This is an extended version of an invited article that will appear in the Communications of the ACM (CACM) in 2009.

Efficiently Searching for Similar Images

Kristen Grauman
Department of Computer Sciences
University of Texas at Austin
Austin, TX, USA
grauman@cs.utexas.edu

ABSTRACT

As it becomes increasingly viable to capture, store, and share large amounts of image and video data, automatic image analysis is crucial to managing visual information. Many problems demand fast, accurate search of very large databases of images, but often the most effective metrics for image comparisons do not mesh well with known efficient search methods. Specifically, useful image representations use "structured" (non-vector) inputs that require specialized distance functions to compare, and the best use of side information complementing the visual data itself (e.g., partial annotations or tags) may require that a task-specific metric be learned. This paper overviews our research developing robust measures of image similarity intended to accommodate complex feature spaces and massive image databases. In particular, we overview efficient strategies for metrics that match local image features or learn from similarity constraints, and show how to perform sub-linear time search with both resulting distance functions.[1]

1. INTRODUCTION

If a tree falls in the forest and no one is there to hear it, does it make a sound? In the realm of content-based image retrieval, the question is: if an image is captured and recorded but no one is there to annotate it, *does it ever again make an appearance?* Over the last decade we have witnessed an explosion in the number and throughput of imaging devices. At the same time, advances in computer hardware and communications have made it increasingly possible to capture, store, and transmit image data at a low cost. Billions of images and videos are hosted publicly on the web; cameras embedded in mobile devices are commonplace. Climatologists compile large volumes of satellite imagery in search of long-term trends that might elucidate glacial activity and its impact on water supplies. Centralized medical image databases archive terabytes of X-ray, CAT scans, and ultrasound images, which may assist in new diagnoses.

Image and video data are certainly rich with meaning, memories, or entertainment, and in some cases they can facilitate communication or scientific discovery. However, without efficient vision algorithms to automatically analyze and index visual data, their full value will remain latent—the ratio of data to human attention is simply too large.

[1]This paper describes work done in collaboration with Trevor Darrell, Prateek Jain, Brian Kulis, and Tuyen Huynh, and summarizes ideas from several of our previous publications [19, 21, 25, 26].

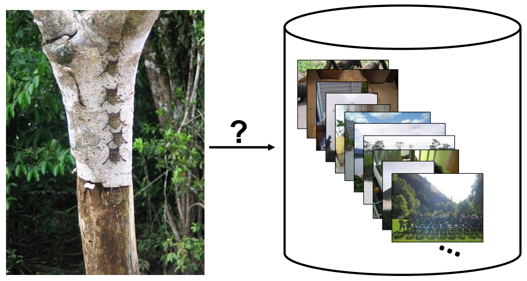


Figure 1: Visual data is complex and often holds valuable information. Image-based search algorithms automatically analyze and organize visual content, with the goal of allowing efficient retrieval from large image or video collections.

Most image search tools in operation today rely heavily on keyword meta-tags, where an image or video is annotated with a limited number of words that are either provided manually, or else are taken from whatever text occurs nearby in its containing document. While such a scheme simplifies the image indexing task to one that well-known information retrieval techniques can handle, it has serious shortcomings. At the surface, accurate manual tags are clearly too expensive to obtain on a large scale, and keywords in proximity to an image are often irrelevant.

Even more problematic, however, is the semantic disconnect between words and visual content: a word is a human construct with a precise intent, whereas a natural image can convey a multitude of concepts within its (say) million pixels, and any one may be more or less significant depending on the context. For example, imagine querying a database for all text documents containing the word "forest". Now imagine conjuring a text query that would find you all images relevant to the one on the left in Figure 1; while you immediately have a visual concept, it may be difficult to pinpoint words to capture it, especially if the objects within the image are unfamiliar. Thus, even if we were to somehow record keywords for all the images in the world, visual data would still not be sufficiently accessible.

Content-based image search streamlines the process by sorting images directly based on their visual information and allowing images themselves to serve as queries. While early work in the area focused on correlating low-level cues such as color and texture [40, 13], more recently the image search problem has become intertwined with the fundamental problem of recognition, in which algorithms must capture higher-level notions of visual object and scene categories.

The technical challenges are considerable. Instances of the same object category can generate drastically different images, depending on confounding variables such as illumination conditions, object pose, camera viewpoint, partial occlusions, and unrelated background "clutter" (see Figure 2). In general, the quality of image search relies significantly on the chosen image representation and the distance metric used to compare examples. Meanwhile, the complexity of useful image representations combined with the sheer magnitude of the search task immediately raises the practical issue of scalability.

This paper overviews our recent work considering how to construct robust measures of image similarity that can be deployed efficiently, even for complex feature spaces and massive image databases. We pose three essential technical questions: (1) what is an effective distance measure between images that can withstand the naturally occurring variability among related examples? (2) when external cues *beyond* observable image content are available, how can that improve our comparisons? and (3) what kind of search strategy will support fast queries with such image-driven metrics, particularly when our database is so large that a linear scan is infeasible? The following sections address each of these issues in turn, and highlight some of our results to demonstrate the impact with real image data.

Our approach enables rapid, scalable search for meaningful metrics that were previously restricted to artificially modestly sized inputs or databases. Additionally, we show how minimal annotations can be exploited to learn how to compare images more reliably. Both contributions support the ultimate goal of harnessing the potential of very large repositories and providing direct access to visual content.

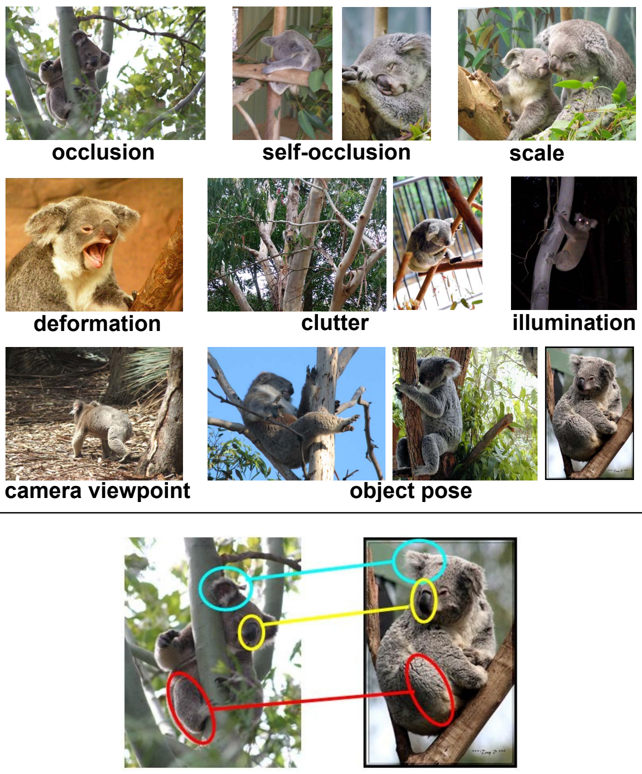


Figure 2: The same object type can generate dramatically different images due to a variety of nuisance parameters (top), but local descriptions can offer substantial robustness (bottom).

2. COMPARING IMAGES WITH LOCAL FEATURE MATCHES

Earlier work in content-based image retrieval focused on global representations that describe each image with a *single* vector of attributes, such as a color histogram, or an ordered list of intensity values or filter responses. While vector representations permit the direct application of standard distance functions and indexing structures, they are known to be prohibitively sensitive to realistic image conditions. For example, consider stacking the images in Figure 2 one on top of the other, and then checking the intensity at any given pixel for each example—it is quite likely that few of them would be in agreement, even though each image contains a koala as its most prominent object.

2.1 Local Image Representations

Much recent work shows that decomposing an image into its component parts (or so-called "local features") grants resilience to image transformations and variations in object appearance [30, 41, 47, 10, 34, 7, 38, 43]. Typically, one either takes a dense sample of regions at multiple scales, or else uses an interest operator to identify the most salient regions in an image. Possible salient points include pixels marking high contrast (edges), or points selected for a region's repeatability at multiple scales (see [41] for a survey). Then, for each detected region, a feature descriptor vector is formed. Descriptors may be lists of pixel values within a patch, or histograms of oriented contrast within the regions [30], for example. The result is one set of local appearance or shape description vectors per image, often numbering on the order of 2,000 or more features per image.

The idea behind such representations is to detect strong similarity between local portions of related images, even when the images appear quite different at the global level. Local features are more reliable for several reasons:

- **Isolate occlusions:** An object may be partially occluded by another object. A global representation will suffer proportionally, but for local representations, any local parts that are still visible will have their descriptions remain intact.
- **Isolate clutter and the background:** Similarly, while the global description may be overwhelmed by large amounts of background or clutter, small parts of an image containing an actual object of interest can emerge if we describe them independently by regions. Recognition can proceed without prior segmentation.
- **Accommodate partial appearance variation:** When instances of a category can vary widely in some dimensions of their appearance, their commonality may be

best captured by a part-wise description that includes the shared re-occurring pieces of the object class.

- **Invariant local descriptors:** Researchers have developed local descriptors designed explicitly to offer invariance to common transformations, such as illumination changes, rotations, translations, scaling, or all affine transformations.

This appealing representation—a set of vectors—does not fit the mold of many traditional distances and learning algorithms. Conventional methods assume vector inputs, but with local representations, each image produces a variable number of features, and there is no ordering among features in a single set. In this situation, computing a correspondence or matching between two images' features can reveal their overall resemblance: if many parts in image A can be associated with some similar-looking part in image B, then they are likely to display similar content (see Figure 2, bottom).

Current strategies for recognition and image matching exploit this notion in some form, often by building spatial constellations of a category's re-occurring local features [43], summarizing images with a histogram of discretized local patches [10, 38], or explicitly computing the least-cost correspondences [7, 46, 15] (see [35] for a survey). However, a real practical challenge is the computational cost of evaluating the optimal matching, which is cubic in the number of features extracted in an image. Compounding that cost is empirical evidence showing how much recognition accuracy improves when larger and denser feature sets are used [34].

2.2 The Pyramid Match Algorithm

To address this challenge, we developed the *pyramid match*—a linear-time matching function over unordered feature sets—and showed how it allows local features to be used efficiently within the context of multiple image search and learning problems [19, 20]. The pyramid match approximates the similarity measured by the *optimal partial matching* between feature sets of variable cardinalities. Because the matching is partial, some features may be ignored without penalty to the overall set similarity. This tolerance makes the measure robust in situations where superfluous or "outlier" features may appear. Note that our work focuses on the image matching and indexing aspects of the problem, and is flexible to the representation choice, i.e., which particular image feature detectors and descriptors are used as input.

We consider a feature space $\mathcal{V}$ of d-dimensional vectors for which the values have a maximal range D. The point sets we match will come from the input space S, which contains sets of feature vectors drawn from $\mathcal{V}$: $S = \{\mathbf{X} | \mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}\}$, where each feature $\mathbf{x}_i \in \mathcal{V} \subseteq \Re^d$, and $m = |\mathbf{X}|$. We can think of each $\mathbf{x}_i$ as a descriptor for one of the elliptical image regions on the koalas in Figure 2. Note that the point dimension d is fixed for all features in $\mathcal{V}$, but the value of m may vary across instances in S.

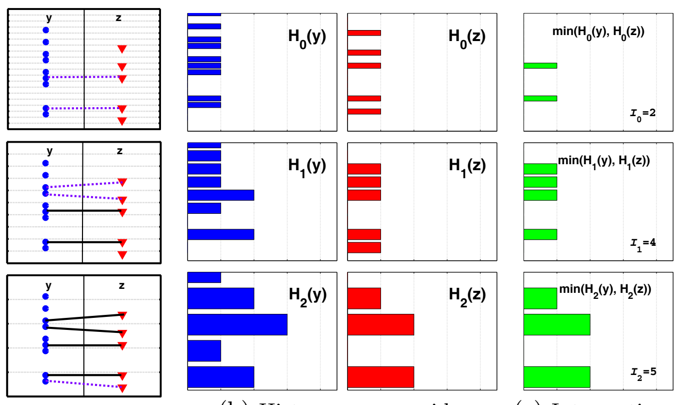
Given point sets $\mathbf{X}, \mathbf{Y} \in S$, with $|\mathbf{X}| \leq |\mathbf{Y}|$, the *optimal partial matching* π^* pairs each point in $\mathbf{X}$ to some unique point in $\mathbf{Y}$ such that the total distance between matched points is minimized: $\pi^* = \text{argmin}_\pi \sum_{\mathbf{x}_i \in \mathbf{X}} ||\mathbf{x}_i - \mathbf{y}_{\pi_i}||_1$, where π_i specifies which point $\mathbf{y}_{\pi_i}$ is matched to $\mathbf{x}_i$, and $|| \cdot ||_1$ denotes the L_1 norm. For sets with m features, the Hungarian algorithm computes the optimal match in $O(m^3)$ time [28], which severely limits the practicality of large input sizes. In contrast, the pyramid match approximation requires only $O(mL)$ time, where $L = \log D$, and $L \ll m$. In practice, this translates to speedups of several orders of magnitude relative to the optimal match for sets with thousands of features.

We use a multi-dimensional, multi-resolution histogram pyramid to partition the feature space into increasingly larger regions. At the finest resolution level in the pyramid, the partitions (bins) are very small; at successive levels they continue to grow in size until a single bin encompasses the entire feature space. At some level along this gradation in bin sizes, any two particular points from two given point sets will begin to share a bin in the pyramid, and when they do, they are considered matched. The key is that the pyramid allows us to extract a matching score without computing distances between any of the points in the input sets—the size of the bin that two points share indicates the farthest distance they could be from one another. We show that a weighted intersection of two pyramids defines an implicit partial correspondence based on the smallest histogram cell where a matched pair of points first appears.

Let a histogram pyramid for input example $\mathbf{X} \in S$ be defined as: $\Psi(\mathbf{X}) = [H_0(\mathbf{X}), \ldots, H_{L-1}(\mathbf{X})]$, where L specifies the number of pyramid levels, and $H_i(\mathbf{X})$ is a histogram vector over points in $\mathbf{X}$. The bins continually increase in size from the finest-level histogram H_0 until the coarsest-level histogram H_{L-1}. For low-dimensional feature spaces, the boundaries of the bins are computed simply with a uniform partitioning along all feature dimensions, with the length of each bin side doubling at each level. For high-dimensional feature spaces (e.g., $d > 15$), we use hierarchical clustering to concentrate the bin partitions where feature points tend to cluster for typical point sets [20]. In either case, we maintain a sparse representation per point set that maps each point to its bin indices. Even though there is an exponential growth in the number of possible histogram bins relative to the feature dimension (for uniform bins) or histogram levels (for non-uniform bins), any given set of features can occupy only a small number of them. An image with m features results in a pyramid description with no more than mL nonzero entries to store.

Two histogram pyramids implicitly encode the correspondences between their point sets, if we consider two points matched once they fall into the same histogram bin, starting at the finest resolution level. The matching is a hierarchical process: vectors not found to correspond at a fine resolution have the opportunity to be matched at coarser resolutions. Thus, for each pyramid level, we want to count the number of "new" matches—the number of feature pairs that were not in correspondence at any finer resolution level. For example, in Figure 3, there are two points matched at the finest scale, two new matches at the medium scale, and one at the coarsest scale.

To calculate the match count, we use histogram intersection, which measures the "overlap" between the mass in two his-



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

Figure 3: An example pyramid match. Here, two 1-D feature sets are used to form two histogram pyramids. Each row corresponds to a pyramid level. In (a), set Y is on the left, and set Z is on the right; points are distributed along the vertical axis. Light lines are bin boundaries, bold dashed lines indicate a new 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; (c) shows their intersections. The pyramid match function $\mathcal{P}_\Delta$ uses these intersection counts to measure how many new matches occurred at each level. Here, $\mathcal{I}_i = \mathcal{I}(H_i(\mathbf{Y}), H_i(\mathbf{Z})) = 2, 4, 5$ across levels, so the number of new matches counted are $2, 2, 1$. The weighted sum over these counts gives the pyramid match similarity. ***Figure is reprinted from [18] with permission, ©2007 AAAI.***

tograms: $\mathcal{I}(P, Q) = \sum_{j=1}^{r} \min(P_j, Q_j)$, where P and Q are histograms with r bins, and P_j denotes the count of the j-th bin. The intersection value effectively counts the number of points in two sets that match at a given quantization level. To calculate the number of *newly* matched pairs induced at level i, we only need to compute the difference between successive levels' intersections. By using the change in intersection values at each level, we count matches without ever explicitly searching for similar points or computing inter-feature distances.

The pyramid match similarity score $\mathcal{P}_\Delta$ between two input sets $\mathbf{Y}$ and $\mathbf{Z}$ is then defined as the weighted sum of the number of new matches per level:

$$\mathcal{P}_\Delta\left(\Psi(\mathbf{Y}), \Psi(\mathbf{Z})\right) = \sum_{i=0}^{L-1} w_i \Big(\mathcal{I}\left(H_i(\mathbf{Y}), H_i(\mathbf{Z})\right) - \mathcal{I}(H_{i-1}(\mathbf{Y}), H_{i-1}(\mathbf{Z}))\Big).$$

The number of new matches induced at level i is weighted by $w_i = \frac{1}{d2^i}$ to reflect the (worst-case) similarity of points matched at that level. This weighting reflects a geometric bound on the maximal distance between any two points that share a particular bin. Intuitively, similarity between vectors at a finer resolution—where features are more distinct—is rewarded more heavily than similarity between vectors at a coarser level.

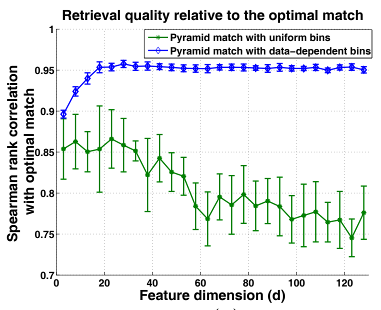
We combine the scores resulting from multiple pyramids with randomly shifted bins in order to alleviate quantization effects, and to enable formal error bounds. The approximation error for the pyramid match cost is bounded in the expectation by a factor of $C \cdot d \log D$ [22], for a constant C. We have also proven that the pyramid match kernel (PMK) naturally forms a *Mercer kernel*, which essentially means that it satisfies the necessary technical requirements to permit its use as a similarity function within a number of existing kernel-based machine learning methods.

Previous approximation methods have also considered a hierarchical decomposition of the feature space to reduce complexity [24, 9, 1, 3, 5]; the method in [24] particularly inspired our approach. However, earlier matching approximations assume equally-sized input sets, and cannot compute partial matches. In addition, while previous techniques suffer from distortion factors that are linear in the feature dimension, we have shown how to alleviate this decline in accuracy by tuning the hierarchical decomposition according to the particular structure of the data [20]. Finally, our approximation is unique in that it forms a valid Mercer kernel, and is useful in the context of various learning applications.

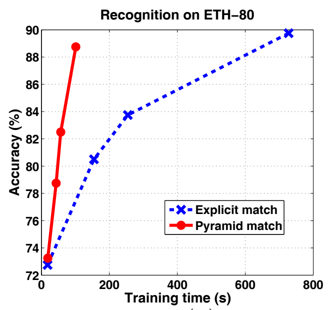
In short, the pyramid match gives us an efficient way to measure the similarity between two images based on the matching between their (potentially many) local features. Now, given a query image such as the one on the left of Figure 1, we can first extract descriptors for its local regions using any standard feature extractor [30, 41], and then find its relevant "neighbors" in the collection on the right by computing and sorting their pyramid match scores. In this way, we are able to search the image collection based on content alone.

Figure 4 shows some illustrative results using two well-known publicly available benchmark datasets, the ETH-80 [12] and Caltech-101 [8]. Both datasets are used to measure image categorization accuracy. The ETH collection is comprised of 80 object instances from eight different categories positioned on simple backgrounds; it is among the first benchmarks established for the categorization task, and since several categories are visually rather similar (e.g., horse and cow, apple and tomato), it is a good test for detailed discrimination. The Caltech collection, first introduced in 2003, contains 101 categories. It is challenging due to the magnitude of the multi-class problem it poses, and for many categories it offers noticeable intra-class appearance variation. It has been the subject of much attention in the research community and today stands as a key point of comparison for existing methods. For all the following results we employ the SIFT descriptor of [30], which is insensitive to shifts and rotations in the image yet provides a distinctive summary of a local patch.

The leftmost plot of Figure 4 demonstrates that when the pyramid match is used to sort the images from the ETH-80 database in a retrieval task, its complete ranking of the database examples is highly correlated to that of the optimal matching. The vertical axis measures how well results from two variants of the PMK agree with the optimal cubic-time results, and the horizontal axis shows the relative impact of the feature dimension d. While for low-dimensional features either a uniform or data-dependent partitioning of the feature space is adequate for good results, due to the curse of dimensionality, a data-dependent pyramid bin structure is



(a)



(b)

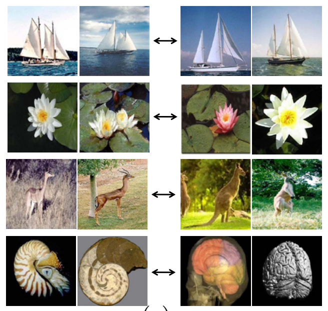

(c)

Figure 4: (a) The image rankings produced by the linear-time pyramid match are closely aligned with those produced by the cubic-time optimal matching. This plot shows how closely rankings computed with our approximate measure correlate with the optimal result, for features of increasing dimensionality. The vertical axis measures the rank correlation; perfect ranking agreement with the optimal measure would yield a score of 1. (b) More features per image lead to more reliable matches, but explicit matching techniques scale poorly with the representation size. The pyramid match makes large feature sets easily affordable. (c) The four category pairs in the Caltech-101 database that our method confused most.

much more effective for high-dimensional features.

The center plot shows accuracy as a function of computation time when the eight categories of the same dataset are learned using local feature matches between images. The plot compares the performance of the pyramid match to an exact matching function that averages the cost between the closest features in one set to the other. The horizontal axis measures the total training time, which is directly affected by the size of the feature sets. To vary the size of a typical set, we tune the saliency parameter controlling how many regions are detected per image. For both methods, more features lead to striking accuracy improvements; this behavior is expected since introducing more features assures better coverage of all potentially relevant image regions. However, the linear-time pyramid match offers a key advantage in terms of computational cost, reaching peak performance for significantly less computation time.

On the Caltech-101 benchmark, we have shown that classifiers employing the PMK with a variety of features currently yield the most accurate performance in the field for object recognition [27], with accuracy over 75% on the 101-way decision problem when training with just five exemplars per class, and 88% when training with 15. Figure 4(c) shows example images from four pairs of categories in the Caltech-101 dataset that cause the most confusion for the pyramid match: schooner and ketch, lotus and water lily, gerenuk and kangaroo, and nautilus and brain. In each row, the two images on the left have local features that match quite well to the two on the right, as compared to images from any of the other 100 classes in the dataset. Some of these confused category pairs have rather subtle distinctions in appearance. However, the case of the gerenuk and kangaroo reveals a limitation of the completely local description, as by definition it fails to capture the significance of the global contour shapes of the two objects.

Overall, approaches based on the pyramid match consistently show accuracy that is competitive with (or better than) the state-of-the-art while requiring dramatically less computation time. This complexity advantage frees us to consider much richer representations than were previously practical. Methods that compute explicit correspondences require about one minute to match a novel example; in the time that these methods recognize a single object, the pyramid match can recognize several hundred [22]. Due to its flexibility and efficiency, the pyramid match has been adapted and extended for use within a number of tasks, such as scene recognition [29], near-duplicate detection [44], human action recognition [31], and robot localization [32].

3. LEARNING IMAGE METRICS

Thus far, we have considered how to robustly measure image similarity in situations where we have no background knowledge; that is, where the system only has access to the image content itself. However, in many cases the system could also receive external side-information that might benefit its comparisons. For example, if provided with partially annotated image examples, or if a user wants to enforce similarity between certain types of images, then we ought to use those constraints to adapt the similarity measure.

A good distance metric between images accurately reflects the true underlying relationships, e.g., the category labels or other hidden parameters. It should report small distances for examples that are similar in the parameter space of interest (or that share a class label), and large distances for unrelated examples. Recent advances in metric learning make it possible to learn distance functions that are more effective for a given problem, provided some partially labeled data or constraints are available (see [45] and references within). By taking advantage of the prior information, these techniques offer improved accuracy when indexing examples. Typically, the strategy is to optimize any parameters to the metric so as to best satisfy the desired constraints.

Figure 5 (a) depicts how metric learning can influence image comparisons: the similarity (solid line) and dissimilarity (dotted lines) constraints essentially warp the feature space to preserve the specified relationships, and generalize to af-

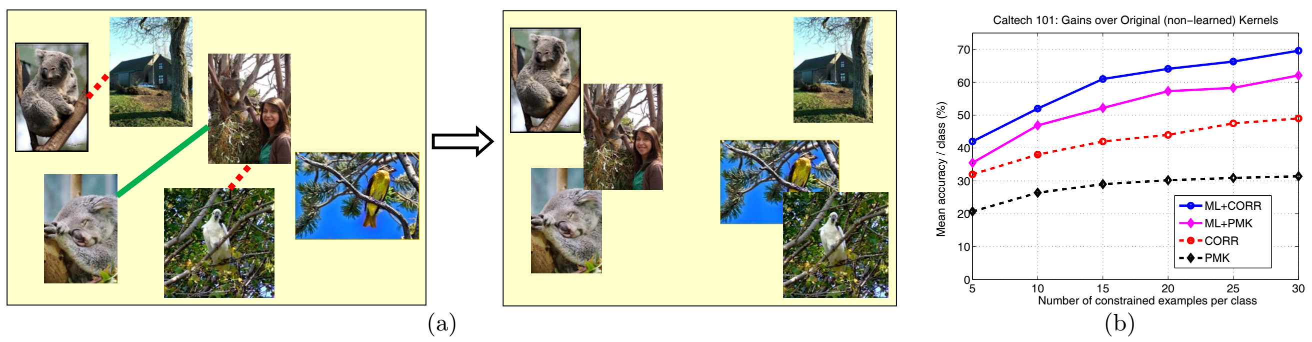


Figure 5: (a) By constraining some examples to be similar (green solid line), and others to be dissimilar (red dotted lines), the learned metric refines the original distance function so that examples are close only when they share the relevant features. (b) Retrieval accuracy is improved by replacing two matching-based image metrics (the PMK and CORR) with their learned counterparts (ML+PMK and ML+CORR). See text for details. *Plot is reprinted from [26] with permission, ©2008 IEEE.*

fect distances between other examples like them. In this illustrative example, even though we may measure high similarity between the greenery portions of both the cockatoo and koala images, the dissimilarity constraint serves to refocus the metric on the other (distinct) parts of the images.

In the case of the pyramid match, the weights associated with matches at different pyramid levels can be treated as learnable parameters. While fixing the weights according to the bin diameters allows the most accurate approximation of true inter-feature distances in a geometric sense, when we have some annotated images available, we can actually learn the weights that will best map same-class images close together [25].

The idea is that the best matching function is distinct for different classes of images, and is inherently defined by the variability a given class exhibits. For example, to distinguish one skyscraper from another, we might expect matchings between same-class examples to contain some very tight local feature correspondences, whereas to distinguish all skyscrapers from koalas, we would expect feature matches to occur at greater distances even among same-class examples. While the same *type* of image feature may be equally relevant in both situations, what is unique is the *distance* at which similarity is significant for that feature. Therefore, by learning the reward (weight) associated with each matching level in the pyramid, we can automatically determine how close feature matches must be in order to be considered significant for a given object class.

To achieve this intuition, we observe that the PMK can be written as a weighted sum of base kernels, where each base kernel is the similarity computed at a given bin resolution. We thus can compute the weights using a form of kernel alignment [4], where we find the optimal combination of kernel matrices that most closely mimics the "ideal" kernel on the training data, i.e., the one that gives maximal similarity values for in-class examples and minimal values for out-of-class examples. See [25] for details.

We have also considered how image retrieval accuracy can benefit from learning the *Mahalanobis parameterization* for several different base metrics, including matching functions [26]. Given points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, with $\boldsymbol{x}_i \in \Re^d$, a positive-definite $d \times d$ matrix A parameterizes the squared Mahalanobis distance:

$$d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^T A (\boldsymbol{x}_i - \boldsymbol{x}_j). \qquad (1)$$

A generalized inner product measures the pairwise similarity associated with that distance: $s_A(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T A \boldsymbol{x}_j$. Thus for a kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$, the parameters transform the inner product in the implicit feature space as $\phi(\boldsymbol{x}_i)^T A \phi(\boldsymbol{x}_j)$. Given a set of inter-example distance constraints, one can directly *learn* a matrix A to yield a measure that is more accurate for a given problem. We use the efficient method of [11] because it is kernelizable. This method optimizes the parameters of A so as to minimize how much that matrix diverges from an initial user-provided "base" parameterization, while satisfying constraints that require small distances between examples specified as similar, and large distances between pairs specified as dissimilar.

Figure 5 (b) shows the significant retrieval accuracy gains achieved when we learn image metrics using two matching-based kernels as the base metrics. The first kernel is the PMK, the approximate matching measure defined above. The second kernel was defined in [46], and uses exhaustive comparisons between features to compute a one-to-many match based on both descriptor and positional agreement; we refer to it as CORR for "correspondence". For this dataset of 101 object types, note that chance performance would yield an accuracy rate of only 1%. Both base metrics do the most they can by matching the local image features; the learned parameters adapt those metrics to better reflect the side-information specifying a handful of images from each class that ought to be near (or far) from the others.

4. SEARCHING IMAGE COLLECTIONS IN SUB-LINEAR TIME

Now that we have designed effective similarity measures, how will image search scale? We must be able to use these metrics to query a very large image database—potentially on the order of millions of examples or more. Certainly, a naive linear scan that compares the query against every database image is not feasible, even if the metric itself is efficient. Unfortunately, traditional methods for fast search cannot guarantee low query time performance for arbitrary

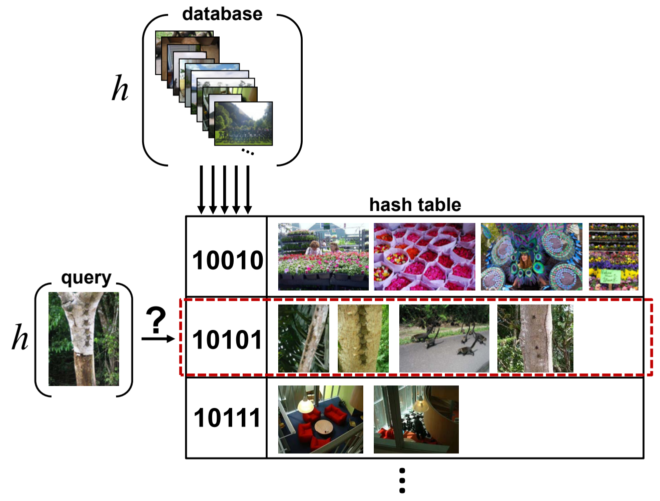


Figure 6: If hash functions guarantee a high probability of collision for images that are similar under a metric of interest, one can search a large database in sub-linear time via locality sensitive hashing techniques. The query hashes directly to bins with similar examples, and only that subset needs to be searched.

specialized metrics.[2] This section overviews our work designing hash functions that enable approximate similarity search for both types of metrics introduced above: a matching between sets, and learned Mahalanobis kernels.

The main idea of our approach is to construct a new family of hash functions that will satisfy the *locality-sensitivity* requirement that is central to existing randomized algorithms [23, 9, 2] for approximate nearest neighbor search. Locality sensitive hashing (LSH) has been formulated in two related contexts—one in which the likelihood of collision is guaranteed relative to a threshold on the radius surrounding a query point [23], and another where collision probabilities are equated with a similarity function score [9]. We use the latter definition here.

A family of LSH functions $\mathcal{F}$ is a distribution of functions where for any two objects $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$,

$$\Pr_{h\in\mathcal{F}}\left[h(\boldsymbol{x}_i) = h(\boldsymbol{x}_j)\right] = sim(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{2}$$

where $sim(\boldsymbol{x}_i, \boldsymbol{x}_j) \in [0, 1]$ is some similarity function, and $h(\boldsymbol{x})$ is a hash function drawn from $\mathcal{F}$ that returns a single bit [9]. Concatenating a series of b hash functions drawn from $\mathcal{F}$ yields b-dimensional hash keys. When $h(\boldsymbol{x}_i) = h(\boldsymbol{x}_j)$, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ collide in the hash table. Because the probability that two inputs collide is equal to the similarity between them, highly similar objects are indexed together in the hash table with high probability. On the other hand, if two objects are very dissimilar, they are unlikely to share a hash key (see Figure 6). Given valid LSH functions, the query time for retrieving $(1+\epsilon)$-near neighbors is bounded by $O(N^{1/(1+\epsilon)})$ for the Hamming distance and a database of size N [16]. One can therefore trade off the accuracy of the search with the query time required.

Note that Eqn. 2 is essentially a gateway to locality-sensitive hashing: if one can provide a distribution of hash functions guaranteed to preserve this equality *for the similarity function of interest*, than approximate nearest neighbor indexing may be performed in sub-linear time. Existing LSH functions can accommodate the Hamming distance [23], L_p norms [2], and inner products [9], and such functions have been explored previously in the vision community [37]. In the following we show how to enable sub-linear time search with LSH for metrics that are useful for image search.

4.1 Matching-sensitive hashing

Even though the pyramid match makes each individual matching scalable relative to the number of features per image, once we want to search a large database of images according to the correspondence-based distance, we still cannot afford a naive linear scan. To guarantee locality-sensitivity for a matching, we form an embedding function that maps our histogram pyramids into a vector space in such a way that the inner product between vectors in that space exactly yields the PMK similarity value [21].

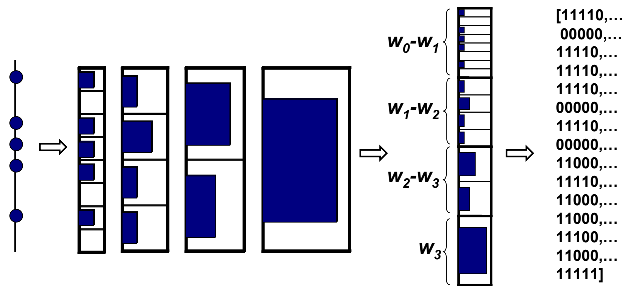
This re-mapping is motivated by the fact that randomized hash functions exist for similarity search with the inner product [9]. Specifically, in [17] it is shown that the probability that a hyperplane $\boldsymbol{r}$ drawn uniformly at random separates two vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is directly proportional to the angle between them: $\Pr\left[\text{sgn}(\boldsymbol{r}^T\boldsymbol{x}_i) \neq \text{sgn}(\boldsymbol{r}^T\boldsymbol{x}_j)\right] = \frac{1}{\pi}\cos^{-1}(\boldsymbol{x}_i^T\boldsymbol{x}_j)$. An LSH function that exploits this relationship is given in [9]. The hash function $h_{\boldsymbol{r}}$ accepts a vector $\boldsymbol{x} \in \Re^d$, and outputs a bit depending on the sign of its product with $\boldsymbol{r}$:

$$h_{\boldsymbol{r}}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{r}^T\boldsymbol{x} \geq 0 \\ 0, & \text{otherwise} \end{cases}. \tag{3}$$

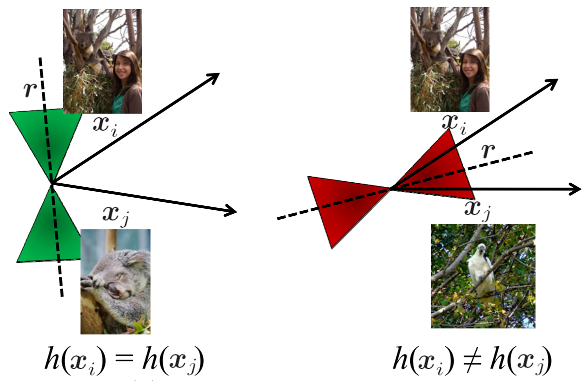
Since $\Pr\left[h_{\boldsymbol{r}}(\boldsymbol{x}_i) = h_{\boldsymbol{r}}(\boldsymbol{x}_j)\right] = 1 - \frac{1}{\pi}\cos^{-1}(\boldsymbol{x}_i^T\boldsymbol{x}_j)$, the probability of collision is high whenever the examples' inner product is high.

To embed the pyramid match as an inner product, we exploit the relationship between a dot product and the min operator used in the PMK's intersections. Taking the minimum of two values is equivalent to computing the dot product of a unary-style encoding in which a value v is written as a list of v ones, followed by a zero padding large enough to allot space for the maximal value that will occur. So, since a weighted intersection value is equal to the intersection of weighted values, we can compute the embedding by stacking up the histograms from a single pyramid, and weighting the entries associated with each pyramid level appropriately. Figure 7 (a) illustrates this process. Our embedding enables LSH for normalized partial match similarity with local features, and we have shown that it can achieve results very close to a naive linear scan when searching only a small fraction of an image database (1-2%). See [21] for more details.

[2]Data structures based on spatial partitioning and recursive decomposition have been developed for faster search, e.g. *k-d*-trees [14] and metric trees [42]. While their expected query time requirement may be logarithmic in the database size, selecting useful partitions can be expensive and requires good heuristics; worse, in high-dimensional spaces all exact search methods are known to provide little query time improvement over a naive linear scan [23]. Additionally, the expected query time for a *k-d* tree contains terms that are exponential in the dimension of the features [14], making them especially unsuitable for the pyramid representation where the dimension can be on the order of millions.



(a) Inner product embedding for the pyramid match



(b) Semi-supervised hash functions

Figure 7: (a) By appropriately embedding weighted pyramids into a vector space, a simple dot product will give the pyramid match similarity. This enables LSH-based similarity search for the pyramid match. (b) A generic hash function would choose the orientation of the hyperplane $\boldsymbol{r}$ uniformly at random, causing collisions only for examples that have small angles between their features ($\boldsymbol{x}_i$ and $\boldsymbol{x}_j$). In contrast, the distribution of our randomized semi-supervised hash functions is such that examples like those constrained to be similar are more likely to collide (left), while pairs like those constrained to be dissimilar are less likely to collide (right). Here the hourglass shapes denote the regions from which our randomized hash functions will most likely be drawn.

4.2 Semi-supervised hashing

To provide suitable hash functions for *learned* Mahalanobis metrics, we propose altering the distribution from which the randomized hyperplanes are drawn. Rather than drawing the vector $\boldsymbol{r}$ uniformly at random, we want to bias the selection so that similarity constraints provided for the metric learning process are also respected by the hash functions. In other words, we still want similar examples to collide, but now that similarity cannot be purely based on the image measurements $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$; it must also reflect the constraints that yield the improved (learned) metric (see Figure 7(b)). We refer to this as "semi-supervised" hashing, since the hash functions will be influenced by any available partial annotations, much as the learned metrics were above in Section 3.

In [26], we present a straightforward solution for the case of relatively low-dimensional input vector spaces, and further derive a solution to accommodate very high-dimensional data for which explicit input space computations are infeasible. The former contribution makes fast indexing accessible for numerous existing metric learning methods, while the latter is of particular interest for commonly used image representations, such as bags-of-words or multi-resolution histograms.

Given the matrix A for a metric learned as above, such that $A = G^T G$, we generate the following randomized hash functions $h_{\boldsymbol{r},A}$:

$$h_{\boldsymbol{r},A}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{r}^T G\boldsymbol{x} \geq 0 \\ 0, & \text{otherwise} \end{cases}, \qquad (4)$$

where the vector $\boldsymbol{r}$ is chosen at random from a d-dimensional Gaussian distribution with zero mean and unit variance.

By parameterizing the hash functions by both $\boldsymbol{r}$ and G, we enforce the following probability of collision:

$$\Pr\left[h_{\boldsymbol{r},A}(\boldsymbol{x}_i) = h_{\boldsymbol{r},A}(\boldsymbol{x}_j)\right] = 1 - \frac{1}{\pi}\cos^{-1}\left(\frac{\boldsymbol{x}_i^T A \boldsymbol{x}_j}{\sqrt{|G\boldsymbol{x}_i||G\boldsymbol{x}_j|}}\right),$$

which sustains the LSH requirement for a learned Mahalanobis metric. Essentially we have shifted the random hyperplane $\boldsymbol{r}$ according to A, and by factoring it by G we allow the random hash function itself to "carry" the information about the learned metric. The denominator in the cosine term normalizes the kernel values.

For low-dimensional data, we could equivalently transform all the data according to A *prior* to hashing. However, the matrix A has d^2 entries, and thus for very high-dimensional input spaces it cannot be represented explicitly, and we must work in the implicit kernel space. For example, for features like the histogram pyramids above, we have $d = 10^6$; the examples are sparse and representable, however the matrix A is dense and is not. This complicates the computation of hash functions, as they can no longer be computed directly as in Eqn. 4 above. To handle this, we derived an algorithm that simultaneously makes implicit updates to both the hash functions and the metric being learned. We show that it is possible to compute the value of $\boldsymbol{r}^T G$ indirectly, based on comparisons between the points involved in similarity constraints and the new example $\boldsymbol{x}$ that we want to hash. See [26] for details.

Figure 8 shows results using our semi-supervised hash functions to index the Caltech data. In the top plot, we see that the learned metric (denoted 'ML') again significantly improves the base metric in the image retrieval task. Additionally, we now can offer sub-linear time search even once the metric has been altered by the input similarity constraints. Note how accuracy varies as a function of ϵ, the parameter controlling how many examples we have to search per query; the more examples we can afford to search, the stronger our guarantee of approximating an exhaustive linear scan.

The bottom plot shows results using another database, $300K$ patches from the PhotoTourism project [39]. Here L_2 is the base metric; the recall rate is substantially improved once we learn a metric on top of it. Negligible accuracy is sacrificed when searching with our semi-supervised hash

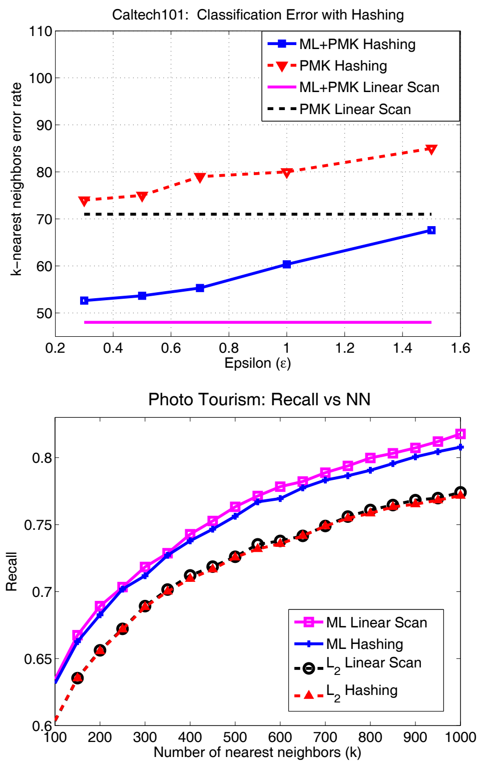


Figure 8: Semi-supervised hash functions encode the learned metric, and allow guaranteed sub-linear time queries that are similar in accuracy to a naive linear scan. ***Figure is reprinted from [26] with permission, ©2008 IEEE.***

functions (as seen by the closeness of the top two curves), yet our hashing strategy requires touching only 0.8% of the patches in the database. In our Matlab implementation, we observe speedup factors of about 400 relative to a linear scan for databases containing half a million examples. Due to the query time guarantees our hash functions enable, that factor grows rapidly with the size of the database.

Relative to traditional exact search data structures, the approximate hashing approach is critical to performance when inputs are high-dimensional. Modifications to classic tree structures have also been explored to improve search time with high-dimensional image features [6, 33], however such approaches cannot provide query-time guarantees, and are not applicable to searching with learned metrics. By hashing to buckets containing a collection of examples with a high probability of being very similar to the query, we are able to sort out the most relevant list of near neighbors. This is important for content-based retrieval, where we do not expect the single nearest exemplar to answer the query, but rather that the pool of nearby content will give the user and/or downstream processes access to relevant candidates.

5. CONCLUSIONS

As the world's store of digital images continues to grow exponentially, and as novel data-rich approaches to computer vision begin to emerge, fast techniques capable of accurately searching very large image collections are critical. The algorithms we have developed aim to provide robust but scalable image search, and results show the practical impact. While motivated by vision problems, these methods are fairly general, and may be applicable in other domains where rich features and massive data collections abound, such as computational biology or text processing.

Looking forward, an important challenge in this research area is to develop the representations that will scale in terms of their distinctiveness; once a data collection is truly massive, the space of images is even more densely populated, and relative differences are subtle. At the same time, flexibility is still key to handling intra-category variation. While our search methods can guarantee query-time performance, it is not yet possible to guarantee a level of discrimination power for the features chosen. In addition, a practical issue for evaluating algorithms in this space is the difficulty of quantifying accuracy for truly massive databases; the data itself is easy to come by, but without ground truth annotations, it is unclear how to rigorously evaluate performance.

An interesting aspect of the image search problem is the subjectivity related to a real user's perception of the quality of a retrieval. We can objectively quantify accuracy in terms of the categories contained in a retrieved image, which is helpful to systematically validate progress. Moreover, example-based search often serves as one useful stage in a larger pipeline with further processing downstream. Nonetheless, when end users are in the loop, admittedly the perception of quality may vary. On the evaluation side, this uncertainty could be addressed by collecting user appraisals of similarity, as is more standard in natural language processing [36]. In terms of the algorithms themselves, however, one can also exploit classic feedback and query refinement devices to tailor retrieval towards the current user. In our work, we could construct learned image metrics with constraints that target the preferences of a given user or group of users.

Currently we are exploring online extensions to our algorithms that will allow similarity constraints to be processed in an incremental fashion, while still allowing intermittent queries. We are also pursuing active learning methods that will allow the system to identify which image annotations seem most promising to request, and thereby most effectively use minimal manual input.

6. ACKNOWLEDGMENTS

I am fortunate to have worked with a number of terrific collaborators throughout the various stages of the projects overviewed in this paper—in particular, Trevor Darrell, Prateek Jain, Brian Kulis, and Tuyen Huynh. This research was supported in part by NSF CAREER award #0747356, a Microsoft Research New Faculty Fellowship, Texas Higher Education Coordinating Board award #003658-01-40-2007, and the Henry Luce Foundation. I would like to thank Kathryn McKinley and Yong Jae Lee for reading drafts of this paper, as well as the anonymous reviewers for their helpful comments. Thanks to the following Flickr users for sharing their photos under the Creative Commons license: belgian-chocolate, c.j.b., edwinn.11, piston9, staminaplus100, rick-yrhodes, Rick Smit, Krikit, Vanessa Pike-Russell, Will Ellis, Yvonne in Willowick Ohio, robertpaulyoung, lin padgham, tkcrash123, jennifrog, Zemzina, Irene2005, and CmdrGravy.

7. REFERENCES

[1] P. Agarwal and K. R. Varadarajan. A Near-Linear Algorithm for Euclidean Bipartite Matching. In *Symposium on Computational Geometry*, 2004.

[2] A. Andoni and P. Indyk. Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. *Communications of the ACM*, 51(1):117–122, 2008.

[3] D. Avis. A Survey of Heuristics for the Weighted Matching Problem. *Networks*, 13:475–493, 1983.

[4] F. Bach, G. Lanckriet, and M. Jordan. Multiple Kernel Learning, Conic Duality, and the SMO Algorithm. In *Proceedings of the International Conference on Machine Learning*, 2004.

[5] Y. Bartal. Probabilistic Approximation of Metric Spaces and its Algorithmic Applications. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, 1996.

[6] J. Beis and D. Lowe. Shape Indexing Using Approximate Nearest-Neighbour Search in High Dimensional Spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1997.

[7] 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, 2002.

[8] Caltech-101. `http://www.vision.caltech.edu/Image-Datasets/Caltech101/Caltech101.html`.

[9] M. Charikar. Similarity Estimation Techniques from Rounding Algorithms. In *ACM Symposium on Theory of Computing*, 2002.

[10] G. Csurka, C. Bray, C. Dance, and L. Fan. Visual Categorization with Bags of Keypoints. In *European Conference on Computer Vision*, Prague, Czech Republic, May 2004.

[11] J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon. Information-Theoretic Metric Learning. In *Proccedings of International Conference on Machine Learning*, 2007.

[12] ETH-80. `http://www.mis.informatik.tu-darmstadt.de/Research/Projects/categorization/eth80-db.html`.

[13] M. Flickner and et al. Query by Image and Video Content: The QBIC System. *IEEE Computer*, 28(9):23–32, 1995.

[14] J. Friedman, J. Bentley, and R. Finkel. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977.

[15] A. Frome, Y. Singer, F. Sha, and J. Malik. Learning Globally-Consistent Local Distance Functions for Shape-Based Image Retrieval and Classification. In *Proceedings of the IEEE International Conference on Computer Vision*, 2007.

[16] A. Gionis, P. Indyk, and R. Motwani. Similarity Search in High Dimensions via Hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, 1999.

[17] M. Goemans and D. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *Journal of the Association for Computing Machinery*, 42(6):1115–1145, 1995.

[18] K. Grauman. The Pyramid Match: Efficient Learning with Partial Correspondences. In *Proceedings of the Association for the Advancement of Artificial Intelligence*, 2007.

[19] K. Grauman and T. Darrell. The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features. In *Proceedings of the IEEE International Conference on Computer Vision*, 2005.

[20] K. Grauman and T. Darrell. Approximate Correspondences in High Dimensions. In *Advances in Neural Information Processing Systems*, 2006.

[21] K. Grauman and T. Darrell. Pyramid Match Hashing: Sub-Linear Time Indexing Over Partial Correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[22] K. Grauman and T. Darrell. The Pyramid Match Kernel: Efficient Learning with Sets of Features. *Journal of Machine Learning Research*, 8:725–760, 2007.

[23] P. Indyk and R. Motwani. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *30th Symposium on Theory of Computing*, 1998.

[24] P. Indyk and N. Thaper. Fast Image Retrieval via Embeddings. In *Intl Workshop on Statistical and Computational Theories of Vision*, 2003.

[25] P. Jain, T. Huynh, and K. Grauman. Learning Discriminative Matching Functions for Local Image Features. Technical report, University of Texas at Austin, April 2007.

[26] P. Jain, B. Kulis, and K. Grauman. Fast Image Search for Learned Metrics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[27] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Gaussian Processes for Object Categorization. *International Journal of Computer Vision*, 2009 (to appear).

[28] H. Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistic Quarterly*, 2:83–97, 1955.

[29] S. Lazebnik, C. Schmid, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[30] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2), 2004.

[31] F. Lv and R. Nevatia. Single View Human Action Recognition using Key Pose Matching and Viterbi Path Searching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[32] A. Murilloa, C. Saguesa, J. Guerreroa, T. Goedemé, T. Tuytelaars, and L. V. Gool. From Omnidirectional Images to Hierarchical Localization. *Robotics and*

Autonomous Systems, 55(5):372–382, May 2007.

[33] D. Nister and H. Stewenius. Scalable Recognition with a Vocabulary Tree. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[34] E. Nowak, F. Jurie, and B. Triggs. Sampling Strategies for Bag-of-Features Image Classification. In *Proceedings of the European Conference on Computer Vision*, 2006.

[35] A. Pinz. Object Categorization. *Foundations and Trends in Computer Graphics and Vision*, 1(4):255–353, 2006.

[36] H. Rubenstein and J. B. Goodenough. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633, 1965.

[37] G. Shakhnarovich, T. Darrell, and P. Indyk, editors. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. The MIT Press, 2006.

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

[39] N. Snavely, S. Seitz, and R. Szeliski. Photo Tourism: Exploring Photo Collections in 3D. In *SIGGRAPH*, 2006.

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

[41] T. Tuytelaars and K. Mikolajczyk. Local Invariant Feature Detectors: A Survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177–280, 2008.

[42] J. Uhlmann. Satisfying General Proximity / Similarity Queries with Metric Trees. *Information Processing Letters*, 40:175–179, 1991.

[43] M. Weber, M. Welling, and P. Perona. Unsupervised Learning of Models for Recognition. In *Proceedings of European Conference on Computer Vision*, 2000.

[44] D. Xu, T. Cham, S. Yan, and S.-F. Chang. Near Duplicate Image Identification with Spatially Aligned Pyramid Matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[45] L. Yang. Distance Metric Learning: A Comprehensive Survey. Technical report, Michigan State University, 2006.

[46] H. Zhang, A. Berg, M. Maire, and J. Malik. SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[47] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study. *International Journal of Computer Vision*, 73(2):213–238, 2007.