental Multiple Kernel Learning (IMKL) approach to object recognition and demonstrated the performance gains on a vehicle classification task.

Acknowledgements. This research was partially supported by the ONR surveillance grant N000140910044. The authors also thank Vlad Morariu for useful discussions.

References

[1] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on PAMI*, 2004.

[2] F. Bach, G. Lanckriet, and M. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. *ICML*, 2004.

[3] A. Berg and J. Malik. Geometric blur for template matching. *CVPR*, 2001.

[4] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. *COLT: Proceedings of the Workshop on Computational Learning Theory*, 1998.

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

[6] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. *CIVR*, 2007.

[7] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. *NIPS*, 2000.

[8] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *CVPR*, 05.

[9] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. *CVPR03*.

[10] O. Javed, S. Ali, and M. Shah. Online detection and classification of moving objects using progressively improving detectors. *CVPR*, 2005.

[11] G. Lanckriet, N. Cristianini, L. El Ghaoui, P. Bartlett, and M. Jordan. Learning the kernel matrix with semi-definite programming. *JMLR*, 2004.

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

[13] L. Li, G. Wang, and L. Fei-Fei. Optimol: automatic object picture collection via incremental model learning. *CVPR*, 07.

[14] M. Pontil and A. Verri. Support vector machines for 3d object recognition. *IEEE Transactions on PAMI*, 1998.

[15] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet. More efficency in multiple kernel learning. *ICML*, 07.

[16] J. Sivic, B. Russell, A. Efros, A. Zisserman, and W. Freeman. Discovering objects and location in images. *ICCV*, 2005.

[17] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *JMLR*, 2006.

[18] N. Syed, H. Liu, and K. Sung. Incremental learning with support vector machines. *IJCAI*, 1999.

[19] M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. *ICCV*, 2007.

[20] B. Wu and R. Nevatia. Improving part based object detection by unsupervised, online boosting. *CVPR*, 2007.

[21] B. Wu and R. Nevatia. Simultaneous object detection and segmentation by boosting local shape feature based classifier. *CVPR*, 2007.

[22] B. Wu and R. Nevatia. Optimizing discrimination-efficiency tradeoff in integrating heterogeneous local features for object detection. *CVPR*, 2008.

[23] H. Zhang, A. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. *CVPR*, 2006.

[24] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *IJCV*, 2007.