Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2005-017
AIM-2005-007

March 17, 2005

Pyramid Match Kernels: Discriminative Classification with Sets of Image Features

Kristen Grauman, Trevor Darrell

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

Abstract

Discriminative learning is challenging when examples are sets of local image features, and the sets vary in cardinality and lack any sort of meaningful ordering. Kernel-based classification methods can learn complex decision boundaries, but a kernel similarity measure for unordered set inputs must somehow solve for correspondences – generally a computationally expensive task that becomes impractical for large set sizes. We present a new fast kernel function which maps unordered feature sets to multi-resolution histograms and computes a weighted histogram intersection in this space. This "pyramid match" computation is linear in the number of features, and it implicitly finds correspondences based on the finest resolution histogram cell where a matched pair first appears. Since the kernel does not penalize the presence of extra features, it is robust to clutter. We show the kernel function is positive-definite, making it valid for use in learning algorithms whose optimal solutions are guaranteed only for Mercer kernels. We demonstrate our algorithm on object recognition tasks and show it to be dramatically faster than current approaches.

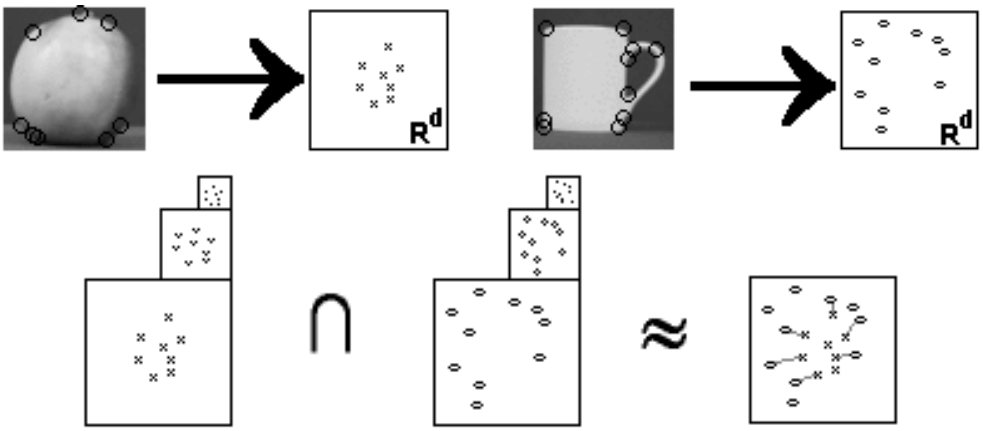


Figure 1: The pyramid match kernel intersects pyramids formed over histograms of local features, approximating the optimal correspondences between the sets' features.

1. Introduction

A variety of representations used in computer vision consist of unordered sets of features, where each set varies in cardinality, and the correspondence between the features across each set is unknown. For instance, an image may be described by a set of detected local affine-invariant regions, a shape may be described by a set of local descriptors defined at each edge point, or a person's face may be represented by a set of patches with different facial parts. In such cases, one set of feature vectors denotes a single instance of a particular class of interest (an object, scene, shape, face, etc.), and it is expected that the number of features will vary across examples due to viewpoint changes, occlusions, or inconsistent detections by the interest operator.

To perform learning tasks like categorization or recognition with such representations is challenging. While generative methods have had some success, kernel-based discriminative methods are known to represent complex decision boundaries very efficiently and generalize well to unseen data [26, 8]. For example, the Support Vector Machine (SVM) is a widely used approach to discriminative classification that finds the optimal separating hyperplane between two classes. Kernel functions, which measure similarity between inputs, introduce non-linearities to the decision functions; the kernel non-linearly maps two examples from the input space to the inner product in some feature space. However, conventional kernel-based algorithms are designed to operate on fixed-length vector inputs, where each vector entry corresponds to a particular global attribute for that instance; the commonly used general-purpose kernels defined on $\Re^n$ inputs (e.g., Gaussian RBF, polynomial) are not applicable in the space of vector sets.

In this work we propose a *pyramid match kernel* – a new kernel function over unordered feature sets that allows such sets to be used effectively and efficiently in kernel-based learning methods. Each feature set is mapped to a multi-resolution histogram that preserves the individual features' distinctness at the finest level. The histogram pyramids are then compared using a weighted histogram intersection computation, which we show defines an implicit correspondence based on the finest resolution histogram cell where a matched pair first appears (see Figure 1).

The similarity measured by the pyramid match approximates the similarity measured by the optimal correspondences between feature sets of unequal cardinality (i.e., the *partial matching* that optimally maps points in the lower cardinality set to some subset of the points in the larger set, such that the summed similarities between matched points is maximal). Our kernel is extremely efficient and can be computed in time that is linear in the sets' cardinality. We show that our kernel function is positive-definite, meaning that it is appropriate to use with learning methods that guarantee convergence to a unique optimum only for positive-definite kernels (e.g., SVMs).

Because it does not penalize the presence of superfluous data points in either input set, the proposed kernel is robust to clutter. As we will show, this translates into the ability to handle unsegmented training examples and test examples with varying backgrounds or occlusions. The kernel also respects the co-occurrence relations inherent in the input sets: rather than matching features in a set individually, ignoring potential dependencies conveyed by features within one set, our similarity measure captures the features' joint statistics.

Other approaches to this problem have recently been proposed [27, 18, 4, 16, 28, 20], but unfortunately each of these techniques suffers from some number of the following drawbacks: computational complexities that make large feature set

Method	Complexity	Captures co-occurrences	Positive-definite	Model-free	Handles unequal cardinalities
Match [27]	$O(dm^2)$			x	x
Exponent match [18]	$O(dm^2)$		x	x	x
Greedy match [4]	$O(dm^2)$	x		x	x
Principal angles [28]	$O(dm^3)$	x	x		
Bhattacharyya [16]	$O(dm^3)$	x	x		x
KL-divergence [20]	$O(dm^2)$	x			x
Pyramid match	$O(dm\ log(D))$	x	x	x	x

Table 1: Comparing kernel approaches to matching unordered sets. Columns show each method's computational cost and whether its kernel captures co-occurrences, is positive-definite, does not assume a parametric model, and can handle sets of unequal cardinality. d is vector dim., m is maximum set cardinality, and D is diameter of vector space. "Pyramid match" refers to the proposed kernel.

sizes infeasible; limitations to parametric distributions which may not adequately describe the data; kernels that are not positive-definite (do not guarantee unique solutions for an SVM); limitations to sets of equal size; and failure to account for dependencies within feature sets.

Our method successfully addresses all of these issues, resulting in a kernel appropriate for comparing unordered, variable length feature sets within any existing kernel-based learning paradigm. We demonstrate our algorithm with object recognition tasks and show that its accuracy is comparable to current approaches, while requiring at least an order of magnitude less computation time.

2. Related Work

In this section, we review related work on discriminative classification in the domain of sets of features, the use of kernels and SVMs for recognition, and multi-resolution image representations.

Object recognition is a challenging problem that requires strong generalization ability from a classifier in order to cope with the broad variety in illumination, viewpoint, occlusions, clutter, intra-class appearances, and deformations that images of the same object or object class will exhibit. While researchers have shown promising results applying SVMs to object recognition, they have generally used global image features – ordered features of equal length measured from the image as a whole, such as color or grayscale histograms or vectors of raw pixel data [6, 22, 21]. Such global representations are known to be sensitive to real-world imaging conditions, such as occlusions, pose changes, or image noise.

Recent work has shown that local features invariant to common image transformations (e.g., SIFT [17]) are a powerful representation for recognition, because the features can be reliably detected and matched across instances of the same object or scene under different viewpoints, poses, or lighting conditions. Most researchers, however, have done recognition with local feature representations using nearest-neighbor (e.g., [2, 12, 24]) or voting-based classifiers followed by an alignment step (e.g., [17, 19]); both may be impractical for large training sets, since their classification times increase with the number of training examples. An SVM, on the other hand, identifies a sparse subset of the training examples (the "support vectors") to delineate a decision boundary.

Kernel-based learning algorithms, which include SVMs, kernel PCA, or Gaussian Processes, have become well-established tools that are useful in a variety of contexts, including discriminative classification, regression, density estimation, and clustering (e.g., see [8]). More recently, attention has been focused on developing specialized kernels that can more fully leverage these tools for situations where the data cannot be naturally represented by a Euclidean vector space, such as graphs, strings, or trees. See the survey [11] for an overview of different types of kernels.

Several researchers have designed similarity measures that operate on sets of unordered features. See Table 1 for a concise comparison of the approaches. The authors of [27] propose a kernel that averages over the similarities of the best matching feature found for each feature member within the other set. The use of the "max" operator in this kernel makes it non-Mercer (not positive-definite – see Section 3), and thus it lacks theoretical convergence guarantees when used in an SVM. A similar kernel is given in [18], which also considers all possible matchings between features but measures overall similarity with a different bias, by raising the similarity between each pair of features to a given power. Both [27] and [18] have a computational complexity that is squared in the number of features. Furthermore, both match each feature in a set independently, ignoring potentially useful co-occurrence information. In contrast, our kernel captures the joint statistics of

co-occurring features by matching them concurrently, as a set.

The method given in [4] is based on finding a sub-optimal matching between two sets using a greedy heuristic; although this results in a non-Mercer kernel, the authors provide a means of tuning the kernel hyperparameter so as to limit the probability that a given kernel matrix is not positive-definite. The authors of [28] measure similarity in terms of the principal angle between the two linear subspaces spanned by two sets' vector elements. This kernel is only positive-definite for sets of equal cardinality [28], and its complexity is cubic in the number of features.

In [16], a Gaussian is fit to each set of vectors, and then the kernel value between two sets is the Bhattacharyya affinity between their Gaussian distributions. As noted by the authors, the method is constrained to using a Gaussian model in order to have a closed form solution. In practice, the method in [16] is also limited to sets with small cardinality, because its complexity is cubic in the number of features. Similarly, the authors of [20] fit a Gaussian to a feature set, and then compare sets using KL-divergence as a distance measure; it is not clear if this designates a positive-definite function. Unlike the kernels of [16] and [20], which are based on parametric models that assume inputs will fit a certain form, our method represents an input set by forming a multi-resolution histogram over the data where the finest level captures the distinct data points themselves.

A performance evaluation given in [9] compares the methods of [16, 28, 27] in the context of an object categorization task using images from a publicly available database. The experiments show the methods of [27] and [16] performing comparably, but the authors were unable to achieve satisfactory results using other methods they tested. However, the study also concluded that the cubic complexity of the method given in [16] made it impractical to use the desired number of features. Below we evaluate our method under the conditions of [9], and show recognition performance comparable to the best reported in [9], for a substantially lower complexity.

An alternative approach when dealing with unordered set data is to designate prototypical examples from each class, and then represent examples in terms of their distances to each prototype; standard algorithms that handle vectors in a Euclidean space are then applicable. The authors of [29] build such a classifier for handwritten digits, and use the shape context distance of [2] as the measure of similarity. The issues faced by such a prototype-based method are determining which examples should serve as prototypes, choosing how many there should be, and updating the prototypes properly when new types of data are encountered. A learning architecture based on a sparse network of linear functions is given in [1] and applied to object detection using binary feature vectors that encode the presence or absence of a pre-established set of object parts. The method only makes binary decisions (detection, not multi-class recognition), and it is limited to objects with a set of parts that are arranged in a fixed 2-D spatial configuration.

Our feature representation is based on a multi-resolution histogram, or "pyramid", which is computed by binning data points into discrete regions of increasingly larger size. Single-level histograms have been used in various visual recognition systems, one of the first being that of [25], where the intersection of global color histograms was used to compare images. Pyramids have been shown to be a useful representation in a wide variety of image processing tasks – see [13] for a summary.

In [14], multi-resolution histograms are compared with L_1 distance to approximate a least-cost matching of equal-mass global color histograms for nearest neighbor image retrievals. This work inspired our use of a similar representation for point sets. However, unlike [14], our method builds a discriminative classifier, and it compares histograms with a weighted intersection rather than L_1. Our method allows inputs to have unequal cardinalities and thus enables partial matchings, which is important in practice for handling clutter and unsegmented images, as we demonstrate in Section 4.

We believe ours is the first work to advocate for the use of a histogram pyramid as an explicit discriminative feature formed over sets, and the first to show its connection to optimal partial matching when used with a hierarchical weighted histogram intersection similarity measure.

3. Approach

The main contribution of this work is a new kernel function based on implicit correspondences that enables discriminative classification for unordered, variable-length sets of vectors. The kernel is provably positive-definite. The main advantages of our algorithm are its efficiency, its use of implicit correspondences that respect the joint statistics of co-occurring features, and its resistance to clutter or "superfluous" data points in either set.

The basic idea of our method is to map sets of features to multi-resolution histograms, and then compare the histograms with a weighted histogram intersection measure in order to approximate the similarity of the best partial matching between the feature sets.

3.1. The Pyramid Match Kernel

Kernel-based learning algorithms [8, 26] are founded on the idea of embedding data into a Euclidean space, and then seeking linear relations among the embedded data. For example, an SVM finds the optimal separating hyperplane between two classes in an embedded space (also referred to as the feature space). A kernel function $K : X \times X \rightarrow \Re$ serves to map pairs of data points in an input space X to their inner product in the embedding space F, thereby evaluating the similarities between all points and determining their relative positions. Note that linear relations are sought in the embedded space, but a decision boundary may still be non-linear in the input space, depending on the choice of a feature mapping function $\Phi : X \rightarrow F$.

We call the proposed kernel a "pyramid match kernel", because input sets are first represented as multi-resolution histograms. We consider an input space X of sets of d-dimensional feature vectors that are bounded by a sphere of diameter D and whose minimum inter-vector distance is 1:[1]

$$X = \left\{ \mathbf{x} | \mathbf{x} = \left\{ [f_1^1, \ldots, f_d^1], \ldots, [f_1^{m_\mathbf{x}}, \ldots, f_d^{m_\mathbf{x}}] \right\} \right\}, \tag{1}$$

where $m_\mathbf{x}$ varies across instances in X.

The feature extraction function Ψ is defined as:

$$\Psi(\mathbf{x}) = [H_{-1}(\mathbf{x}), H_0(\mathbf{x}), \ldots, H_L(\mathbf{x})], \tag{2}$$

where $L = \lceil \log(D) \rceil$, $\mathbf{x} \in X$, $H_i(\mathbf{x})$ is a histogram vector formed over data $\mathbf{x}$ using d-dimensional bins of side length 2^i, and $H_i(\mathbf{x})$ has a dimension $r_i = \left(\frac{D}{2^i}\right)^d$. In other words, $\Psi(\mathbf{x})$ is a vector of concatenated histograms, where each subsequent component histogram has bins that double in size (in all d dimensions) compared to the previous one. The bins in the finest-level histogram H_{-1} are small enough that each d-dimensional data point from sets in X falls into its own bin, and then the bin size increases until all data points from sets in X fall into a single bin at level L.

The pyramid match kernel K_Δ is a similarity function that is applied to inputs in this multi-resolution histogram space. The similarity between two input sets is defined as the weighted sum of the number of feature matchings found at each level of the pyramid formed by Ψ:

$$K_\Delta \left(\Psi(\mathbf{y}), \Psi(\mathbf{z}) \right) = \sum_{i=0}^{L} w_i N_i, \tag{3}$$

where N_i signifies the number of newly matched pairs at level i. A new match is defined as a pair of features that were not in correspondence at any finer resolution level.

The kernel implicitly finds correspondences between point sets, if we consider two points matched once they fall into the same histogram bin (starting at the finest resolution level where each point is guaranteed to be in its own bin). The matching is equivalent to a hierarchical process: vectors not found to correspond at a high resolution have the opportunity to be matched at lower resolutions. For example, in Figure 2, there are three points matched at the finest scale, one new match at the medium scale, and two at the coarsest scale. K_Δ's output value reflects the overall similarity of the matching: each newly matched pair at level i contributes a value w_i that is proportional to how similar two points matching at that level must be, as determined by the bin size. Note that the sum in Eqn. 3 starts with index $i = 0$, because the definition of Ψ insures that no points match at level $i = -1$.

To calculate N_i, the kernel makes use of a histogram intersection function $\mathcal{I}$, which measures the "overlap" between two histograms' bins:

$$\mathcal{I}\left(\mathbf{A}, \mathbf{B}\right) = \sum_{j=1}^{r} \min\left(\mathbf{A}^{(j)}, \mathbf{B}^{(j)}\right), \tag{4}$$

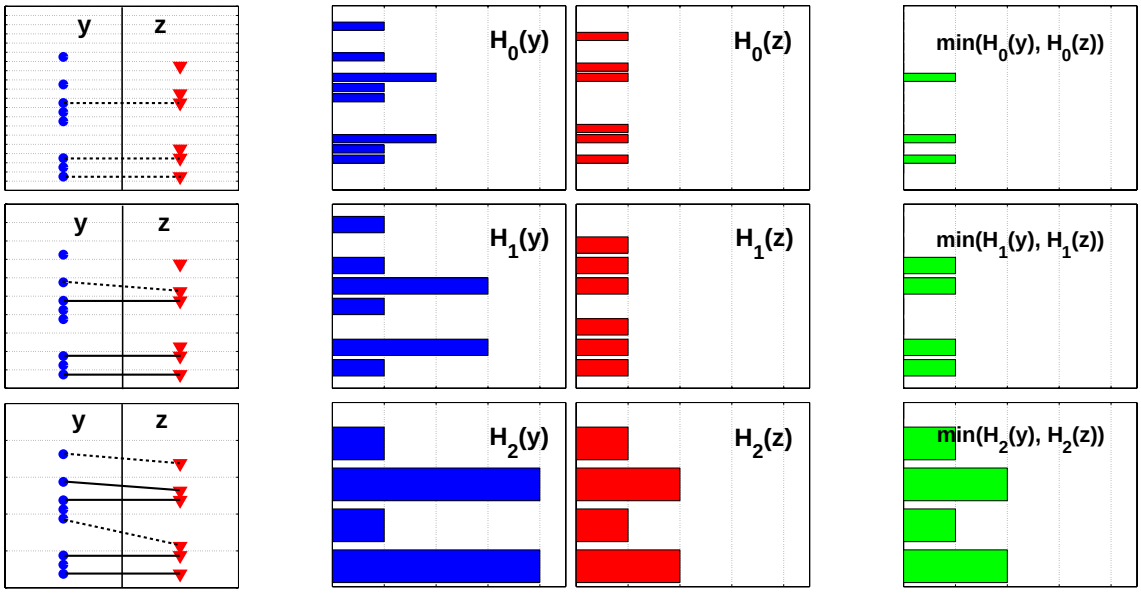
where $\mathbf{A}$ and $\mathbf{B}$ are histograms with r bins, and $\mathbf{A}^{(j)}$ denotes the count of the j^{th} bin of $\mathbf{A}$.

The histogram intersection counts the number of points in two sets which match at a given quantization level, i.e., are similar enough to fall into the same bin. To calculate the number of newly matched pairs N_i, it is sufficient to compute the difference between successive histogram levels' intersections:

$$N_i = \mathcal{I}\left(H_i(\mathbf{y}), H_i(\mathbf{z})\right) - \mathcal{I}\left(H_{i-1}(\mathbf{y}), H_{i-1}(\mathbf{z})\right), \tag{5}$$

where H_i refers to the i^{th} component histogram generated by Ψ in Eqn. 2. Note that the kernel is not searching explicitly for similar points – it never computes distances between the vectors in each set. Instead, it simply uses the change in intersection values at each histogram level to count the matches as they occur.

[1] A minimum inter-point distance equal to 1 may be enforced by scaling the data appropriately.



(a) Point sets (b) Histogram pyramids (c) Intersections

Figure 2: A pyramid match determines a partial correspondence by matching points once they fall into the same histogram bin. In this example, two 1-D feature sets are used to form two histogram pyramids. Each row of the figure corresponds to a pyramid level. H_{-1} is not pictured here because no matches are formed at the finest level. In (a), points are distributed along the vertical axis; the first set $\mathbf{y}$ is on the left side, the second set, $\mathbf{z}$ on the right. These same points are repeated at each level. Light dotted lines are bin boundaries, bold dashed lines indicate a pair matched at this level, and bold solid lines indicate a match already formed at a finer resolution level. In (b) multi-resolution histograms are shown, with bin counts along the horizontal axis. In (c) the intersection pyramid between the histograms in (b) are shown. K_Δ uses this to measure how many new matches occurred at each level. In this example, $\mathcal{I} = 3, 4, 6$ across levels, and therefore $N_i = 3, 1, 2$, corresponding to the number of new matches found at each level. The sum over N_i, weighted by $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}$, gives the pyramid match similarity.

The number of new matches found at each level in the pyramid is weighted according to the size of that histogram's bins: matches made within larger bins are weighted less than those found in smaller bins. Intuitively, this means that similarity between vectors (features in $\mathbf{y}$ and $\mathbf{z}$) at a finer resolution – where features are most discriminative – is rewarded more heavily than similarity between vectors at a coarser level. The weights are set to $w_i = \frac{1}{2^i}$ to reflect the fact that bin sizes increase by a factor of two at each level, making it twice as "easy" for two points to match at each subsequent coarser level; that is, matches made at a given level are on average half as similar as those made at the previous finer level, with similarity measured as the inverse of the distance between the points.

From Eqns. 3, 4, and 5, we define the (un-normalized) pyramid match kernel function:

$$\tilde{K}_\Delta\left(\Psi(\mathbf{y}), \Psi(\mathbf{z})\right) = \sum_{i=0}^{L} \frac{1}{2^i}\Big(\mathcal{I}\left(H_i(\mathbf{y}), H_i(\mathbf{z})\right) - \mathcal{I}(H_{i-1}(\mathbf{y}), H_{i-1}(\mathbf{z}))\Big), \tag{6}$$

where $\mathbf{y}, \mathbf{z} \in X$, and $H_i(\mathbf{x})$ is the i^{th} histogram in $\Psi(\mathbf{x})$. We normalize this value by the product of each input's self-similarity to avoid favoring larger input sets, arriving at the final kernel value $K_\Delta\left(\mathbf{P}, \mathbf{Q}\right) = \frac{1}{\sqrt{C}}\tilde{K}_\Delta\left(\mathbf{P}, \mathbf{Q}\right)$, where $C = \tilde{K}_\Delta\left(\mathbf{P}, \mathbf{P}\right)\tilde{K}_\Delta\left(\mathbf{Q}, \mathbf{Q}\right)$.

Our kernel allows sets of unequal cardinalities, and therefore it enables *partial matchings*, where the points of the smaller set are mapped to some subset of the points in the larger set. Dissimilarity is only judged on the most similar part of the empirical distributions, and superfluous data points are ignored; the result is a robust similarity measure that accommodates inputs expected to contain extraneous vector entries. This is of course a common situation when recognizing objects in images, due for instance to background variations, clutter, or changes in object pose that cause different subsets of features

to be visible. Thus, the proposed kernel is equipped to handle unsegmented training examples, as we will demonstrate in Section 4.

By construction, the pyramid match offers an approximation of the optimal correspondence-based matching between two feature sets, in which the overall similarity between corresponding points is maximized. When input sets have equal cardinalities, histogram intersection can be reduced to an L_1 distance: $\mathcal{I}(H(\mathbf{y}), H(\mathbf{z})) = m - \frac{1}{2}||H(\mathbf{y}) - H(\mathbf{z})||_{L_1}$ if $m = |\mathbf{y}| = |\mathbf{z}|$ [25]. Intersection over the pyramid with weights set to $w_i = \frac{1}{2^i}$ then strictly approximates the optimal bipartite matching [14]. With variable cardinalities no similar proof is available, but we show empirically below that the intersection of multi-resolution histograms approximates the best partial matching both in simulation and in practice.

Since the pyramid match defines correspondences across entire sets simultaneously, it inherently accounts for dependencies between various features occurring in one set. In contrast, previous approaches have used each feature in a set to independently index into the second set; this ignores possibly useful information that is inherent in the co-occurrence of a set of distinctive features, and it fails to distinguish between instances where an object has varying numbers of similar features since multiple features may be matched to a single feature in the other set [27, 18].

3.2. Satisfying Mercer's Condition

Only positive-semi-definite kernel functions guarantee an optimal solution to kernel-based algorithms based on convex optimization, including SVMs. According to Mercer's theorem, a kernel K is positive-semi-definite if and only if

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle, \ \forall \mathbf{x}_i, \mathbf{x}_j \in X, \tag{7}$$

where $\langle \cdot \rangle$ denotes a scalar dot product. This insures that the kernel does correspond to the inner product in some feature space, where kernel methods can search for linear relations (see e.g. [8]). In the following, we will show that there does exist a function Φ that maps the input data to a feature space where dot products are equivalent to the kernel value computed on the original inputs, meaning that the kernel is semi-positive-definite (a "Mercer's kernel").

Histogram intersection (HI) on single resolution histograms over multi-dimensional data was shown to be a positive-definite similarity function in [21]. The proof shows that there is an explicit feature mapping after which the HI is an inner product. Specifically, the mapping $\mathcal{V}$ encodes an r-bin histogram H as a p-dimensional binary vector, $p = m \times r$, where m is the number of input points:

$$\mathcal{V}(H) = \left(\underbrace{\overbrace{1, \ldots, 1}^{H^{(1)}}, \overbrace{0, \ldots, 0}^{m - H^{(1)}}}_{1^{st} \text{ bin}}, \ldots, \underbrace{\overbrace{1, \ldots, 1}^{H^{(r)}}, \overbrace{0, \ldots, 0}^{m - H^{(r)}}}_{\text{last bin}} \right). \tag{8}$$

The inner product between the binary strings output from $\mathcal{V}$ is equivalent to the original histograms' intersection value [21]. If m varies across examples, the above holds by setting $p = M \times r$, where M is the maximum size of any input.

An extension of this idea shows that the pyramid match kernel given in Eqn. 6 also satisfies Mercer's condition. The mapping Φ illustrating this is defined as:

$$\Phi(\Psi(\mathbf{x})) = [\mathcal{V}'(H_0), \mathcal{V}'(H_1), \ldots, \mathcal{V}'(H_L)], \quad \text{where}$$

$$\mathcal{V}'(H_i) = \left(\underbrace{\overbrace{\sqrt{w_i}, \ldots, \sqrt{w_i}}^{H_i^{(1)}}, \overbrace{0, \ldots, 0}^{M - H_i^{(1)}}}_{1^{st} \text{ bin of } H_i}, \ldots, \underbrace{\overbrace{\sqrt{w_i}, \ldots, \sqrt{w_i}}^{H_i^{(r_i)}}, \overbrace{0, \ldots, 0,}^{M - H_i^{(r_i)}}}_{\text{last bin of } H_i} \right.$$
$$\left. \underbrace{\overbrace{-\sqrt{w_i}, \ldots, -\sqrt{w_i}}^{H_{i-1}^{(1)}} \overbrace{0, \ldots, 0}^{M - H_{i-1}^{(1)}}}_{1^{st} \text{ bin of } H_{i-1}}, \ldots, \underbrace{\overbrace{-\sqrt{w_i}, \ldots, -\sqrt{w_i},}^{H_{i-1}^{(r_{i-1})}} \overbrace{0, \ldots, 0}^{M - H_{i-1}^{(r_{i-1})}}}_{\text{last bin of } H_{i-1}} \right), \tag{9}$$

where $w_i = \frac{1}{2^i}$. Note that Φ is never computed explicitly; it only serves to show a function exists that will map inputs to a space where inner products are equivalent to the values our kernel produces. This indicates that K_Δ is valid for use in existing learning methods that require Mercer kernels.

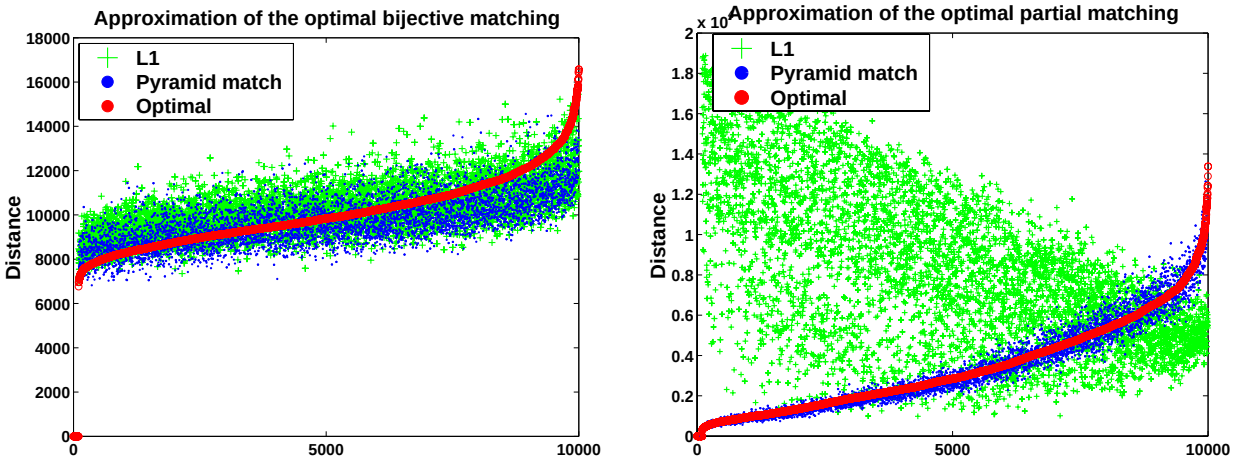


Figure 3: The pyramid match approximates the optimal correspondences, even for sets of unequal cardinalities (right). See text for details. (This figure is best viewed in color.)

3.3. Efficiency

The time required to compute Ψ for an input set with m d-dimensional features is $O(dm\log(D))$, since it is possible to record bin counts directly from the input vectors by going through them and computing their bin coordinates at each quantization level, then counting the occurrences of each bin. The multi-resolution histogram that results is high-dimensional, but very sparse, with on the order of $O(m\log(D))$ non-zero entries that need to be stored.

The complexity of K_Δ is $O(dm\log(D))$: sorting the histograms by their non-zero entries' indices requires only $O(dm\log(D))$ operations using a counting sort (a linear time sorting algorithm suitable for integers [7], which the bin indices are), and computing HI on the sparse vectors is also $O(dm\log(D))$. Note that HI calculations depend only on the number of non-zero entries – not the number of actual bins – if the histograms are sorted by bin indices. In contrast, current approaches have polynomial dependence on m, which limits the practicality of large input sizes. See Table 1 for complexity comparisons.

4. Results

In this section we show that in simulation the pyramid match kernel approximates the best partial matching of feature sets, and then we report on object recognition experiments with baseline comparisons to other methods.

4.1. Approximating the Optimal Partial Matching

As described in Section 3, the pyramid match approximates the optimal correspondence-based matching between two feature sets. While for the case of equal cardinalities it reduces to an L_1 norm in a space that is known to strictly approximate the optimal bijective matching (see [14]), here we show empirically how well the pyramid kernel approximates the optimal partial matching of unequal cardinality sets.

The idea of this experiment is to evaluate how close the correspondences implicitly assigned by the pyramid match are to the true optimal correspondences – the matching that results in the maximal summed similarity between corresponding points. To do this, we compared our kernel's outputs to the similarities (inverse distances) produced by the optimal partial matching obtained via a linear programming solution to the transportation problem [23]. [2]

We considered two data sets, each with 100 point sets containing 2-D points with values ranging from one to 1000. In one data set, each point set had equal cardinalities (100 points each), while in the other cardinalities varied randomly from 5 to 100. Figure 3 shows the results of 10,000 pairwise set-to-set comparisons computed according to the correspondences produced by the optimal matching, the pyramid match, and the L_1 embedding of [14], respectively, for each of these sets. Note that in these figures we plot distance (inverse similarity), and the values were sorted according to the optimal measure's magnitudes for visualization purposes.

[2] Note that this method has a computational complexity that is exponential in the number of features in the worst case, although it often exhibits polynomial-time behavior in practice. In contrast, the pyramid kernel's complexity is only linear in the number of features.

Figure 4: Example images from the ETH-80 objects database. Five images from each of the eight object classes (apple, cow, dog, pear, car, cup, horse, and tomato) are shown here.

This figure shows that our method does indeed find matchings that are consistently on par with the optimal solution. In the equal cardinality case (plot on left), both the pyramid match and the L_1 embedding produce good approximations; both are on average less than 9% away from the optimal measure. However, more importantly, the pyramid match can also approximate the partial matching for the unequal cardinality case (plot on right): its matchings continue to follow the optimal matching's trend since it does not penalize outliers, whereas the L_1 embedding fails because it is forced to match all points to something. Our method is again on average less than 9% away from the optimal matching's measure for the unequal cardinality case, while the L_1 matching has an average error of 400%.

4.2. Object Recognition

For our object recognition experiments we use SVM classifiers, which are trained by specifying the matrix of kernel values between all pairs of training examples. The kernel's similarity values determines examples' relative positions in an embedded space, and quadratic programming is used to find the optimal separating hyperplane in this space between the two classes. We use the implementation given by [5].

Local affine- or scale- invariant feature descriptors extracted from a sparse set of interest points in an image have been shown to be an effective, compact representation (e.g. [17, 19]). This is a good context in which to test our kernel function, since such local features have no inherent ordering, and it is expected that the number of features will vary across examples. In the following we experiment with two publicly available databases and demonstrate that our method achieves good object recognition performance at a much lower computational cost than other approaches. All run-times reported below for the pyramid match kernel include both the time needed to compute the histograms as well as the weighted intersections.

In [9], a performance evaluation is given to compare the methods of [16, 28, 27] on an object categorization task using images from the ETH-80 database.[3] The experiment uses eight object classes, with 10 unique objects and five widely separated views of each, for a total of 400 images (see Figure 4). A Harris detector is used to find interest points in each image, and various local descriptors (SIFT, JET, patches) are used to compose the feature sets. A one-versus-all SVM classifier is trained for each kernel type, and performance is measured via cross-validation, where all five views of an object are held out at once. Note that no instances of a test object are ever present in the training set, so this is a categorization task (as opposed to recognition of the same object).

In [9], the best classification performance using an average of 40 interest points is 74%, and it is achieved by both the polynomial-time methods of [16, 27] using SIFT [17] features. Using 120 interest points, the Bhattacharyya kernel [16] achieves 85% accuracy, but with cubic time complexity.

We evaluated our method on this same subset of the ETH-80 database, and it achieved a correct recognition rate of 82% using PCA-SIFT [15] features from all Harris-detected interest points (averages 153 points per image), with a time complexity that is linear in the number of features. Restricting ourselves to an average of 40 interest points yields a recognition rate of

[3] http://www.vision.ethz.ch/projects/categorization/

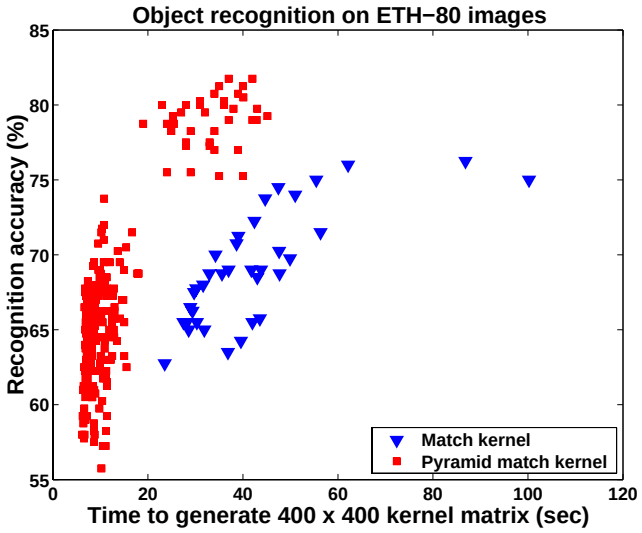


Figure 5: Allowing the same run-time, the pyramid match kernel produces better recognition rates than an approach that computes pairwise distances between features in order to match them. See text for details.

68%. Thus our method performs comparably to the others at their best for this data set, but is much more efficient than those tested above. The Bhattacharyya kernel's complexity is cubic in the number of features, and as observed in [9], this severely limits the practicality of using more than 100 features.

In fact, the ability of a kernel to handle large numbers of features can be critical to its success. An interest operator may be tuned to select only the most "salient" features, but in our experiments we found that the various approaches' recognition rates always benefitted from having larger numbers of features per image with which to judge similarity. Figure 5 depicts the run-time versus recognition accuracy of our method as compared to the kernel of [27] (called the "match" kernel), which has $O(dm^2)$ complexity. Each point in the figure represents one experiment; the saliency threshold of the Harris interest operator was adjusted to generate varying numbers of features, thus trading off accuracy versus run-time. Computing a kernel matrix for the same ETH data is significantly faster with our pyramid match kernel, and it produces much better recognition results for similar run-times.

We also tested our method with a challenging database of 101 objects recently developed at Caltech.[4] This database was obtained using Google Image Search, and the images contain significant clutter, occlusions, and intra-class appearance variation (see Figure 6). The authors of [3] report a multi-class recognition accuracy of 48% on this database using a nearest-neighbor classifier with a correspondence-based distance and manually segmented training examples. A generative model approach given in [10] obtained a recognition rate of 16%. Chance performance would be less than 1% on this database.

We ran our method on the 101 Objects database with a one-versus-all SVM classifier. We used the SIFT interest operator of [17] and the PCA-SIFT descriptor of [15] to form the input feature sets, and tested under the conditions used above, with 15 training examples and 50 test examples per class, all randomly selected. Note however that while the authors of [3] manually segmented the objects in the training examples, we trained our algorithm with raw *unsegmented* images, i.e., we included all detected interest point features in the input sets. This is a significant advantage of our approach: since it essentially seeks the best correspondence between some subset of either of the images' features, it handles unsegmented, cluttered data well. Despite not having hand-segmented training examples, our pyramid match kernel obtains a recognition rate of 43% on this database.

The approach in [3] requires $O(m^2 n log(n))$ time to solve for approximate correspondences via linear programming, for n test features and m model features. To be concrete, the authors quote a run-time of approximately 2 seconds to compute a single matching when $m = 50$, $n = 2550$, $d = 60$; our method is an order of magnitude faster, requiring only 0.1 second on average for the same parameters.

[4] http://www.vision.caltech.edu/feifeili/101ObjectCategories

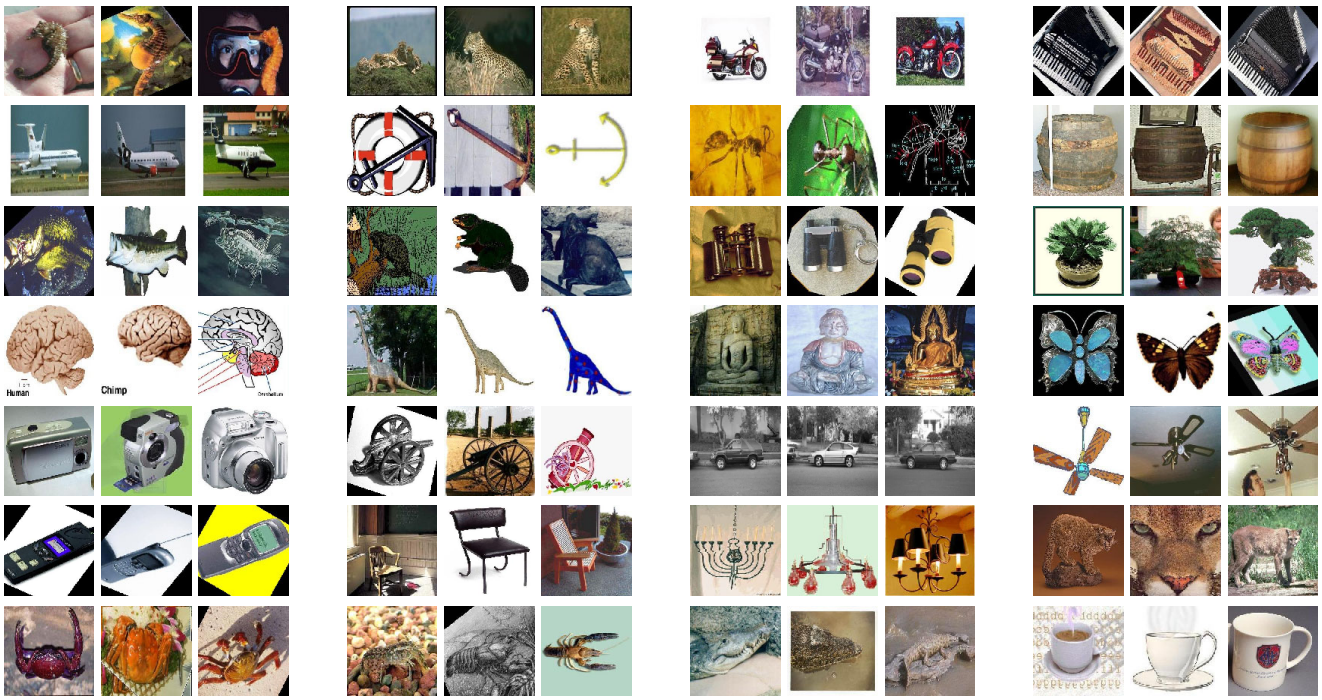


Figure 6: Example images from the Caltech 101 Objects database. Three images are shown for each of 28 of the 101 categories.

5. Conclusions

We have developed a new fast kernel function that is suitable for discriminative classification with unordered sets of local features. Our pyramid match kernel approximates the optimal partial matching by computing a weighted intersection over multi-resolution histograms, and requires time linear in the number of features. The kernel is robust to clutter since it does not penalize the presence of extra features, respects the co-occurrence statistics inherent in the input sets, and is provably positive-definite. We have applied our kernel to SVM-based object recognition tasks, and demonstrated good recognition performance with accuracy comparable to current methods, but with an order of magnitude speed improvement.

References

[1] S. Agarwal, A. Awan, and D. Roth. Learning to Detect Objects in Images via a Sparse, Part-Based Representation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 26, pages 1475–1490, November 2004.

[2] S. Belongie, J. Malik, and J. Puzicha. Shape Matching and Object Recognition Using Shape Contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(24):509–522, April 2002.

[3] A. Berg, T. Berg, and J. Malik. Shape Matching and Object Recognition using Low Distortion Correspondence. Technical report, U.C. Berkeley, Dec 2004.

[4] S. Boughhorbel, J-P. Tarel, and F. Fleuret. Non-Mercer Kernels for SVM Object Recognition. In *British Machine Vision Conference*, London, UK, Sept 2004.

[5] C. Chang and C. Lin. *LIBSVM: a library for support vector machines*, 2001.

[6] O. Chapelle, P. Haffner, and V. Vapnik. SVMs for Histogram-Based Image Classification. *Transactions on Neural Networks*, 10(5), Sept 1999.

[7] T. Cormen, C. Leiserson, and R. Rivest. *Intro. to Algorithms*. MIT Press, 1990.

[8] N. Cristianini and J. Taylor. *An Introduction to Support Vector Machines and other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

[9] J. Eichhorn and O. Chapelle. Object Categorization with SVM: Kernels for Local Features. Technical report, MPI for Biological Cybernetics, 2004.

[10] L. Fei-Fei, R. Fergus, and P. Perona. Learning Generative Visual Models from Few Training Examples: an Incremental Bayesian Approach Tested on 101 Object Cateories. In *Workshop on Generative Model Based Vision*, Washington, D.C., June 2004.

[11] T. Gartner. A Survery of Kernels for Structured Data. *Multi Relational Data Mining*, 5(1):49–58, July 2003.

[12] K. Grauman and T. Darrell. Fast Contour Matching Using Approximate Earth Mover's Distance. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, Washington D.C., June 2004.

[13] E. Hadjidemetriou, M. Grossberg, and S. Nayar. Multiresolution Histograms and their Use for Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(7):831–847, July 2004.

[14] P. Indyk and N. Thaper. Fast Image Retrieval via Embeddings. In *3rd Intl Wkshp on Statistical and Computational Theories of Vision*, Nice, France, Oct 2003.

[15] Y. Ke and R. Sukthankar. PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, Washington, D.C., June 2004.

[16] R. Kondor and T. Jebara. A Kernel Between Sets of Vectors. In *Proccedings of International Conference on Machine Learning*, Washington, D.C., Aug 2003.

[17] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, Jan 2004.

[18] S. Lyu. Mercer Kernels for Object Recognition with Local Features. In *Tech Report 2004-520, Dartmouth College*, October 2004.

[19] K. Mikolajczyk and C. Schmid. Indexing Based on Scale Invariant Interest Points. In *Proceedings IEEE International Conference on Computer Vision*, Vancouver, Canada, July 2001.

[20] P. Moreno, P. Ho, and N. Vasconcelos. A Kullback-Leibler Divergence Based Kernel for SVM Classification in Multimedia Applications. In *Advances in Neural Information Processing*, Vancouver, Dec 2003.

[21] F. Odone, A. Barla, and A. Verri. Building Kernels from Binary Strings for Image Matching. *IEEE Trans. on Image Processing*, 14(2):169–180, Feb 2005.

[22] D. Roobaert and M. Van Hulle. View-Based 3D Object Recognition with Support Vector Machines. In *IEEE Intl Workshop on Neural Networks for Signal Processing*, Madison, WI, Aug. 1999.

[23] Y. Rubner, C. Tomasi, and L. Guibas. The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[24] J. Sivic and A. Zisserman. Video Google: A Text Retrieval Approach to Object Matching in Videos. In *Proceedings IEEE International Conference on Computer Vision*, Nice, France, Oct 2003.

[25] M. Swain and D. Ballard. Color Indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[26] V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, New York, 1998.

[27] C. Wallraven, B. Caputo, and A. Graf. Recognition with Local Features: the Kernel Recipe. In *Proceedings IEEE International Conference on Computer Vision*, Nice, France, Oct 2003.

[28] L. Wolf and A. Shashua. Learning Over Sets Using Kernel Principal Angles. *Journal of Machine Learning Research*, pages 913–931, Dec 2003.

[29] H. Zhang and J. Malik. Learning a Discriminative Classifier Using Shape Context Distances. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, Madison, WI, June 2003.