Locality Sensitive Hash Functions Based on Concomitant Rank Order Statistics

Kave Eshghi and Shyamsundar Rajaram
HP Laboratories
HPL-2007-192R1

Keyword(s):
Locality Sensitive Hashing, Order Statistics, Concomitants, Image Similarity, Discrete Cosine Transform.

Abstract:
Locality Sensitive Hash functions are invaluable tools for approximate near neighbor problems in high dimensional spaces. In this work, we are focused on LSH schemes where the similarity metric is the cosine measure. The contribution of this work is a new class of locality sensitive hash functions for the cosine similarity measure based on the theory of concomitants, which arises in order statistics. Consider n i.i.d sample pairs, $f(X1; Y1); (X2; Y2); : : : ; (Xn; Yn)g$ obtained from a bivariate distribution $f(X; Y)$. Concomitant theory captures the relation between the order statistics of X and Y in the form of a rank distribution given by Prob(Rank(Yi)=jjRank(Xi)=k). We exploit properties of the rank distribution towards developing a locality sensitive hash family that has excellent collision rate properties for the cosine measure.

The computational cost of the basic algorithm is high for high hash lengths. We introduce several approximations based on the properties of concomitant order statistics and discrete transforms that perform almost as well, with significantly reduced computational cost. We demonstrate the practical applicability of our algorithms by using it for finding similar images in an image repository.

External Posting Date: July 6, 2008 [Fulltext] Approved for External Publication

Internal Posting Date: July 6, 2008 [Fulltext]

To be presented and published in the 14th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'08), August 2008

Locality Sensitive Hash Functions Based on Concomitant Rank Order Statistics

Kave Eshghi & Shyamsundar Rajaram
Hewlett Packard Laboratories
Palo Alto, CA
{kave.eshghi,shyam.rajaram}@hp.com

Abstract

Locality Sensitive Hash functions are invaluable tools for approximate near neighbor problems in high dimensional spaces. In this work, we are focused on LSH schemes where the similarity metric is the cosine measure. The contribution of this work is a new class of locality sensitive hash functions for the cosine similarity measure based on the theory of concomitants, which arises in order statistics. Consider n i.i.d sample pairs, $\{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$ obtained from a bivariate distribution $f(X, Y)$. Concomitant theory captures the relation between the order statistics of X and Y in the form of a rank distribution given by Prob(Rank(Y_i)=j|Rank(X_i)=k). We exploit properties of the rank distribution towards developing a locality sensitive hash family that has excellent collision rate properties for the cosine measure.

The computational cost of the basic algorithm is high for high hash lengths. We introduce several approximations based on the properties of concomitant order statistics and discrete transforms that perform almost as well, with significantly reduced computational cost. We demonstrate the practical applicability of our algorithms by using it for finding similar images in an image repository.

1 Introduction

This paper is about a new family of Locality Sensitive Hash functions for the cosine distance measure. Traditionally, nearest neighbor search in high dimensional spaces has been expensive, because with increasing dimensionality indexing schemes such as KD Trees very quickly deteriorate to a linear scan of all the items. Locality Sensitive Hash Functions [7] were introduced to solve the approximate nearest neighbor problem in high dimensional spaces and several advancements [4, 1, 2, 11] have been done in this area. Simply put, a locality sensitive hash function is designed in such a way that if two vectors are close in the intended distance measure, the probability that they hash to the same value is high; if they are far in the intended distance measure, the probability that they hash to the same value is low. Thus the hashes of the objects can be used to create an index whereby approximate close neighbor search could be achieved efficiently. The details of what 'close' and 'far' mean depend on the distance measure used, and there are different exact formulations depending on the distance measure.

In this work, we are concerned with cosine similarity, i.e., the case where the similarity between two vectors v_1 and v_2 is measured as $cos(v_1, v_2)$, and the distance is measured as $1 - cos(v_1, v_2)$. The cosine measure of similarity is a popular one for a variety of applications such as in document retrieval [13], natural language processing [12] and image retrieval [14]. In this paper we show that it is an appropriate measure of similarity in the image similarity setting where we use PCA SIFT [8][10] descriptors to represent images.

The most successful existing LSH scheme for cosine similarity is the random hyperplane method [2]. The random hyperplane method suffers from a rapid loss in collision rate with increasing hash length. The contribution of this work is a new family of locality sensitive hash functions for the cosine measure based on the concomitant order statistics of the bivariate normal distribution, which overcomes the above issue. We will compare our scheme with the random hyperplane method and demonstrate its improved collision rate both on a theoretical basis and the resulting impact in a practical setting.

The rest of the paper is structured as follows. In Sec. 2, we introduce notation and basics that sets up the rest of the paper. In Sec. 3, we review the random hyperplane hash family. Sec. 4 introduces the theory behind concomitants, which we use to develop the concomitant hash family. In Sec. 5, we extend the concomitant hash family to generate multiple hashes per vector. In their most simple form, the computational cost of concomitant based hash functions rise exponentially with the length of the hash. In Sec. 6 and Sec. 7, we provide approximations to the hash function that significantly reduce the computational cost of these hash functions. In Sec. 8, we present the superior performance of our hashing algorithm in an image similarity setting.

2 Basics

Definition 1 (Locality Sensitive Hashing [2]). *A locality sensitive hashing scheme is a distribution on a family $\mathcal{F}$ of hash functions on a set of items, such that for two items x and y,*

$$\mathbf{Pr}_{h\in\mathcal{F}}\left[h(x) = h(y)\right] = f(\mathrm{sim}(x, y)) \tag{1}$$

where, $\mathrm{sim}(x, y)$ *is some similarity function defined on the item collection and f is a monotonically increasing function.*

Definition 2 (Cosine Similarity). *The cosine of two vectors $A \in \mathbb{R}^m$ and $B \in \mathbb{R}^m$ is defined as* $cos(A, B) = \frac{A.B}{|A||B|}$

When the the two vectors are zero centered (i.e. the mean of the vectors is zero), the cosine measure

is the same as the correlation coefficient between the vectors [1]. In some applications where the vectors are very high dimensional and sparse, e.g. in the vector model of document retrieval, the vectors are not zero centered, but the mean of the vectors is very close to zero anyway, due to the sparseness and high dimensionality.

Definition 3 (Cosine Hash Family). *A set of functions* $H = \{h_1, h_2, \ldots\}$ *constitute a cosine hash family over* $\mathbb{R}^m$ *iff for some finite* $\mathbf{U} \subset \mathbb{N}$*,*

- $h_k \in H : \mathbb{R}^m \to \mathbf{U}$
- *For any four vectors* A, A', B, B' *in* $\mathbb{R}^m$ *where* $\cos(A, A') = \cos(B, B')$*, the following is true:*

$$\mathbf{Pr}_{h \in H}(h(A) = h(A')) = \mathbf{Pr}_{h \in H}(h(B) = h(B'))$$

- *For any four vectors* A, A', B, B' *in* $\mathbb{R}^m$ *where* $\cos(A, A') > \cos(B, B')$*, the following is true:*

$$\mathbf{Pr}_{h \in H}(h(A) = h(A')) > \mathbf{Pr}_{h \in H}(h(B) = h(B'))$$

Definition 4 (Collision Rate). *Let* H *be a cosine hash family over* $\mathbb{R}^m$ *and* ρ *a real number such that* $-1 \leq \rho \leq 1$*. Let* A *and* B *be two vectors in* $\mathbb{R}^m$ *such that* $\cos(A, B) = \rho$*. Then the* collision rate *of* H *for* ρ*, designated as* $C_H(\rho)$*, is defined as follows:*

$$C_H(\rho) = \mathbf{Pr}_{h \in H}(h(A) = h(B))$$

2.1 False positive rate and hash length

In order to compare two hash functions, it is not enough to compare their collision rate for the case where the distance between the two vectors is low, i.e. the true positive rate. We also need to compare the collision rate when the two vectors are distant from each other, i.e. the false positive rate. In the case of the hash functions designed for the cosine measure, the false positive rate is simple to define: it is the collision probability for the case when the two vectors are uncorrelated, i.e. when the cosine is zero.

Definition 5 (False positive rate). *For a family* H *of cosine hash functions over* $\mathbb{R}^m$*, the false positive rate is the collision rate at* $\rho = 0$*, i.e.* $C_H(0)$*.*

It is worth noting that for all the hash families considered in this paper, the false positive rate corresponds to the collision rate had we chosen the hashes randomly from the universe $\mathbf{U}$. In fact, we set the false positive rate by choosing the size of $\mathbf{U}$. The bigger the size of $\mathbf{U}$, the 'stronger' the hash, and it improves the precision of the hash. At the same time, increasing the size of $\mathbf{U}$ reduces the collision rate for values of ρ close to one, i.e. it reduces the recall rate. The right choice for the size of U depends on the hash family, and on the application. We will have more to say on this in the experimental section.

Instead of using $|\mathbf{U}|$, we use $\log_2(|\mathbf{U}|)$ to characterize the strength of the hash family, and we call it the *length* of the hash family. The rationale is simple: $\lceil \log_2(|\mathbf{U}|) \rceil$ is the minimum number of bits required to encode the output of the hash function. It seems easier to visualize "an eight bit hash function" than "a hash function with an output universe of size 256".

[1] When the cosine measure is interpreted as correlation coefficient, $cos(A, B) = 1$ indicates that the two vectors are identical up to a scaling factor, $cos(A, B) = 0$ indicates that the two vectors are uncorrelated and negative values of $cos(A, B)$ refers to negative correlation.

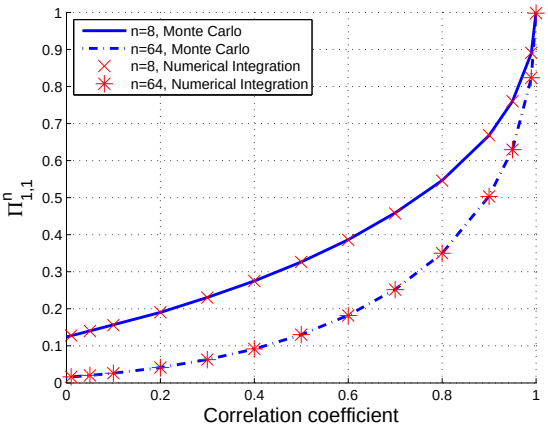

Figure 1: Plot giving the dependence between $\Pi^n_{1,1}$ and correlation coefficient for different sample sizes n.

3 Random Hyperplane Hash Family

In this section, we introduce the basic cosine hash function called the random hyperplane class [2]. The random hyperplane hash algorithm works as follows: for a desired hash length of l, generate an $m \times l$ matrix M of real numbers, where each element is chosen independently and at random from a $\mathcal{N}(0,1)$ distribution. The hash of a vector $A \in \mathbb{R}^m$ is computed in two steps,

- Compute the vector $P = AM$
- The hash of A is the l-bit integer whose ith bit is 0 if $P_i < 0$, and 1 otherwise.

The l-bit random hyperplane hash family denoted as Ψ_l is the set of hash functions corresponding to all possible values of M. Using the results in [2], it is easy to show that Ψ_l is a cosine hash family, and that its collision rate is $C_{\Psi_l}(\rho) = (1 - \frac{\arccos(\rho)}{\pi})^l$. The false positive rate, i.e. $C_{\Psi_l}(0)$, is $(\frac{1}{2})^l$ and the hash length is l.

The problem with the random hyperplane hash is that for moderate values of l, for example $l = 12$, the collision rate is quite small even for values of ρ close to one. For example, if $l = 12$ and $\rho = 0.9$, the collision rate is 0.1557. In other words, with a 12 bit hash only 15.5% of the vectors with 0.9 similarity will have the same hash. Our hash algorithm with the same false positive rate setting, as introduced in the next section, has a collision rate of 0.333 for $\rho = 0.9$, i.e. more than twice random hyperplane's collision rate.

4 Theory of Concomitants

We introduce a new class of hash functions based on the concomitant rank order statistics of bivariate normal distributions. The basic set up from which we proceed is as follows: Let $(X_1, Y_1), (X_2, Y_2) \ldots (X_n, Y_n)$ be n independent samples of a bivariate normal distribution with correlation ψ. In [5], David et al. call Y_k as the *concomitant* of X_k. Let X_k be the smallest of $X_1, X_2, \ldots X_n$. What is the probability that Y_k is the smallest of $Y_1, Y_2, \ldots, Y_n$? In other words, what is the probability that the concomitant of the smallest of X_i is the smallest of Y_i? In [5], the authors use $\Pi^n_{1,1}$ to denote this probability. More generally, they use $\Pi^n_{r,s}$ to denote the probability that the concomitant of the rth smallest of X_i is the sth smallest of Y_i.

The above question is addressed by the theory of concomitants [5], which arises naturally in order statistics. Consider a medical testing or an interviewing scenario, where two tests, a cheap one and a

more reliable expensive test needs to be conducted on a large set of subjects in a cost effective way. The common practise is to administer the cheap test on all the subjects and rank them based on the results. A smaller subset is constructed from the top of the rank set and the expensive test is now conducted on the subset. The success of the methodology lies in the statistical dependency between the results of the two tests.

The link between concomitant order statistics and cosine hash families is provided later through Lemma 1, where we show that $\Pi^n_{1,1}$ is the collision rate for our concomitant based hash family. But before we get there, we need to further explore $\Pi^n_{1,1}$, and in particular show that for a given n, it is a monotonically increasing function of ψ.

In [5], it is shown that

$$\Pi^n_{1,1} = n\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left(\int_{x}^{\infty}\int_{y}^{\infty} f(x,y)\right)^{n-1} f(x,y)\,dx\,dy \tag{2}$$

where $f(x,y)$ is the bivariate distribution function for X, Y. Of course, we are interested in the case where the distribution is bivariate normal, thus

$$f(x,y) = \frac{1}{2\pi(1-\psi^2)^{\frac{1}{2}}}\exp\left\{-\frac{1}{2(1-\psi^2)}\left[x^2 - 2\psi xy + y^2\right]\right\},$$

Theorem 1. *$\Pi^n_{1,1}$ is a monotonically increasing function with respect to the correlation coefficient ψ.*

Proof. Using the bivariate normal assumption[2], considering $\psi \geq 0$ and, substituting $c = \frac{\psi}{\sqrt{1-\psi^2}}$ and $y = (u + cx)\sqrt{1-\psi^2}$ in Eqn. 2, we get:

$$\Pi^n_{1,1} = n\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(\lambda)^{n-1}\varphi(x)\varphi(u)\,dx\,du \tag{3}$$

where, $\lambda := \int_0^{\infty}\varphi(x+\theta)(1-\phi(u-c\theta))\,d\theta$. φ and ϕ represent the standard normal probability density function and the cumulative density function respectively. Differentiating Eqn. 3,

$$\frac{\partial\Pi^n_{1,1}}{\partial\psi} = n\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\frac{\partial\lambda}{\partial\psi}\underbrace{(n-1)(\lambda)^{n-2}\varphi(x)\varphi(u)}_{>=0}\,dx\,du \tag{4}$$

$$\frac{\partial\lambda}{\partial\psi} = -\int_0^{\infty}\varphi(x+\theta)\frac{\partial}{\partial\psi}\left(\int_{-\infty}^{u-c\theta}\frac{1}{\sqrt{2*\pi}}\exp(-\frac{\gamma^2}{2})d\gamma\right)d\theta$$

Using Leibniz integral rule,

$$\frac{\partial\lambda}{\partial\psi} = \int_0^{\infty} c\varphi(x+\theta)\varphi(u-c\theta)d\theta \geq 0$$

Every term under the integral in Eqn. 4 is positive and a similar result can be proved for $\psi < 0$.

While there is no closed form solution for $\Pi^n_{1,1}$, we can use the simplified form shown in Eqn. 3 to apply numerical integration techniques such as Laguerre Gauss quadrature [16] and Hermite Gauss [16] to closely approximate it. Fig. 1 illustrates the monotonic relation between $\Pi^n_{1,1}$ and the correlation coefficient ψ for different values of n where the rank distribution was computed using two methods based on numerical integration and Monte Carlo simulations. Another useful property of the rank distribution under the bivariate normal assumption is a symmetry relation given by $\Pi^n_{r,s} = \Pi^n_{n+1-r,n+1-s}$, which we exploit a bit later in this work in Sec. 6.

[2]We would like to point out an errata in [5] in their reduced form of the rank distribution under the bivariate normal assumption. The correct form follows in this work.

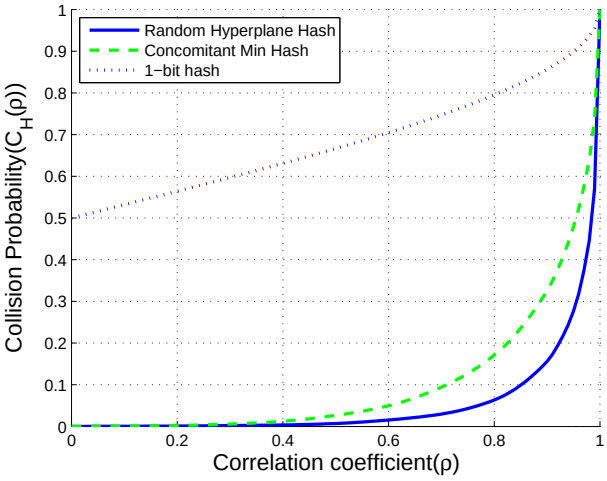

Figure 2: Collision rate comparison for a false positive rate of 2^{-12}

4.1 Concomitant Hash Family

The main contribution of this work is to use the above theory of concomitants to develop a hash function for cosine similarity. The link is provided by the following key lemma:

Lemma 1. *Let $A \in \mathbb{R}^m$ and $B \in \mathbb{R}^m$ be two real vectors, where $\cos(A, B) = \rho$. Let $M \in \mathbb{R}^{m \times n}$ be a matrix of real numbers, each element of which is drawn independently and at random from a standard normal distribution, $\mathcal{N}(0, 1)$. Let $P = AM$ and $Q = BM$. Then $(P_1, Q_1), (P_2, Q_2), \ldots, (P_n, Q_n)$ are i.i.d. samples from a bivariate normal distribution with correlation coefficient ρ.*

The proof of this lemma follows straightforwardly from the basic results of random projection mentioned in [15][9]. Based on this lemma, here is the most basic of our hash function for $\mathbb{R}^m$:

Hash Family 1 (Concomitant min hash algorithm). *The algorithm is as follows: generate a random matrix $M \in R^{m \times n}$ in which each element is generated independently from a standard normal distribution $\mathcal{N}(0, 1)$. The concomitant min hash of a vector $A \in \mathbb{R}^m$ is computed in two steps:*

- *Compute the vector $P = AM$*
- *The hash of A is given by k where k is the index of the smallest element of P.*

The choice of n determines the hash length l, which is given by $l = \log_2(n)$, as $\log_2(n)$ bits are required to encode a number between 1 and n. The l-bit concomitant Min hash family, designated as Ω_l, is the family of concomitant hash functions corresponding to all possible M.

Using the concomitant rank distribution result from [5], Thm. 1 and Lemma. 1, it is easy to prove that Ω_l is a cosine hash family, and that the collision rate is $\Pi^n_{1,1}$ corresponding to $\psi = \rho$. Furthermore, from Definition. 5, the false positive rate can be proved to be $\frac{1}{n}$. For the 1-bit case, it can be shown that the concomitant min hash algorithm and the random hyperplane hash algorithm are identical.

4.2 Comparison with the random hyperplane hash family

Fig. 2 shows the collision rate for the 12-bit random hyperplane hash family(Ψ_{12}) and the 12-bit concomitant min hash family(Ω_{12}). For visual clarity, the collision rate plots throughout this paper focus on

cosine similarity ranging from 0 to 1. The collision rate for the concomitant min hash family was obtained through Monte Carlo simulations. It is clear that the collision rate for high values of ρ, is significantly higher for the Ω_{12} family than the Ψ_{12} family. Fig. 2 also includes the performance of both families for the 1-bit hash case for which their collision rates are identical.

5 Multi-Hash Families

A common shortcoming of LSH schemes is that the collision probabilities rapidly decreases with decreasing similarity, so that even for relatively similar items the collision probability is quite low. The remedy has been to use multiple independent LSH functions to construct several hash tables in order to improve the collision probability. There is no advantage to be gained by joining the hash tables, since the output of two different hash functions matching does not have any significance in terms of the similarity. Alternatively, sampling techniques [11] have been used in a query setting to overcome the above issue.

An appealing property of our concomitant based hash algorithm is that it can be naturally extended to generate a set of integers as the hash, rather than just one integer. We call the set of integers created by the hash function for a given item the *multi-hash* of the item. We define two multi-hashes to collide if they are not disjoint.

The basic idea is as follows: as before, choose a projection matrix M of i.i.d. instances of the $\mathcal{N}(0,1)$ distribution. To hash a vector A, compute $P = AM$. Let the smallest element of P be P_{s_1}, the second smallest P_{s_2} etc. Then the k-multi-hash of A is the set $\{s_1, s_2, \ldots, s_k\}$.

Multi-hashes can then be used for information retrieval in the following way: for each vector in the data base, we compute the k-multi-hash of the vector and add all the elements of this multi-hash to the hash table. To do similarity based retrieval, given the query vector Q, compute the k-multi-hash of Q and then query hash table with all the elements of the multi-hash. Any of the k hashes that returns a match is considered a positive hit.

Clearly, as we represent a vector by more than one hash, the false positive rate goes up. In order to match the false positive rate of the single hash case, we increase the size of $\mathbf{U}$. The result is a much improved true positive rate with the same false positive rate as before. The cost, of course, is that we have to store more hashes per item in the hash table.

We use $\binom{\mathbf{U}}{k}$ to refer to the set $\{s : s \subset \mathbf{U} \wedge |s| = k\}$ i.e., the set of all subsets of $\mathbf{U}$ with cardinality k.

Definition 6 (Cosine Multi-Hash). *For a set $\mathbf{U} \subset \mathbb{N}$, the set*

$$H = \{h_1, h_2, \ldots\}$$

is a cosine k-multi-hash family over $\mathbb{R}^m$ iff each h_k is a function satisfying

$$h_k : \mathbb{R}^m \rightarrow \binom{\mathbf{U}}{k}$$

and

- *For any four vectors A, A', B, B' in $\mathbb{R}^m$ such that $\cos(A, A') = \cos(B, B')$,*

$$\mathbf{Pr}_{h \in H}(h(A) \cap h(A') \neq \emptyset) = \mathbf{Pr}_{h \in H}(h(B) \cap h(B') \neq \emptyset)$$

- *For any four vectors A, A', B, B' in $\mathbb{R}^m$ such that $\cos(A, A') > \cos(B, B')$,*

$$\mathbf{Pr}_{h \in H}(h(A) \cap h(A') \neq \emptyset) > \mathbf{Pr}_{h \in H}(h(B) \cap h(B') \neq \emptyset)$$

The collision rate *of H designated as $C_H(\rho)$, is defined as follows:*

$$C_H(\rho) = \mathbf{Pr}_{h\in H}(h(A) \cap h(A') \neq \emptyset).$$

where $\cos(A, A') = \rho$. *As before,* false positive rate *corresponds to* $C_H(0)$.

Definition 7 (Concomitant Min k-Multi-Hash). *Let* $\mathbf{U} = \{1..n\}$. *Let M be an $m \times n$ matrix of real numbers each element of which is generated independently from a standard $(0, 1)$ normal distribution. Then the k-multi-hash of the vector A is computed in two steps:*

- *Compute the vector $P = AM$*
- *Compute the set $\{i_1, i_2, \ldots i_k\}$ where i_1 is the index of the smallest element of P, i_2 is the index of the second smallest element of P etc. This set is the k-multi-hash of A*

The family Ω_n^k of hash functions over $\mathbb{R}^m$ is defined as the set of all such hash functions for all choices of M.

Using the concomitant rank statistic theory, it is possible to prove that Ω_n^k is a cosine multi-hash family. We omit the proof in this paper.

5.1 Collision Rate of the concomitant Min k-Multi-Hash family

Though in general the collision rate for this class of hash functions does not have a closed form solution, in the case of $\rho = 0$ there is a closed form solution.

Theorem 2. *The false positive rate for the Ω_n^k family is given by*

$$1 - \frac{((n-k)!)^2}{(n-2k)!n!}$$

For $k = 2$ this reduces to $\frac{4}{n-1} - \frac{6}{n^2-n}$. To facilitate the comparison with the other hash functions, we have used the approximation for the false positive rate to be $\frac{4}{n}$.

We used Monte Carlo simulation to compute the value of $C_{\Omega_n^k}(\rho)$ for various values of k, n and ρ. The graph in Fig. 3 shows the collision rate for the three hash families. For the random hyperplane and single concomitant hash, we chose the hash length to be 12. For the k-multi-hash case, we chose $k = 2$ and $n = 2^{14}$, thus ensuring the same false positive rate for all three hashes, namely 2^{-12}. As can be seen, the improvement in collision rate for higher values of ρ is spectacular; for example, for $\rho = 0.9$, the collision rate for the 2-concomitant-hash is 0.5854, where the collision rate for the single concomitant hash is 0.3278 and for random hyperplane is 0.1556. Thus for this value of ρ, the collision rate for the 2-concomitant hash is more than 3 times the collision rate for the random hyperplane hash. Of course the collision rate for $\rho = 0$ is the same for all three hashes, i.e. 2^{-12}.

6 Cascaded Hash Families

When LSH schemes are used in real similarity based retrieval applications, there is a tension between the hash length, the precision of the retrieval operation, and the recall rate. Increasing the hash length improves precision, but hurts recall. For every application and hash algorithm, there is an optimum value for the hash length that optimizes the precision/recall ratio.

While with the random hyperplane algorithm the computational cost of the hash algorithm is linear with the hash length, with the concomitant hash algorithms the computational cost is exponential with the hash length. Thus it is impractical to increase the hash length beyond a certain limit. A hash length of 10

needs 1024 vector multiplications per hash, which is tolerable, but a hash length of 20 requires $1,048,576$ vector multiplications, which while possible, is clearly too high. The solution is to use cascading. Before we go any further, let's define what we mean by cascading.

Definition 8 (Cascading). *The cascade of two l bit integers a and b, written as $cascade(a, b)$, is the $2l$ bit integer $2^l a + b$. The cascade of two sets of l bit integers A and B is the set $\{cascade(x, y) : x \in A, y \in B\}$.*

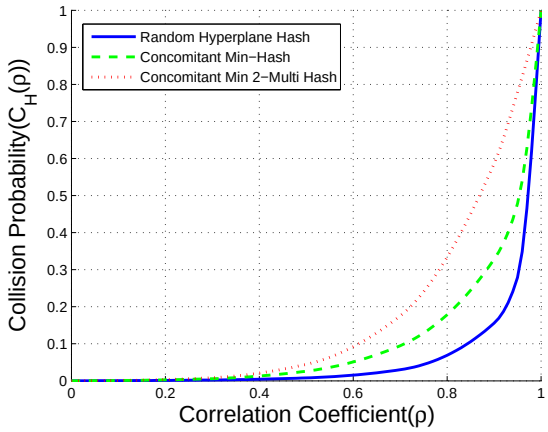

Figure 3: Collision rate comparison for a false positive rate of 2^{-12}

So, to generate a 20 bit cascaded concomitant hash, we generate two 10 bit hashes from two independent hash functions, and cascade them. While the collision rate is not going to be quite so good as the 20 bit concomitant hash, it is still much better than the 20 bit random hyperplane hash. Thus instead of spending 2^{20} vector multiplications for generating the hash, we perform 2^{10} vector multiplications twice.

But we can do even better: instead of doing two independent concomitant hashes, we can exploit a symmetry property of the concomitant rank order statistics, $\Pi^n_{1,1} = \Pi^n_{n,n}$ and that $\Pi^n_{1,1}$ is independent of $\Pi^n_{n,n}$. So we cascade the max and min indices to construct the cascaded concomitant min & max hash.

Hash Family 2 (Cascaded Concomitant Min & Max Hash Algorithm). *The algorithm is as follows: generate a random matrix $M \in R^{m \times n}$ in which each element is generated independently from a standard normal distribution $\mathcal{N}(0, 1)$. The hash of a vector $A \in \mathbb{R}^m$ is computed in two steps:*

- *Compute the vector $P = AM$*
- *The hash of A is given by $cascade(j, k)$ where j is the index of the smallest element of P and k is the index of the largest element of P.*

A combination of the k-multi-hash family with the min & max strategy results in the Concomitant k^2 min & max multi-hash family.

Hash Family 3 (Concomitant k^2 Min & Max Multi-Hash). *The algorithm is as follows: generate a random matrix $M \in R^{m \times n}$ in which each element is generated independently from a standard normal distribution $\mathcal{N}(0, 1)$. The hash of a vector $A \in \mathbb{R}^m$ is computed in two steps:*

- *Compute the vector $P = AM$*
- *The hash of A is given by $cascade(\{a_1, a_2, \ldots a_k\}, \{b_1, b_2, \ldots b_k\})$ where, $k \leq \lfloor \frac{n}{2} \rfloor$, a_1 is the index of the smallest element of P, a_2 is the index of the second smallest element of P, etc. and b_1 is the index of the largest element of P, b_2 is the index of the second largest element of P, etc.*

Theorem 3. *The false positive rate for the $\Omega_n^{k^2}$ family is given by*

$$\left(1 - \frac{((n-k)!)^2}{(n-2k)!n!}\right)^2$$

For $k = 2$, we approximate the false positive rate to be $\frac{16}{n^2}$.

6.1 Empirical Comparison

Fig. 4 shows the collision rate for the concomitant 2^2 min & max hash obtained with $n = 2^8$. The hash length for the multi-hash family is set based on Thm. 3 in order to ensure the same false positive rate as the 12-bit random hyperplane hash family. We note that the collision rate for the 2^2 min & max multi-hash case is similar to that of the min 2-multi-hash at a significantly smaller value of n, 2^8 versus 2^{14} for concomitant min 2-multi-hash, which leads to a reduction in computational cost. However, the reduction in computational effort is at the cost of more storage, as we store 2^2, 16-bit hashes compared to 2, 14 bit hashes in the concomitant min 2-multi-hash case.

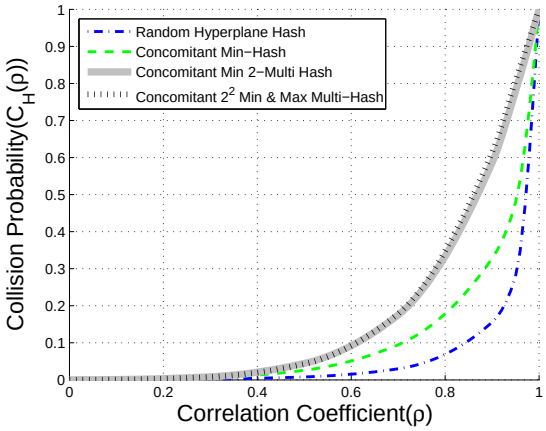

Figure 4: Collision rate comparison for a false positive rate of 2^{-12}

The collision rate improvement of our concomitant based schemes over random hyperplane hashing can be visualized in another interesting way. For the sake of this experiment, we fix the collision rate curve for the random hyperplane hash scheme and vary the hash length of the 2^2 min & max multi-hash scheme. This experiment serves as a relaxation of the strict definition of the false positive rate, which corresponds to the collision rate at zero cosine similarity. Depending on the nature of the application, we may require a lower collision rate than the random hyperplane scheme below a certain similarity threshold and a higher collision rate beyond the same threshold. Fig. 5, illustrates such desirable behavior obtained by modifying the hash length for the concomitant 2^2 min & max multi-hash algorithm.

7 Using Discrete Transforms For Concomitant Hashing

The main computational cost of the concomitant hash family described above is the cost of performing the matrix multiplication AM, which involves $m \times n$ floating point multiplications and additions for generating $\log(n)$ bits. We found that when $m <= n$, we can reduce this cost to $n \log_2(n)$ operations by using row permutations of orthogonal arrays used in discrete transforms. In this section, we will describe the concomitant hash based on the Discrete Cosine Transform (DCT). We also tested the idea with the Fast Hadamard Transform and it works just as well.

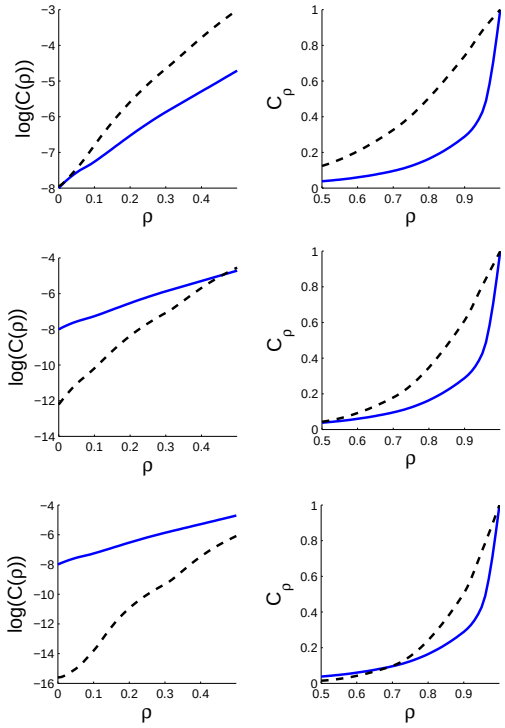

Figure 5: Comparison of collision rate curves(left($0 \leq \rho \leq 0.5$), right($0.5 \leq \rho \leq 1$)) for varying hash lengths of the 2^2 min & max multi-hash family(black, dashed) and $n = 8$ Random hyper plane scheme(blue). The curves on the left compare the collision rate performance in the log scale for better illustration. The curves from top to bottom correspond to hash lengths $6, 8, 10$ for the 2^2 min & max multi-hash scheme.

Hash Family 4 (DCT Hash). *Let M be a row permutation of the $n \times n$ Discrete Cosine Transform matrix, and let $l = \log_2(n)$. The DCT min hash of a vector $A \in \mathbb{R}^n$ is computed in two steps:*

- *Compute the vector $P = AM$*
- *The hash of A is given by k where k is the index of the smallest element of P.*

The l-bit DCT min hash family is the family of hash functions corresponding to all possible M, i.e. all row permutations of the DCT matrix.

Of course, the whole point of using the DCT hash is that the multiplication AM can be performed using the discrete cosine transform, which takes $nlog(n)$ steps rather than n^2 steps. To take advantage of the DCT, the multiplication matrix must be the original DCT matrix, and not a permutation of it. We overcome this problem by performing the random permutation on the input vector and leaving the matrix M to be the DCT matrix.

Another point to make regarding the DCT hash is that it requires a square matrix, which means that the length of the input vector A has to be the same as $n = 2^l$, where l is the desired hash length. But what if the length of the original vector is less than n? Of course we could just append zeroes to the original vector, and this would preserve the cosine, but we found that if the length of the original vector is a small fraction of n, appending zeroes will seriously impact the performance of the hash function, because it will effectively remove the majority of the rows of M from consideration. Instead, we transform the original vector, of length m, into another vector of length n by another randomization step such that

cosine similarity between vectors is preserved, and the majority of the entries in the resulting vector are non-zero. Here is the algorithm we use for achieving this. In this algorithm, B is the original input vector (of length m), A is the input vector to the DCT hash (of length n) and $s = \lfloor n/m \rfloor$.

- Generate a vector R of length s by randomly sampling the $(0, 1)$ normal distribution s times, and then subtracting the average of the samples from each sample. Thus R is zero centered. R is created once and for all, and used for all vectors.
- Create A as follows:

  For $1 \leq i \leq m$, $1 \leq j \leq s$, $A_{mj+i} = B_i R_j$.

  For $ms < i \leq n$, $A_i = 0$.

It is easy to prove that for any two vectors B, B' of length m and the two corresponding vectors A, A' generated by this algorithm, $cos(B, B') = cos(A, A')$. Thus we can use the A vectors for generating the hashes.

Multi-hash and cascading versions of the DCT hash can be created using exactly the same procedure as we use for the concomitant hash; we omit the description of these for brevity.

Empirical results show that the collision rate for the various versions of the DCT hash are almost identical to the collision rate for the corresponding concomitant hash. Thus in practice we can use the DCT hash for all applications where the size of the input vector is smaller than 2^l. We have not been able to establish the theoretical properties of DCT hashes in a rigorous way, but for various reasons we feel it has to do with the orthogonality of the columns of the DCT matrix. For example, when we remove the majority of the rows of the matrix (by zero padding the input vector), orthogonality no longer applies, and the quality of the hash deteriorates. It is not surprising that we could not develop any rigorous theoretical understanding of the DCT hash. As observed in [9], theoretical understanding of the statistics of P obtained from projection matrices other than the one obtained from an i.i.d. Gaussian distribution is an open problem.

8 Image Similarity: A practical application

In this section, we demonstrate the practical applicability and the good performance of our hashing algorithm by applying it to identify similar images[3] in a image repository.

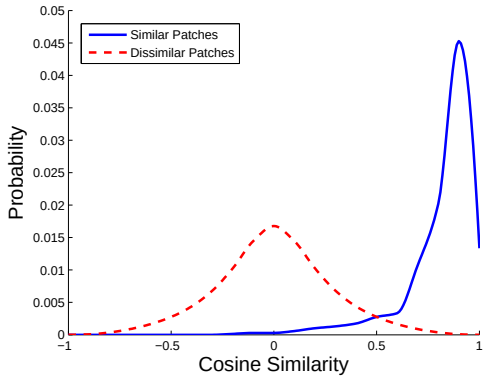

Figure 6: Distribution of Cosine Similarity for similar and dissimilar image descriptors

[3]Our notion of image similarity involves variations due to translation, scale, rotation, focus levels, illumination, foreground occlusion, and adjacent view points and transformations such as cropping, enhancement, resizing.

Successful image similarity algorithms involve two main steps. The first being a key point detection algorithm such as SIFT [10], which identifies a representative set of keypoints for every image along with a local descriptor vector for each keypoint. In our experiments, we used the PCA-SIFT descriptor used in [8]. The keypoint extraction step is followed by a matching algorithm, which computes the matching keypoint descriptors between pairs of images. The biggest problem that arises in such an approach is one of scalability. Consider an image database with 500 images and let each image be represented by 200 keypoints each. A straightforward matching algorithm has to perform $250 \times 499 \times 200 \times 200$ descriptor comparison operations to identify all the matches. Clearly, such an approach would not scale for larger databases. However, locality sensitive hashing algorithms(LSH) can be used in such a setting for performing efficient matching and it has been used for image similarity applications [8, 6, 3].

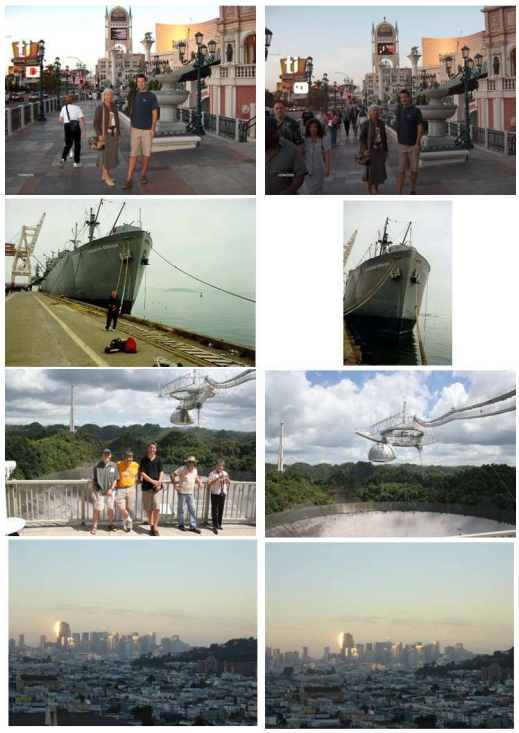

Figure 7: A representative image set capturing variations between similar images.

Fig. 6 illustrates the applicability of the cosine measure for image similarity using a distribution of the cosine similarity between matching descriptor pairs belonging to similar images and random descriptor pairs from dissimilar ones. It is important to note that for cosine similarity to work well for image descriptors, they must be zero centered.

We employ cosine hash family functions namely, random hyperplane hashing and the concomitant hashing scheme for performing the matching operation. The matching algorithm is as follows:
Step 1. For every image, extract 200 SIFT keypoints[4] and obtain the zero-centered 36 dimensional PCA-SIFT descriptor for each keypoint.
Step 2. Obtain a single(multi) hash for each descriptor and store the hash with its corresponding image id in a list

[4]We order the keypoints based on decreasing scale and pick the top 200 keypoints

Step 3. Sort the list based on the hash
Step 4. Perform a linear scan of the sorted list to identify matching hashes from different images. Create an image-image map, identifying matching image pairs with their corresponding matching hash count
Step 5. Image pairs with matching count $>$ a threshold T are labeled as similar image pairs.

Traditionally, image similarity algorithms are evaluated using synthetic datasets in which a set of images are processed using various image transforms to generate a set of duplicate images. In this work, we adopt a more practical approach and our experiments were performed on a personal image collection of 500 images. Fig. 7 shows a set of representative images from our dataset. We are restricted to a 500 image dataset because of our elaborate evaluation scheme, which we describe next.

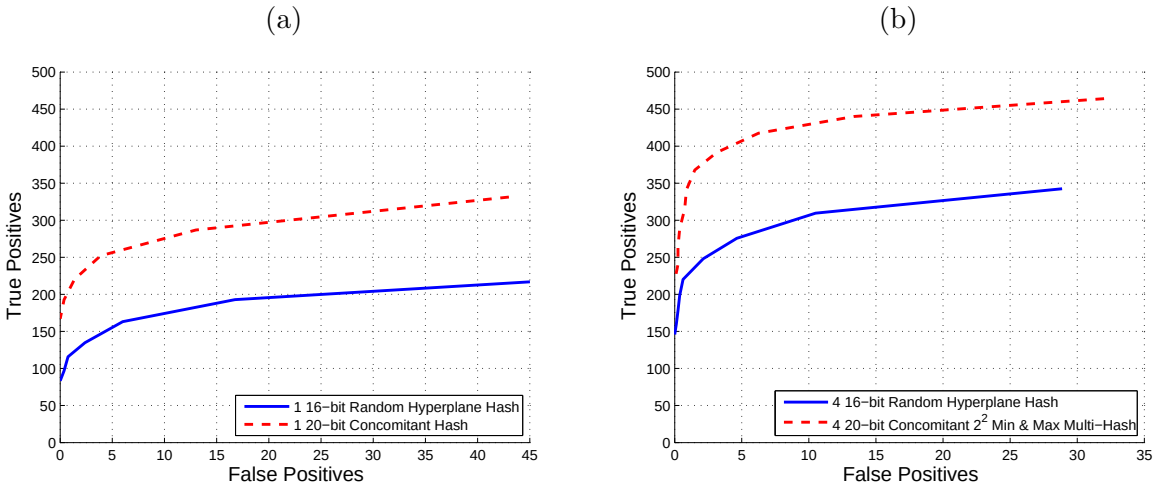

Figure 8: True positive vs. False positive curves for (left) single hash cosine hash families and (right) multi hash cosine hash families

Evaluation through progressive labeling: The matching algorithm elaborated above is used to obtain a list of the top 500 image pairs sorted based on the matching hash count. For step 2 of the matching algorithm, we experiment with four hashing schemes namely, the random hyperplane hash (single and multi-hash), concomitant hash (single and k^2 min & max multi-hash), DCT based hash and Hadamard transform based hash. The hash lengths for each algorithm were identified using cross validation on a small labeled set. Further, we ran the algorithm 10 times for each setting using different random matrices. Evaluating the different algorithms involves labeling their top 500 image pairs as similar/dissimilar. Towards achieving such an objective, we use a progressive labeling scheme in which we manually label all the image pairs list in progressive fashion, i.e., an image pair once labeled for a particular setting, need not be labeled again. Such an approach clearly limits the total labeling effort. The threshold T mentioned in step 5 is varied to obtain the true positive and false positive count for each T. These counts are then averaged over the different runs of a setting to obtain averaged true positive vs. false positive curves. Since, we do not perform an exhaustive labeling of all the 500×250 image pairs, we present the results in terms of counts rather than rates.

First, we present the comparative performance of single hash cosine hash family algorithms in Fig. 8(a). It shows the significant improvement of the concomitant-min-hash in identifying true positives for the same false positive count. Fig. 8(b) presents a similar improvement in the multi hash case where 4 hashes are used to represent each descriptor. Further, notice the multi-hash schemes outperform the single hash schemes in a big way. As illustrated earlier through Fig. 5, the hash lengths of our scheme can be manipulated so as to provide higher collision rate performance than the random hyperplane hashing beyond a certain similarity threshold while ensuring a lower collision probability below the same threshold. This effect is

reflected in the different hash lengths (16 for the random hyperplane hash and 20 for the concomitant hash) at which the two schemes exhibit their best performance.

Fig. 9 illustrates the transform based approximations of the concomitant hash algorithm, which is crucial in ensuring applicability in practical situations. The DCT based approximation follows the curves of the concomitant 2^2 min & max multi-hash algorithm very closely and the more efficient Hadamard based approximation curves are slightly worse. This effect is expected as the Hadamard matrix is made up of 1s and -1s whereas, the DCT matrix is made up of real numbers between -1 and 1 leading to better Gaussianity in the DCT case.

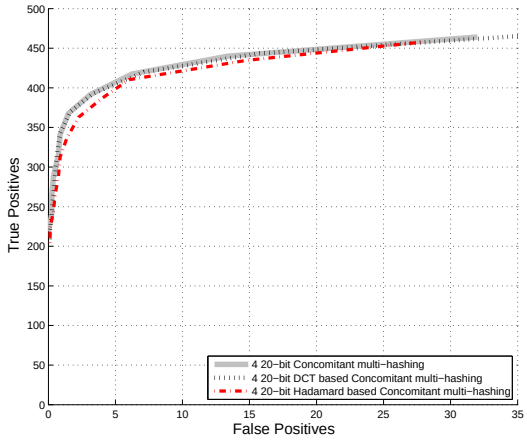

Figure 9: Comparison of the performance of multi hash concomitant with its transform based approximations

9 Conclusion

In this paper, we introduce a novel family of locality sensitive hash functions for the cosine measure based on the theory of concomitant order statistics. We address the rapid collision rate decreasing issue of the state of the art random hyperplane method and show the better collision rate curves of our scheme. We illustrate the real impact of the collision rate improvement through the excellent performance of our algorithm in identifying similar images.

10 Acknowledgements

We would like to thank Ira Cohen, Jaap Suermondt, Lyle Ramshaw and Charlie Dagli for several interesting discussions regarding this work. We thank Jaap Suermondt for sharing his personal picture collection for this work.

References

[1] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *FOCS '06: Proceedings of the 47th Annual IEEE Symp. on Foundations of Computer Science*, pages 459–468, Washington, DC, USA.

[2] Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, NY, USA.

[3] Ondřej Chum, James Philbin, Michael Isard, and Andrew Zisserman. Scalable near identical image and shot detection. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 549–556, NY, USA.

[4] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, NY, USA.

[5] H. A. David, M. J. O'Connell, and S. S. Yang. Distribution and expected value of the rank of a concomitant of an order statistic. *The Annals of Statistics*, 5(1):216–223, January 1977.

[6] K. Grauman and T. Darrell. The pyramid match kernel: Efficient learning with sets of features. *Journal of Machine Learning Research*, April 2007.

[7] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proc. of 30th STOC*, pages 604–613, 1998.

[8] Yan Ke, Rahul Sukthankar, and Larry Huston. An efficient parts-based near-duplicate and sub-image retrieval system. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 869–876, NY, USA, 2004.

[9] Ping Li, Trevor J. Hastie, and Kenneth W. Church. Very sparse random projections. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 287–296, NY, USA, 2006.

[10] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal in Computer Vision*, 60(2):91–110, 2004.

[11] Rina Panigrahy. Entropy based nearest neighbor search in high dimensions. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1186–1195, NY, USA, 2006.

[12] Deepak Ravichandran, Patrick Pantel, and Eduard Hovy. Randomized algorithms and nlp: using locality sensitive hash function for high speed noun clustering. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 622–629, NJ, USA, 2005.

[13] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[14] K. Singh, M. Ma, and D. W. Park. A content-based image retrieval using fft & cosine similarity coefficient. In *Signal and Image Processing*, 2003.

[15] Santosh Vempala. *The Random Projection Method*. American Mathematical Society, Providence, RI, 2004.

[16] Daniel Zwillinger. *CRC Standard Mathematical Tables and Formulae, 31st Edition*. CRC Press, 2003.