Assigning affinity-preserving binary hash codes to images

Jason Filippou
jasonfil@cs.umd.edu

Varun Manjunatha
varunm@umiacs.umd.edu

June 10, 2014

Abstract

We tackle the problem of generating binary hashcodes for retrieval in image databases. We try several methods and compare them against the state-of-the-art. Our algorithms are based on well-established ideas from the Unsupervised Learning literature and, in particular, Principal Component Analysis [16] and Vector Quantization [9]. We show that one of our algorithms, termed *unbalanced PQ-2Means*, performs similarly to the renowned Spectral Hashing algorithm [25], but is outperformed by ITQ [8].

1 Introduction

In recent years, the Computer Vision community has shown significant interest in the problem of producing binary codes for images. This task is frequently referred to as *image hashing*. A frequent miscommunication between the Vision and Databases community concerns the meaning of the term "hashing". In Databases, hashing consists of two steps. First is finding an appropriate transformation from the input space to a small discrete set, typically the subset of integers from zero up to a small prime integer.The transformation is known as a *hash function*.Second, it is required that a hashing method supplies a mechanism for resolving *collisions*, which occur when two items hash to the same number. In Vision, only the former element of hashing is examined. In a database setting, collisions are an unavoidable negative consequence of hashing and they affect performance negatively. In Vision, and Image Retrieval in particular, collisions can be beneficial, because they effectively mean that - depending on the properties of the hash function used - the images that collide or nearly collide are in some sense "similar" and this should be made evident to the user.

In the Image Retrieval setting that we examine, the hash functions are essentially binary embeddings, that is, the input image is transformed into binary strings. The advantages of this transformation are two-fold. First, in terms of storage. If we have just 4GB of main memory, storing every image as a 32-bit hashcode results in over a billion images possibly being stored. Second, in terms of retrieval efficiency. If the embedding is such that the Hamming distance between the hashcodes respects the locality structure of the images in the original domain, then even the naive nearest neighbor search algorithm of complexity $O(n \cdot d)$ can be computed very efficiently, given highly optimized code for computing the Hamming Distance between two binary strings. Furthermore, the embedding makes approximate similarity techniques well-studied in the Data Mining domain, such as MinHash[4], possible.

In the Database community, it might seem counter-intuitive that locality-preserving hash functions are useful. Database researchers are often exposed to *cryptographic* hash functions, a key property of which is -approximately- uniform key distribution. The Vision community, however, has shown time and again that similarity preserving hash functions (binary embeddings) are possible. Locality Sensitive Hashing (LSH) [11] is one of the first methods that successfully demonstrated the concept. In [1], the authors introduce locality-sensitive hash function families for the Hamming, L_1 and L_2 distances.

From a Data Structures perspective, the use of hashing for finding similar images might seem hasty at first. One might advocate the use of spatial data structures such as quadtrees[19] or kd-trees [3] for

finding similar images in the original d-dimensional space. Unfortunately, the former suffer from an exponential blow-up in the nodes of each level as d grows large (which is why they are typically limited to at most 3 dimensions, at which point they are called *oct-trees*) and the nearest-neighbor problem in a kd-tree takes $O(2^d + logn)$ time to solve for a perfectly balanced kd-tree (and there are no balance guarantees in the structure). Therefore, tree-based representations are ill-suited to solve the general image retrieval problem, and we resort to hashing schemes. The obvious trade-off is that now we have probabilistic guarantees for finding an *approximate* nearest neighbor instead of the best set of nearest neighbors. This is nothing new, and it has been examined in previous setings as well. [4]

The remainder of this report is organized as follows. Section 2 outlines the related work on the field with particular emphasis on the methods that inspired us and we compared our algorithms with. Section 3 outlines the algorithms that we tried with particular emphasis on two: *PCA_Direct* and *PQ-2Means*, and discusses their pros and cons. Section 4 contains our experimental results. We conclude in Section 5 and give pointers for future work.

2 Related work

The Vision literature is replete with works introducing similarity-preserving binary embedding methods. See [8, 22, 11, 5, 12, 17, 25, 18, 24] for some representative examples. In this section, we will focus on what are arguably the most influential of these methods, namely Spectral Hashing [25], Product Quantization [12] and ITQ [8].[1]

Spectral Hashing (SH) has been shown [25] to outperform Semantic Hashing [18], which is itself based on the Restricted Boltzmann Machine [10]. In [25], the authors show that the problem of finding an optimal binary code given a dataset is NP-Hard. By relaxing the original constraints, a closed-form eigenvalue solution is obtained which is based on thresholding the eigenvectors of the dataset's graph Laplacian. Despite the fact that SH has been since outperformed by the state-of-the-art methods [12, 8], we consider some of the insights learned from the algorithm to be important. Firstly, the authors of [25] outline the properties that a good binary code should have. Those should be:

(a) The code should be short enough to simultaneously allow storing a large number of images in memory.

(b) The code for a test point should be computed fast.

(c) The code should be *similarity-preserving*, in the sense that distances in the original space should be correlated with distances in the Hamming space.

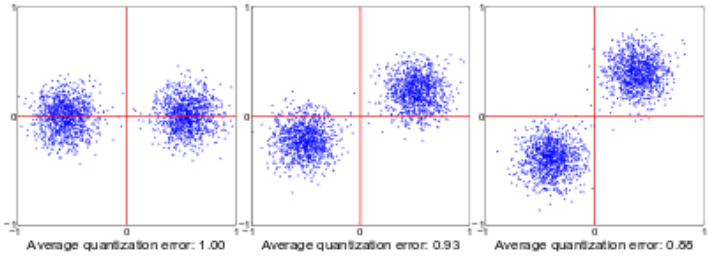
Furthermore, one of the major novelties of SH is that its optimization problem constrains the solution such that the produced bits are *pairwise uncorrelated* and that each bit fires *50% of the time*. The reasoning behind both of these constraints is intuitive; for the former constraint, a bit firing at index i should have no bearing about whether a bit at index j should fire, otherwise the utility of j is put into question. Similarly, if we have an encoding which consists of k bits and one of these bits fires 99% of the time, then we could just as well get rid of that bit, since its *information content* [20] is very small.

Product Quantization (PQ) [12] is the next important work that we focus on. In this framework, Vector Quantization [9] is used to decompose the original space into a cartesian product of low-dimensional subspaces. These subspaces are clustered, and the input vectors are now represented as a sequence of subspace cluster indices. Two different distance metrics, one termed "symmetric" and the other "asymmetric" are devised and their efficacy with respect to finding approximate nearest neighbors is estimated. The method demonstrated top-of-the-line results at the time, outperforming SH by using both local patch-based image descriptors (SIFT [14], SURF [2]) and global descriptors (GIST [23]) for variable-length codes. It should be mentioned, however, that, in and of itself, PQ is *not* a binary embedding method; it is simply a technique for approximating the Euclidean distance between two vectors by the distance between certain cluster centroids, such that the quality of the approximation to the true nearest neighbors is high. As we will show later on, however, it is possible to use PQ to find high-quality

[1] Technically speaking, Product Quantization is not a binary embedding method, but we demonstrate that it can be considered such.

binary embeddings.

Finally, Iterative Quantization (ITQ) [8] is currently the state-of-the-art method in producing binary codes for retrieval. The idea behind ITQ is to first start by centering the data and projecting it onto its principal components. Then, the sign of the projections is taken, yielding certain binary codes. As is made evident by Figure 1, however, this simple embedding does not necessarily respect the locality structure of the original data. For this reason, an iterative Expectation/Maximization-like process is devised to rotate the points as close as possible to the vertices of a binary hypercube, thus minimizing the quantization error of the binary mapping. Experimental results on the TinyImages[21] and CIFAR datasets [13] reveal that ITQ outperforms all other known binary embedding methods, such as Semantic Hashing [18], Shift-Invariant Kernels[17], LSH[11] and Spectral Hashing [25]. A comparison with PQ is not included because, as we mentioned earlier, PQ has not, until now, been perceived as a method that might produce binary embeddings.



(a) PCA aligned. (b) Random Rotation. (c) Optimized Rotation.

Figure 1: Iterative Quantization.

Our method is mostly influenced by PQ. In a nutshell, the same way that ITQ begins with PCA embeddings and then essentially performs clustering in order to find the optimal rotation to minimize the quantization error of the data projections, we begin with spliting the original vectors into orthogonal subspaces and then perform k-means clustering with $k = 2$, hence the name *PQ-2means*. When a test point arrives, we split it into r orthogonal subspaces and then assign each and every one of these sub-vectors into its closest cluster, as computed at training time. This produces a binary embedding of length r. As will be shown in Section 4, this remarkably simple approach performs as well as SH, which is currently the second-best method for learning binary embeddings. It is lacking, however, when compared to the much more sophisticated ITQ.

3 Algorithms

In this section, we detail the algorithms that we used. We tried a large number of algorithms, but only two of them ended up having reasonable performance and we will thus focus on those the most.

3.1 PCA-Direct

Before we even surveyed the literature, we came up with the idea that projecting the data onto its principal components and taking the sign of the projections seemed to yield a conceptually correct result. Figure 2 shows a qualitative view of the process. An obvious problem with this approach is that it tends to cluster large portions of the data elements into similar distances. However, this is a common problem with all of the approaches in the literature. While it is true that the space of possible Hamming encodings is exponential in the length of the bitcodes, which means that we can store a tremendous amount of data with a relatively short code, the space of possible Hamming distances is linear [2]. This means that it is possible, in these problems, to generate false positives; this problem is typically alleviated by performing an additional post-processing step to filter out such false positives by ranking the returned neighbors [11].

[2]For k bits, we have $k + 1$ possible Hamming distances between two vectors: Either both vectors agree on all bits, or they disagree on k different possible index combinations

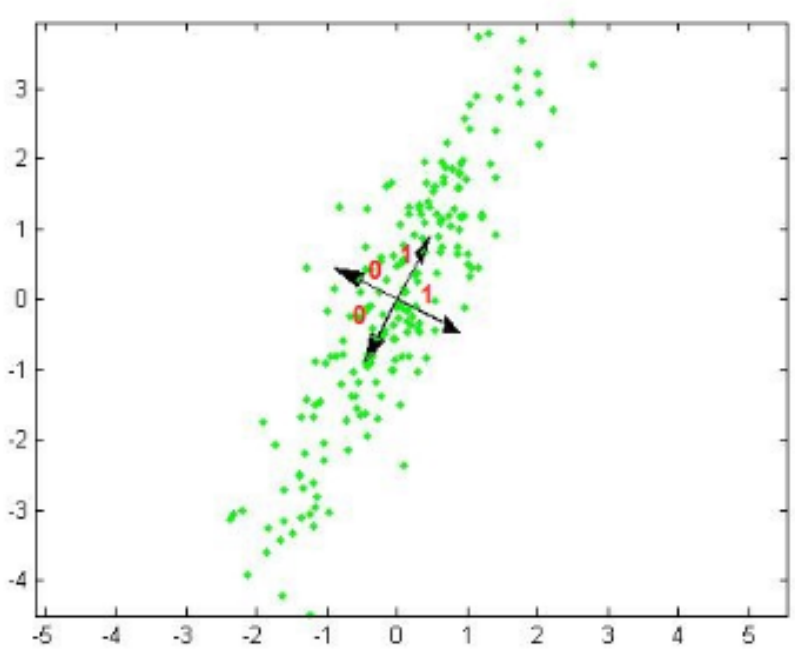


Figure 2: PCA Direct in 2 dimensions.

A closer look at the literature, however, revealed that PCA-Direct was compared against as a baseline of ITQ [8]. Recall that ITQ begins from PCA-Direct and iteratively rotates the data until it is optimally mapped to the vertices of the k-dimensional binary hypercube. In fact, one of the interesting findings of [8] was that even if a random rotation was applied to the result of PCA-Direct (a technique that they call *PCA-RR* for "PCA with Random Rotation"), the results can be significantly improved.

Nevertheless, we include PCA-Direct in our list of attempted methods, it is important to investigate how it rates compared to our other methods and the competition.

3.2 PQ-2 means

Our most promising method was a combination of PQ and 2-means clustering. Inspired by [12], we first split the input vectors into sub-vectors of the same dimensionality. Following that, we perform k-means clustering, where we constrain k to be 2, hence the full name "PQ-2Means". Influenced by [25], where the different bits were constrained to fire 50% of the time and to be pairwise uncorrelated, we consider two different approaches: a "balanced" 2-means, where the two clusters are constrained to contain the same number of clusters and a classic, "unbalanced" 2-means. Figure 3 gives a visual interpretation of how these algorithms differ.

The main intuition behind using a "balanced" 2-means is that, when the two clusters formed by the subspaces are of the same cardinality, the training data's bits will all fire exactly 50% of the time (since a '1' will be assigned to exactly half of the vectors at that particular index, and the other half vectors will have a '0' at the same index).To perform this balanced 2-means clustering, we employed the following algorithm :

Balanced 2-means :

1. Initialize centers using the k-means++ algorithm.
2. While $n\_iterations \leq MAX\_ITERATIONS$
   (a) Create k rank lists which arrange points in order of proximity from each center.
   (b) Assign one point from the ith rank list and remove it from all the other rank lists.
   (c) Iterate amongst all the rank lists.

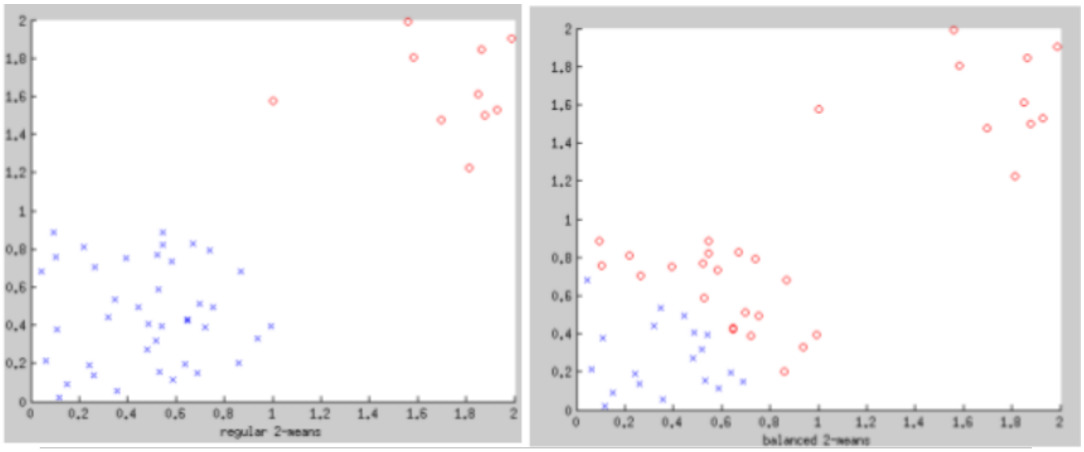


Figure 3: Unbalanced (regular) vs Balanced 2-means

(d) Recompute the centers using the new assignments.

(e) If the centers converge, break.

3.3 Other methods

In addition to the above methods, we tried some other techniques, which did not yield good results. We describe our methods below :

- In Spectral Hashing, each bit is given equal importance. However, we suspect that since the eigenvalues are arranged in increasing order, the bits are actually of unequal importance. Therefore, we tried to compute Hamming distances by weighting the bits with the eigenvalues. However, this yielded poor results, likely because our weighting scheme was extremely naive. Still, this could be a promising direction for research.
- Our product quantization approach involved splitting the feature space into subspaces, so that each subspace could be used to compute a single bit. A different way to do the subspace sampling would be random draws with replacement from the feature space. However, this method performed worse than our previous unbalanced 2-means approach.
- Finally, we tried to perform a 4-means clustering, rather than a 2-means, to allocate two bits per subspace. Our motivation behind this was that a vector space with four clusters would lead to more non-linear clusters than with 2-means. In fact, the experiments of [12] support the strategy of increasing the number of clusters per subspace, instead of increasing the number of subspaces and keeping the number of clusters low. However, this approach did not do better than our unbalanced 2-means approach either. We suspect this is because our assignment of bits to cluster centers was not optimal.

4 Experiments

To test our algorithms, we use image features from the Scene UNderstanding (SUN dataset)[15] dataset. Out of the 19000 dimensional feature vectors, 1005 were randomly selected. We then performed centering and unit normalization on these features, and plotted precision recall curves for the various methods over a retrieval problem. We computed nearest-neighbor ground truth by brute force. The results are visualized in Figure 4. Overall, we find that ITQ works extremely well for this dataset, while unbalanced PQ-2means works as well as Spectral Hashing. Balanced PQ-2means and PQ-2means with random sub-space draws all outperform the base-line of PCA Direct.

5 Conclusions / Future Work

In this project, we investigated several simple, yet for the most part novel approaches for producing similarity-preserving binary hash codes for image retrieval. We compare our algorithms against the

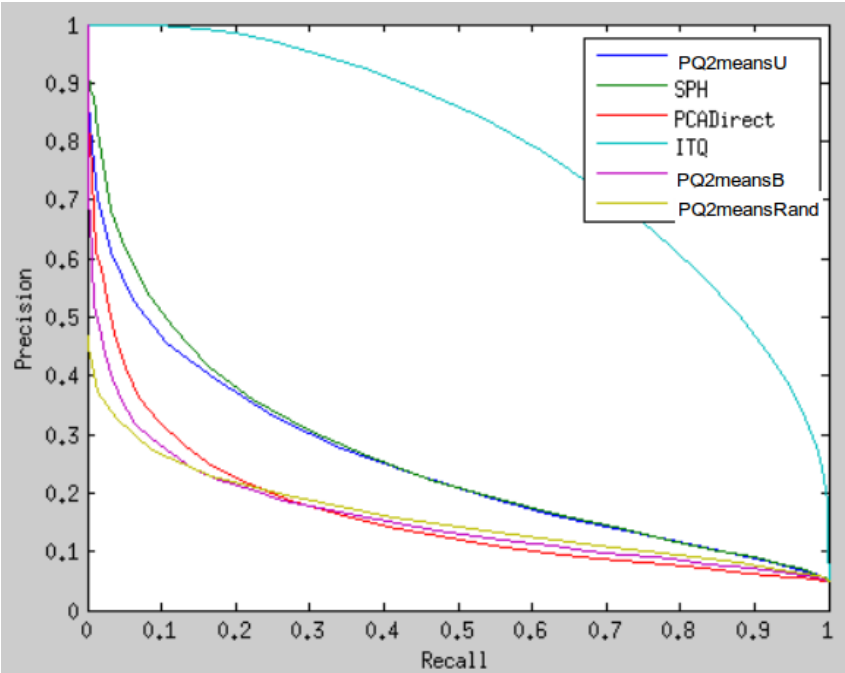


Figure 4: Precision-Recall curve showing various methods. PQ2meansU refers to unbalanced 2-means, PQ2meansB refers to balanced 2-means and PQ2meansRand refers to 2-means where the sub-spaces for product quantization are random draws from the feature space.

state-of-the-art on a portion of the SUN dataset and draw several important conclusions. While our method does not beat the competition, it comes in second, performing similarly to the very popular Spectral Hashing algorithm. It currently appears as if Iterative Quantization is unbeatable by any other method. It seems as if the only way to perform better is simply to start with ITQ and optimize over it. However, we arrive to some very important conclusions.

First and foremost, it is very easy to reach the level of performance exhibited by Spectral Hashing. We do so without solving a constrained optimization problem and we do not have to compute an out-of-sample extension for the test points. Our method is therefore inherently faster and easier to conceptualize. More importantly, our solution demonstrates that the Shannon-inspired heuristic of requiring the bits to be uncorrelated and making them 50% likely to fire is not a requirement for good codes. This has been theorized over the last few years by the community and in our lab in particular.

Second, Iterative Quantization is currently by far the strongest method. It seems as if the only way to beat ITQ is by using it as a "stepping stone" and doing some further post-processing (e.g [7]).

Third, we expected balanced PQ-2means to perform better than unbalanced PQ-2means, but this was not found to be the case. Perhaps the balanced assumption that we made about our data was incorrect. In any case our results indicate that the "balancing argument" may not be sound. A further assumption, that the bits be uncorrelated with each other, is likely to be universally true across all datasets.

Lastly, our work opens up broad avenues for research. This project, in conjunction with recent works (e.g [6]) show that PQ can be used successfully for the task of Image Retrieval. The algorithms presented in [6, 8] are all iterative in nature, which is a different approach than that taken in SH and similar works: instead of solving a relaxed optimization problem in a spectral framework, an attempt is made to reach an approximately optimal solution through Vector Quantization. In the future, we want to experiment further with our method in other datasets beyond SUN, where the variance of every dimension might be markedly different.

Acknowledgements

The authors would like to thank UMD Computer Science student Mohammad Rastegari for supplying evaluation code and spending time with us on enlightening discussions.

References

[1] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008.

[2] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[3] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975.

[4] A.Z. Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*, pages 21–29, Jun 1997.

[5] O. Chum, M. Perdoch, and J. Matas. Geometric min-hashing: Finding a (thick) needle in a haystack. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 17–24, June 2009.

[6] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized Product Quantization for Approximate Nearest Neighbor Search. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 0:2946–2953, 2013.

[7] Tiezheng Ge, Kaiming He, and Jian Sun. Product Sparse Coding. In *Computer Vision and Pattern Recognition*, 2014.

[8] Yunchao Gong and Svetlana Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, pages 817–824, 2011.

[9] R.M. Gray and D.L. Neuhoff. Quantization. *Information Theory, IEEE Transactions on*, 44(6):2325–2383, Oct 1998.

[10] G E Hinton and R R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006.

[11] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM.

[12] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(1):117–128, 2011.

[13] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, 2009.

[14] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1150–, Washington, DC, USA, 1999. IEEE Computer Society.

[15] Genevieve Patterson and James Hays. Sun attribute database: Discovering, annotating, and recognizing scene attributes. In *Proceeding of the 25th Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[16] Karl Pearson. LIII. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine Series 6*, 2(11):559–572, 1901.

[17] Maxim Raginsky and Svetlana Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *NIPS*, pages 1509–1517, 2009.

[18] Ruslan Salakhutdinov and Geoffrey E. Hinton. Semantic hashing. *Int. J. Approx. Reasoning*, 50(7):969–978, 2009.

[19] Hanan Samet. The quadtree and related hierarchical data structures. *ACM Comput. Surv.*, 16(2):187–260, 1984.

[20] Claude Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, July, October 1948.

[21] Antonio Torralba, Robert Fergus, and William T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(11):1958–1970, 2008.

[22] Antonio Torralba, Robert Fergus, and Yair Weiss. Small codes and large image databases for recognition. In *CVPR*, 2008.

[23] Antonio Torralba, Kevin P. Murphy, William T. Freeman, and Mark A. Rubin. Context-based vision system for place and object recognition. In *ICCV*, pages 273–280, 2003.

[24] Jun Wang, S. Kumar, and Shih-Fu Chang. Semi-supervised hashing for scalable image retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3424–3431, June 2010.

[25] Yair Weiss, Antonio Torralba, and Robert Fergus. Spectral Hashing. In *NIPS*, pages 1753–1760, 2008.