Efficient Nearest Neighbors via Robust Sparse Hashing

Anoop Cherian Suvrit Sra Vassilios Morellas Nikolaos Papanikolopoulos

Abstract—**This work presents a new Nearest Neighbor (NN) retrieval framework: *Robust Sparse Hashing* (RSH). Our approach is inspired by the success of dictionary learning for sparse coding. Our key idea is to sparse code the data using a learned dictionary, and then to generate hash codes out of these sparse codes for accurate and fast NN retrieval. But direct application of sparse coding to NN retrieval poses a technical difficulty: when data are noisy or uncertain (which is the case with most real world datasets), for a query point, an exact match of the hash code generated from the sparse code seldom happens, thereby breaking the NN retrieval. Borrowing ideas from robust optimization theory, we circumvent this difficulty via our novel robust dictionary learning and sparse coding framework called RSH, by learning dictionaries on the robustified counterparts of the perturbed data points. The algorithm is applied to NN retrieval on both simulated and real world data. Our results demonstrate that RSH holds significant promise for efficient NN retrieval against the state of the art.**

Index Terms—**Dictionary Learning, Sparse Coding, Robust Optimization, Nearest Neighbors.**

I. Introduction

Finding data points nearest to a query point is an important operation that arises in many areas of computer science including computer vision and pattern recognition. It is well-known that as data dimensionality increases, finding the Nearest Neighbors (NN) becomes computationally expensive [16]. This is a short coming for NN retrieval on real-world data descriptors, which are often high dimensional (e.g. Scale Invariant Feature Transform (SIFT) descriptors (128D) [25], Generalized image descriptors (960D) [36], etc.).

Faster NN retrieval schemes are valuable. Traditionally, many approaches have been advocated to solve the NN problem in large dimensions. One approach is to search for Approximate Nearest Neighbors (ANN) so that the recovered neighbor is in an ϵ-neighborhood of the true solution. A more efficient way to approach this problem is hashing, in which similar data points are mapped to compact codes called hash codes, with the condition that similar data points will have similar hash codes. This is the primary theme of Locality Sensitive Hashing (LSH) [13]. In this paper, we propose a novel hashing algorithm based on the concepts of Sparse Coding (SC) and Dictionary Learning (DL). The main idea of SC is to represent each data vector as a sparse linear combination of basis vectors from an overcomplete dictionary learned using DL. Our NN algorithm builds a hash code over this sparse code using the indices of the active basis in the learned dictionary.

Traditionally, for each input vector, SC algorithms select a set of dictionary atoms that minimize the reconstruction error between the original data point and its sparsity based approximation. Since these algorithms are not concerned with the active set of dictionary elements for a given data vector, it often happens that two data vectors that are close to each other in the input space might have different atoms involved in their sparse approximations. Since our hash coding scheme requires that similar data points have the same basis activations, such a scheme might not help us in robust hash code generation or NN retrieval. This mismatch in the basis activations can arise due to two reasons, namely (i) the vectors are noise corrupted versions of each other, whereby some of the dictionary atoms might reconstruct the noise; and (ii) the true vectors themselves are perturbed or uncertain.

Existing DL methods such as [11], [15], [35], that address noise corruption, *impose* a noise model on the data, with the hope that filtering out the noise and sparse coding the denoised data points might produce similar hash codes for similar data points. Unfortunately, such an idea assumes that the noise distribution is *known*. This is seldom true for real-world data. We study the second scenario instead, and explicitly model data uncertainty from a robust optimization perspective. The main advantage of our approach is that rather than assuming any noise distribution, we estimate the maximum perturbation that the data points may undergo. This estimation is based on a supervised learning scheme. Such worst case perturbations are then used to immunize the data points against any further perturbations; a sparse coding scheme over these immunized data descriptors produces robust hash codes. Our experiments show that such a scheme can generalize and subsume a diverse set of noise distributions using a single perturbation model, which might otherwise have required investigations into disparate noise models when using the denoising approach.

Before we describe the details of our approach, let us enlist the main contributions of this paper.

- **Formulation:** We present a new Robust Sparse Hashing (RSH) framework for learning robust sparse codes, by showing a practical extension to the traditional DL setup.
- **Algorithm:** Our RSH formulation leads to a difficult non-

A. Cherian is with the LEAR project team at INRIA Rhone-Alpes, 655 Avenue de l'Europe, 38330 Montbonnot, France. His email id is anoop.cherian@inria.fr.

S. Sra is with the Max Planck Institute for Intelligent Systems, Spemannstrasse 38, 72076 Tübingen, Germany. His email id is suvrit.sra@tubingen.mpg.de

V. Morellas and N. Papanikolopoulos are with the Department of Computer Science and Engineering, University of Minnesota, MN-55455, USA. Their email ids are {morellas, npapas}@cs.umn.edu respectively.

convex optimization problem. We investigate properties of this optimization problem, and propose a novel algorithm for its efficient local solution.

- **Application:** We illustrate the utility of RSH by applying it to the task of NN retrieval on benchmark datasets and compare it against state-of-the-art algorithms.

II. Related Work

a) Nearest Neighbors: Low dimensional data has several data structure based schemes for efficient ANN retrieval, such as k-d trees, R-trees, etc.([12], [22]). Unfortunately, when the data dimensionality increases moderately (say beyond 20), the efficiency of such schemes starts dropping as shown in [5]. It is seen that in high dimensional spaces, the computational expense of these classical algorithms is no better than a linear scan over the entire dataset [16]. Of the several schemes advocated for efficient ANN, the most popular has been *Locality Sensitive Hashing* (LSH) [13], which introduces the traditional hashing schemes into a probabilistic framework in which similar data points have a greater probability of being assigned to the same hash bucket. It is often seen that the hash family selected for LSH do not have any connection to the structural properties of the original data, whose knowledge might help in generating smaller hash codes. Towards this end, Restricted Boltzmann Machines (RBMs) [33] and adaptations of Boosting [34] have been proposed. These algorithms need to store the original data points in the hash table to resolve collisions; hindering their scalability. When there is a choice of multiple alternatives to solve a problem, it is often difficult to choose the best algorithm. Automating this selection process for ANN algorithms has been tried in [27], which suggests a combination of two popular ANN techniques: (i) hierarchical K-means and (ii) randomized k-d trees. This hybrid algorithm is well-known as Fast Large scale ANN (FLANN). Similar to the above approaches, this algorithm also suffers from issues of scalability when higher retrieval accuracy is desired, due to the need for storing higher number of cluster centroids.

Recent efforts in ANN retrieval have been towards prioritizing for lower memory footprint, while relaxing retrieval accuracy, and resulted in the design of efficient hashing schemes that capture the data properties. An effort in this direction is the well-known Spectral Hashing (SH) algorithm ([38]) that embeds the data in a Hamming space in such a way that the data neighborhood in the original space is preserved. To make the resultant optimization objective tractable, SH makes the assumption that the data is uniformly distributed. This assumption is avoided in [24] by estimating a low dimensional data distribution. Learning such low dimensional structures has also been adopted in other hashing algorithms such as [18]. There have also been efforts to leverage standard machine learning techniques to address ANN via LSH. For example, in [23], random linear data projections are extended to non-linear functions through a kernel mapping. A general trend in all these methods is the selection of a set of random hyperplanes to which the data is projected, later these projections are used for hashing in the binary Hamming space. It is often seen that the selection of these hyperplanes has a tremendous impact on the accuracy of the respective algorithm. More recently, quantization techniques have been extended to produce efficient indexing techniques in [1], [19], in which a data vector is split into multiple disjoint sub-vectors, each such sub-vector quantized independently using centroids learned via k-means. There could be redundancies in the centroids used for quantizing sub-vectors, using which could help improve ANN retrieval.

Building similarity metrics on sparse codes has been investigated several times in the recent past ([32], [9]). Unfortunately, the adequacy of these metrics for retrieval problems is not thoroughly investigated. Sparse coding for image classification has been suggested in [37], [39]. A sparse coding framework that approximates the inner-product distance between data points for retrieval is presented in [40]. All these papers agree that sparse coding is sensitive to variations in the original data vectors; as a result, applying these approaches to retrieval problems is difficult.

b) Robustness and Dictionary Learning: Robust optimization (RO) [4] is a well-established branch of optimization that has lately been gaining importance in machine learning [8]. The normal DL problem has been viewed from certain robust perspectives previously, but these are different from the formal notions of robustness propounded by RO. For example, in [2], the image sparse coding problem is made robust to image rotations and scaling through learning a dictionary in the log-polar space. Dictionary learning in the presence of Gaussian noise is suggested in [15], while other noise models are considered in [35]. A fundamental limitation of these approaches are the assumptions that they impose on the noise model; often such assumptions can be infeasible or even incorrect. In [29], a robust formulation of the dictionary learning problem is suggested in the context of outlier detection by using the Lorentzian norm instead of the L1 norm. This paper also proposes constraints on the dictionary atoms for better incoherence and thereby improve the robustness of the reconstruction. Another method using LSH via dictionary learning method is presented in [11], where the noise is modeled as a convolution of Gaussian and Laplacian distributions, followed by a denoising approach to generate robust hash codes. The density of this convolution is difficult to characterize analytically leading to approximate inference schemes.

In contrast to the existing methods, we ground RSH using the notion of robustness from RO, wherein the solution sought must be invariant to certain bounded uncertainty in the data. To our knowledge, the problem of learning a dictionary based on an RO formalism and its application to NN retrieval are novel.

III. Background

A. Dictionary Learning and Sparse Coding

Before we describe our formulation, let us recall the standard ideas of dictionary learning and sparse coding. Learning overcomplete dictionaries has long been a key problem in sparse approximation [15], [31]. Compressed sensing formalisms suggest that for a vector that is dense (in some basis), there exists an incoherent overcomplete basis over which the vector has a sparse representation [7]; e.g., a vector encoding a signal that is dense in the time domain, will have a sparse representation in the frequency domain. Learning an overcomplete basis over which the input vectors have sparse representations is roughly the goal of DL.

A commonly used formulation of DL is the following. Let $\mathcal{T} = \{v_1, \cdots, v_m\}$ ($v_i \in \mathbb{R}^d$) be input (training) vectors. DL seeks a dictionary $\mathcal{D} \in \mathbb{R}^{d\times n}$ that is overcomplete (i.e., $n \gg d$), so that each $v_i \approx \mathcal{D}c_i$ for some *sparse* vector c_i. A natural optimization problem for learning $\mathcal{D}$ and c_i is then,

$$\begin{aligned} \min_{\mathcal{D},c_1,\ldots,c_m} \quad & \sum\nolimits_{i=1}^{m} \bigl(\frac{1}{2}\|v_i - \mathcal{D}c_i\|_2^2 + \beta\|c_i\|_1\bigr) \\ \text{s.t.} \quad & \|b_j\|_2 \le 1, \text{ for } j = 1, 2, \ldots, n, \end{aligned} \tag{1}$$

where $\beta > 0$ is a sparsity tuning parameter (or regularization) while the constraint $\|b_j\|_2 \le 1$ normalizes each column b_j of $\mathcal{D}$ to prevent degeneracy. Problem (1) is computationally difficult due to its non-convexity, so we can at best hope for locally optimal solutions.

B. Dictionary Learning for NN retrieval

Intuitively, the motivation for the use of DL for NN comes from the fact that there is a large set of basis vectors of which only a very few are active. Suppose we use a dictionary with n basis vectors, and if only k of them are required to reconstruct a given data vector to a reasonable level of accuracy, then overall $s = \binom{n}{k}$ unique active basis combinations are possible. Assuming the active set is uniformly distributed across all the basis vectors (that is, every basis in the dictionary is equally-likely to be used in an active set), a corpus of m data vectors can be mapped into s hash bins, such that each bin will have on an average m/s data vectors. For typical choices of n and k, there is a high probability that each data vector gets a unique active set. To give the reader an idea of the large set of basis combinations possible using this approach, we will consider a typical case when using SIFT descriptors as the dataset. Suppose we use $n = 1024$ basis dictionary and assume that $k = 10$ basis vectors are active for hashing, then there are approximately 3×10^{23} unique basis combinations possible. This is a large number and on an average there is a high likelihood that only a few data points from the dataset will share the same active set. This leads to unique hashing and subsequently faster NN retrieval using a hash table.

Mathematically, for an arbitrary $v \in \mathcal{T}$, let $J(v) := \{j_1, \ldots, j_k\}$ be the corresponding set of indices such that $c_{j_l} \neq 0$ for $1 \le l \le k$ and $v \approx \mathcal{D}c$. The set $J(v)$ may be viewed as a hash-code for v. Indeed, we can build a data structure that stores $\mathcal{D}$ along with the vectors $\mathcal{T}$ hashed according to $\{J(v_1), \ldots, J(v_m)\}$. Now, suppose a new (test or query) point v' is presented. We first obtain the corresponding sparse representation $v' \approx \mathcal{D}c'$ by solving

$$c' = \operatorname{argmin}_c \quad \frac{1}{2}\|v' - \mathcal{D}c\|_2^2 + \beta\|c\|_1, \tag{2}$$

to obtain the hash key $J(v')$, which we can rapidly look up in the hashtable. The nearest data points are retrieved by scanning through (linearly or via a further data structure) the points associated with the key $J(v')$. The idea is illustrated in Figure 1.

A theoretical investigation into the connection between sparse coding and NN retrieval for inner product distances is established in [40] and for Euclidean distances in [10]. Since we assume the latter, we will briefly review the main idea proposed in [10]. This paper suggests that the Euclidean distances between data points is upper-bounded by the Euclidean distance between their sparse representations. The idea is captured in the following theorem.

Theorem 1 (Distances): Let $v_1, v_2 \in \mathbb{R}^d$ be two zero-mean data points and let $\mathcal{D}$ be the dictionary. Further, let c_1 and c_2 be the sparse coefficient vectors of v_1 and v_2 respectively, produced as per the formulation (2) so that $\|v_i - \mathcal{D}c_i\|_2^2 \le \delta^2$, for $i \in \{1, 2\}$, where δ is a constant and is inversely related to β (it is actually the regularization parameter when writing (2) in the traditional LASSO form [14]). Then, we have:

$$\|v_1 - v_2\|_2 \le 2\delta + \|\mathcal{D}\|_2\|c_1 - c_2\|_2. \tag{3}$$

Proof: See [10]. ■

Further, the paper [10] also shows that when the data vectors are closer to each other in radial angles (assuming the data vectors are centralized to have zero-mean), they are more likely to share the same set of basis elements from the dictionary when they are sparse coded. This idea, when combined with Theorem 1, provides a powerful NN retrieval framework for locality sensitive hashing based on angles (and a subsequent linear search using Euclidean distances on the sparse coefficients). Unfortunately, the method in [10] needs sparse coding with mutliple regularization constants; a deficiency that we will fix in the current paper.

Although the above NN scheme we described seems attractive, it could be brittle. Neither while building $\mathcal{D}$ nor while testing a query point, do we really enforce any constraints that $J(v_1)$ should be similar to $J(v_2)$ whenever v_1 and v_2 are similar (or strongly correlated) vectors in the original input space. In symbols, suppose $v_1 \approx \mathcal{D}c$ and $v_2 = (1 + \gamma)v_2$, for some small perturbation $\gamma \neq 0$. For facilitating NN retrieval, we desire $J(v_1) = J(v_2)$; but since $\mathcal{D}$ is overcomplete there is *no guarantee* that this happens. This problem is illustrated in Figure 2, and overcoming it is precisely the goal of RSH.

C. Robust Optimization

One practical way to avoid difficulties in dealing with data uncertainties is to resort to robust optimization [3]; the main idea is to explicitly model the worst case data uncertainty. Traditionally, in the robust optimization literature, methods that assume probability distributions on the perturbations fall under the so called *chance constrained* problems. In the context of robustifying the data point, we can generically write such a problem in the following form: given a data point v, we seek an associated sparse code c such that:

$$\min_{c,t} \{t : Prob_{\theta\sim\mathcal{P}}\left(\phi(c; \theta, v) \le t\right) \ge 1 - \epsilon\}, \tag{4}$$

where ϕ is an appropriate loss function, θ encapsulates the parameters of the model including the dictionary $\mathcal{D}$, $\mathcal{P}$ represents the distribution of the perturbations (such as the Gaussian

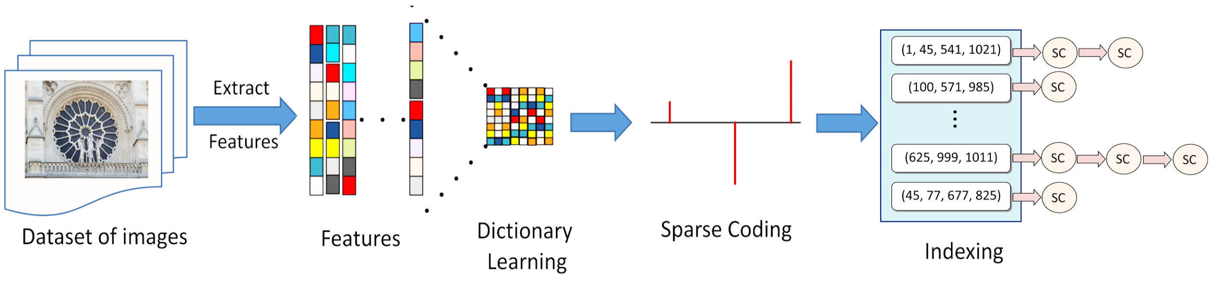


Fig. 1. An illustration of the various steps involved in NN retrieval via sparse coding. An overcomplete dictionary is first learned from the data. Next, each data vector is sparse coded using this learned dictionary. The indices in the dictionary for each active basis in the sparse code is then used to construct a tuple based hash code for each data vector. This tuple hash code is later used for hashing. Data vectors having the same hash code are placed in a linked list associated with the corresponding hash bucket.

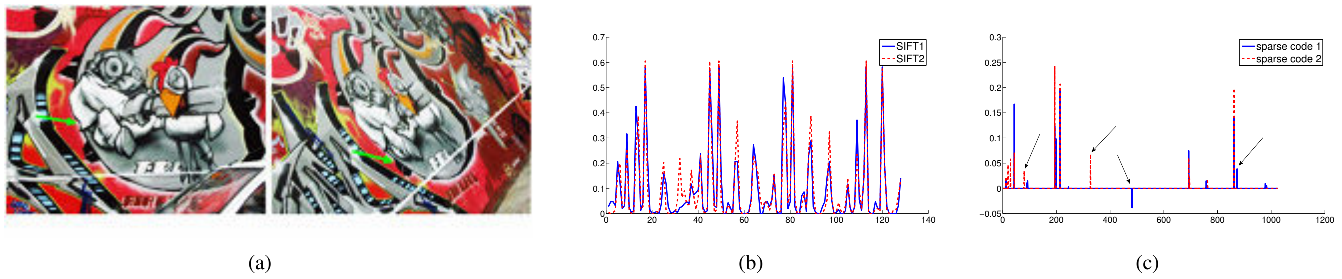


Fig. 2. (a) shows (with green arrows) two SIFT descriptors corresponding to the same key point, (b) plots the two SIFT descriptors in Figure 2(a) as a discrete time signal in 128 dimensions, and (c) shows the sparse representation of the descriptors using a 1024 dimensional learned dictionary. The arrows in (c) show dimensions where the non-zero indices mismatch.

and Laplacian presented in [11]), and $\epsilon \approx 0$ is a tolerance. The semi-colon notation separates the unknown parameters on its left from the known parameters on its right. When the distribution of $\mathcal{P}$ is unknown, we have an *ambiguous chance constrained* problem [3], which takes the following variant of (4):

$$\min_{c,t} \{t : Prob_{\theta\sim\mathcal{P}}\left(\phi(c;\theta,v) \le t\right) \ge 1-\epsilon, \forall\mathcal{P}\in\mathcal{U}\}, \quad (5)$$

where now the distribution $\mathcal{P}$ is not well-defined, but what is known is that it belongs to a set $\mathcal{U}$ of all valid distributions. Such a scheme provides a rich setup for inference, but unfortunately, care must be taken to make sure that the robust model remains tractable. In the rest of this paper, we will look at the robust counterparts of chance constrained problems for our case of sparse coding. Our approach is basically to model the allowed uncertainties that a given data point might undergo, subsequently incorporating the worst case uncertainty in our sparse coding formulation so that our hash codes are immune to these perturbations.

IV. Robust Sparse Hashing

Let there be a set of nominal data vectors $\bar{\mathcal{T}} = \{\bar{v}_1, \dots, \bar{v}_m\}$ that is sparse in an appropriate dictionary $\mathcal{D}$. Let $\mathcal{U}(\bar{v})$ model *uncertainty* for the point $\bar{v}$, i.e., it is some set that models perturbations in $\bar{v}$. To be robust against the perturbations, RSH seeks a dictionary so that all points $v \in \mathcal{U}(\bar{v})$ have the same hash codes: $J(v) = J(\bar{v})$. Consider therefore, the *robust* counterpart of (1):

$$\begin{aligned} \min_{\mathcal{D},c_1,\dots,c_m} \quad & \sum_{i=1}^{m}\left(\frac{1}{2}\|\mathcal{S}(v_i)-\mathcal{D}c_i\|_2^2+\beta\|c_i\|_1\right) \\ \text{s.t.} \quad & \|b_j\|_2 \le 1, \text{ for } j=1,2,\dots,n \\ & v_i \in \mathcal{U}(\bar{v}_i), \text{ for } i=1,2,\dots,m. \end{aligned} \quad (6)$$

where the notation $\mathcal{S}(v)$ stands for the robustified data point v. Based on [4], a few choices of $\mathcal{U}$ are:

- *Norm-ball:* $\mathcal{U}(\bar{v}) := \{v : \|v-\bar{v}\|_p \le \alpha\}$ for $p \ge 1$
- *Ellipsoidal:* $\mathcal{U}(\bar{v}) := \{v : v = \bar{v} + Pu, \ \|u\|_2 \le 1, \ P \succ 0\}$,

where the notation $P \succ 0$ stands for P is positive definite. A natural choice for the norm-ball constraints is the L_2 ball, which assumes the data perturbations are uniform in all directions. Such an assumption is seldom true in practice and we found it not to produce any relevant retrieval results in our experiments. Other choices of the norm-ball (such as the L_1 or L_∞ balls) result in objectives that are difficult to optimize. Thus, we prefer the ellipsoidal uncertainty model in this paper, which models non-uniform perturbations by ellipsoids parameterized by a symmetric positive definite matrix P. Since the parameter P of the ellipsoids is learned from the data (as we will detail in Section V-B, this model provides a tractable trade-off between the various models. Also, the model shows better empirical performance as well. Using the ellipsoidal uncertainty leads to the following two stage min-max type version of (1):

$$\min_{\mathcal{D},C} \sum_{i=1}^{m} \left(\frac{1}{2}\|\bar{v}_i + Pu_i^* - \mathcal{D}c_i\|_2^2 + \beta\|c_i\|_1\right)$$
$$\text{where} \quad u_i^* := \max_{\|u\|_2\le 1} \frac{1}{2}\|\bar{v}_i + Pu\|_2^2 \tag{7}$$
$$\text{and } \|b_j\|_2 \le 1, \text{ for } j = 1, 2, \ldots, d,$$

where $C = \{c_1, \ldots, c_m\}$. The formulation (7) is the core RSH problem of this paper.

V. Algorithms for RSH

The basic difference between the traditional dictionary learning problem (1) and the robust formulation (7) is in computing u^* which is the direction of the worst case perturbation. Since u is dependent only on the data points v, we can solve for it independently of $\mathcal{D}$ and C. Once u is computed for each $\bar{v}$, (7) reduces to (1) where the data points are now the robustified vectors $\hat{v} = v + Pu^*$. To this end, the primary optimization problem that we need to tackle centers in efficiently solving for u; avoiding the subscripts for simplicity, and replacing maximization by minimization, the formulation in (7) on u^* can be rewritten as:

$$f(u) := \min_{\|u\|_2\le 1} \quad -\frac{1}{2}u^T P^T P u - u^T P v - \frac{1}{2}v^T v, \tag{8}$$

At first sight (8) might look difficult to optimize, because the quadratic term is $-P^T P$, which is negative-definite instead of positive-definite. That is, (8) is a non-convex quadratic program. Fortunately, because there is just one constraint, (8) still remains tractable. In fact, for this particular type of non-convex problems, strong-duality holds thanks to the so-called S-lemma [see e.g.[6]]. S-lemma says that for a quadratically constrained quadratic program for which the objective is non-convex, but the constraint is convex, the Lagrangian of this objective function will satisfy *strong duality*. Recall that the Lagrangian is always concave irrespective of whether the primal objective is convex or not. Further, strong duality implies that a solution to the Lagrangian will imply an exact solution to the primal objective as well, with a zero duality gap between the primal-dual problems.

Problem (8) is even more special: it is a trust-region subproblem, although a non-trivial one because of the negative-definite matrix $-P^T P$. Though, we can use a generic trust-region solver such as the LSTRS (Large Scale Trust Region Subproblem) method [30] for this problem, we show below that this problem can be solved more efficiently.

A. Efficient Implementation via Newton Descent

Using a multiplier γ, the Lagrangian of (8) can be written as:

$$L(\gamma, u) := -\frac{1}{2}u^T P^T P u - u^T P v - \frac{1}{2}v^T v + \gamma\left(u^T u - 1\right). \tag{9}$$

Setting $\frac{\partial L}{\partial u} = 0$, we have:

$$u = \left(P^T P - 2\gamma\mathcal{I}\right)^{-1} P v, \tag{10}$$

where $\mathcal{I}$ is the $d \times d$ identity matrix. Let $P^T P = U\Sigma U^T$ for an orthonormal U and diagonal Σ. Now, substituting for $P^T P$ in (10) using this singular value decomposition, and applying the unit norm constraint on u, we have the following root finding problem in the scalar γ:

$$v^T U^T \Sigma \left(\Sigma - 2\gamma\mathcal{I}\right)^{-2} \Sigma U v = 1.$$

Using the substitution $q = \Sigma U v$ and using the Hadamard product $\hat{q} = (q \circ q)$ then leads to a simple vector equation:

$$\sum_{i=1}^{d} \frac{\hat{q}_i}{(\sigma_i - 2\gamma)^2} = 1. \tag{11}$$

where $\hat{q}_i$ is the ith component of $\hat{q}$ and σ_i is the ith eigenvalue of Σ. As is clear, (11) is convex in γ and can be efficiently solved by the *Newton-Raphson* method for a suitable initialization. For a reasonably well-conditioned matrix $P^T P$, (in our implementation, we initialize the iterations with $\gamma = -\max(diag(\Sigma)^2)$), on an average, 3–6 times speedup[1] was observed against the LSTRS method. Figure 3 shows the average speedup produced by our algorithm compared to the LSTRS method for an increasing dimensionality of the P matrix and simulated data.

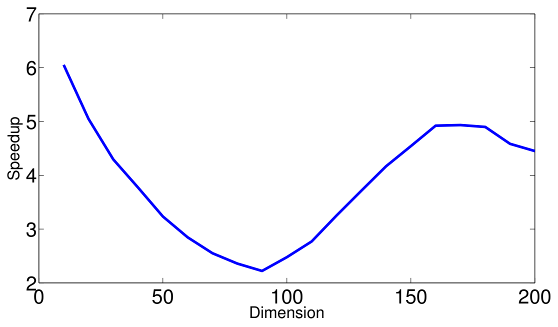


Fig. 3. A plot of the speedup produced by the Newton Descent solution of RSH problem against the LSTRS solver [30]. The x-axis is the dimensionality of the data uncertainty matrix.

B. Computing the Uncertainty Matrix

The parameter P of the uncertainty ellipsoid plays a crucial role in quantifying uncertainty and thus care must be taken in learning this parameter. Towards this end, we propose a supervised approach by setting P to the *Löwner-Jones* uncertainty ellipsoid computed on matching data pairs. That is; suppose we have a training set of matching data pairs $\{(v_i, v_i'), i = 1, 2, \cdots, N\}$, where we assume $v_i = \bar{v}$ is the nominal data point and $v_i' = \bar{v} + \epsilon_i$ for some noise ϵ_i. We compute the difference $\epsilon_i = v_i - v_i', i = 1, 2, \cdots, N$. An uncertainty elliposid can be defined as an ellipsoid of *minimum volume* enclosing all the ϵ_i's. If P represents the parameters of this ellipsoid and if ϵ_c is its centroid, then the following optimization problem ensues to find P;

$$\min_{P\succ 0, c} -\log det(P)$$
$$\text{s. t. } (\epsilon_i - \epsilon_c)^T P (\epsilon_i - \epsilon_c) \le 1, \tag{12}$$

[1]We define speedup as the ratio of the time taken by LSTRS method to solve the objective on u^* to the time taken by our algorithm for the same input data vectors.

for $i = 1, \cdots, N$. The objective (12) can be solved via the well-known *Khachiyan* algorithm [21], which is a polynomially bounded linear program that generates a sequence of ellipsoids whose volume shrinks in subsequent iterations, and approximates to the parameters of the ellipsoid that will enclose the data points. Since we work with zero-mean data, ϵ_c is generally seen to be close to the origin and thus can be neglected.

VI. Experiments and Results

In this section, experiments are presented to evaluate the effectiveness of RSH. First, we will define the metrics against which performance is measured and later compare RSH against the state-of-the-art methods.

A. Performance Metrics

1) Basis Overlap: Assume two test data vectors v_1 and v_2 and a learned basis dictionary $\mathcal{D}$. Let a_1 and a_2 represent the set of active indices of the sparse codes for each of the data points respectively. Let c_1 and c_2 represent the set of coefficient values corresponding to each index in a_1 and a_2 respectively. If T (> 0) is a threshold, then we define *Basis Overlap* (BO) as:

$$BO(v_1, v_2) = \frac{\#(a_1^{|c_1|>T} \cap a_2^{|c2|>T})}{\max(\#a_1^{|c1|>T}, \#a_2^{|c2|>T})}, \qquad (13)$$

where the notation $\#a^{|c|>T}$ means: the number of indices in a whose corresponding coefficient values c are greater than a threshold T in absolute value. Intuitively, *BO* describes the proportion of the overlap of the indices in the sparse representation of v_1 and v_2. Ideally, when the data points are similar, we expect BO to be unity which we call a *Perfect Basis Overlap*. We note in passing that the standard metric for evaluating set intersections is the *Jaccard distance* [17], which has a similar form as (13), but uses the cardinality of union of the two sets in the denominator. In this paper, we prefer BO to Jaccard distance as BO directly provides an intuition into the proportion of the longest index set that overlapped. This information is useful for deciding the threshold T and the regularization constants for sparse coding; longer sparse codes without significant BO is not useful for hashing.

2) Accuracy: When a dataset that we use does not have a ground truth associated (e.g., SIFT eight category dataset in Section VI-C1), the baselines are decided by the standard distance measure (Euclidean distance) via a linear scan. For this dataset, we create a database and a query set of q items. For each query item i, k ground truth neighbors (G_i^k) were found using the linear scan, followed by k nearest neighbors (A_i^k) retrieved using the respective algorithm. We define

$$Accuracy = \frac{1}{q} \sum_i \frac{|G_i^k \cap A_i^k|}{|G_i^k|}. \qquad (14)$$

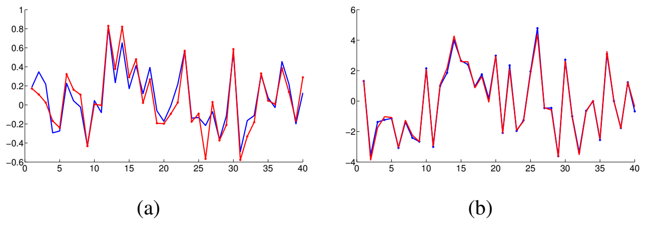
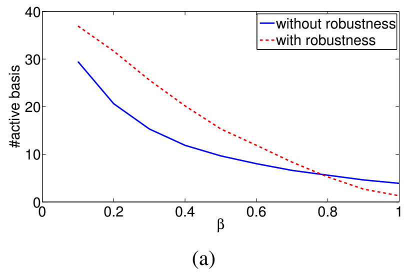


Fig. 4. Simulation results: (a) Two 40D perturbed data points (that are known to have the same active basis before adding noise) and (b) their robustified counterparts. As is clear from the plot, robust formulation brings the noisy data points belonging to the uncertainty set closer in the new mapping.



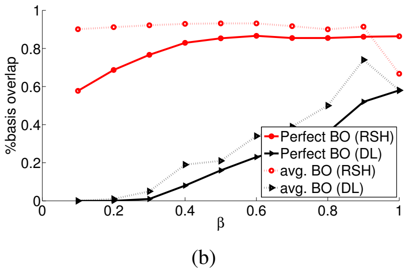


Fig. 5. Simulation results: (a) Number of active non-zero dictionary indices against an increasing regularization, (b) Average performance of NN against increasing regularization. As is clear, using the RSH formulation, we have more number of non-zero indices as well as increased basis overlap.

B. Simulations

In this section, we will first illustrate using simulated data, how incorporating the worst case uncertainty increases the basis overlap of the sparse codes. Recall that, in our robust formulation we map a given data point to a new vector $\hat{v} = v + Pu$ along the direction of worst case perturbation u. This vector $\hat{v}$ is later sparse coded via the robust dictionary $\mathcal{D}$. To evaluate our idea and understand how robustness is achieved, we created a dataset of 50K vectors each 40D using a known 40×120 dictionary (that is, first we used simulated sparse codes, which were then multiplied with the dictionary to produce dense data vectors). The data vectors were later corrupted by Gaussian and Laplacian noise to produce noisy data pairs (as suggested in [11]). Figure 4 illustrates two matching[2] data pairs (Figure 4(a)) and their robustified counterparts (Figure 4(b)). As is clear, the magnitudes of the sparse codes are now different, while their dimensions match almost exactly, showing that any sparse code created from these robustified descriptors are more likely to result in the same hash code.

[2]That is, two data points that are known to have the same basis originally, but were perturbed with noise.

We would like to highlight the importance of our robust formulation against robustness achieved via increasing the regularization constant in the LASSO formulation. Recall that regularization controls the degree of sparsity; i.e., a larger regularization means shorter and more robust sparse codes. Since the length of the sparse codes play a significant role in the uniqueness of our hashing formulation, shorter codes are not desired. We used a training set of 20K data points from the above simulated dataset to learn a dictionary using the RSH framework. Next, we applied RSH to generate sparse codes for the data pairs in the test set. Figure 5(a) shows the average active basis set size for increasing regularization, while Figure 5(b) plots the average BO between the bases in the noisy pairs, averaged on a randomly chosen 1K test vectors from the 10K test set. For the test set, we plot average BO for all the data points in the dataset, while perfect BO plots the ratio of the number of times perfect BO was achieved to the number of test data points. The plot of average BO is an upper-bound to the plot of perfect BO as is clear from the Figure 5(b). More importantly, both average BO and perfect BO are better for RSH than using just the traditional DL objective. In brief, Figures 5(a) and 5(b) together implies that RSH leads to longer hash codes, at the same time improve basis overlaps against a potential robustness achieved by increasing the regularization.

C. Real Data Experiments

The primary goal of this section is to show the robustness achieved by RSH on real data under various noise models; the idea is to show that a single robust uncertainty model (as in RSH) provides the needed robustness, that could otherwise have needed different probabilistic denoising models. To verify this claim, we used two different benchmark datasets, namely (i) the SIFT dataset[3], and (ii) the MNIST digits[4] dataset. Below, we provide details of these datasets.

1) SIFT Dataset: This dataset consists of eight image categories, each category consisting of six images of the same scene, but undergoing deformations of a particular type. The categories are as follows: (i) *BARK*, with images undergoing magnification, (ii) *BIKES*, with fogged images, (iii) *LEUVEN*, with illumination differences, (iv) *TREES*, with Gaussian blur, (v) *BOAT*, with magnification and rotation, (vi) *GRAPHIC*, with 3D camera transformation, (vii) *UBC*, with JPEG compression noise, and (viii) *WALL*, with changes in view angle. Note that these distortion categories characterize almost all the types of deformations that images in real-world undergo and the SIFT descriptors generated from these images capture the potential variabilities that one can expect in these descriptors. A point that we would like to highlight here is that each distortion category adheres to a different noise model, but we will be learning a single uncertainty model from all the categories together, which will be used for RSH. Each image in the dataset produced approximately 5K SIFT descriptors.

The ground truth was decided as follows: we used an approximate NN strategy to ensure an ϵ-neighborhood. We first computed the NNs for a given query point using linear scan. Suppose d_{ls} denotes the distance of the NN for a query point q to its closest neighbor in the database. If d_q is the distance of the closest neighbor found by an algorithm, then we say the NN is correct if $\frac{d_{ls}}{d_q} > \epsilon$ ($\epsilon = 0.9$ for all the algorithms).

2) MNIST Dataset: This dataset contains 28×28 gray scale images of digits (from 0 to 9). For tractability of our experiments, we resized the images to half their sizes. The images where then vectorized to 196D. There are 60K database images and 10K query images along with the ground truth class label for each image. From the database, we used 20K images to train the dictionary.

D. Dictionary Size Estimation

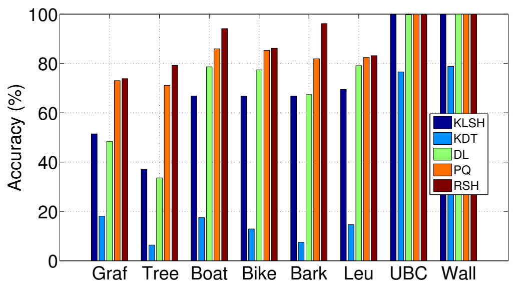
1) SIFT DL: To choose the right size of the dictionary for the NN operation, we used a cross-validation approach. We trained multiple dictionaries with number of basis varying from 256 to 4096 at steps of 256 and measured the NN performance on a small validation set of 10K SIFT descriptors. To train the dictionary, we used 5M SIFT descriptors from INRIA Holidays[5] dataset (although the uncertainty model parameters were computed on the SIFT descriptors from the eight category model). Interestingly, we found that the NN performance decreased after a dictionary size of 2048, which we speculate is due to the increase in the coherence of dictionary atoms leading to ambiguous sparse codes. Thus, we used a dictionary of size 128×2048 in this experiment. We used a regularization value $\beta = 0.2$ in RSH. On an average 7 dictionary atoms were activated in sparse coding of the robustified descriptors.

2) MNIST DL: We used a similar cross-validation approach to choose the right dictionary size for the MNIST dataset. A random sample of 1K digit images were used to query the dataset while the dictionary size was varied from 392 till 3920. Contrary to the SIFT DL, we found that there was no drop in performance when the number of basis increased, but the time for sparse coding increased many-fold, with meager improvement in accuracy. Thus, compromising on tractability of our approach and accuracy, we decided to use 1960 basis vectors in the dictionary. We used 20K data points to train the dictionary.

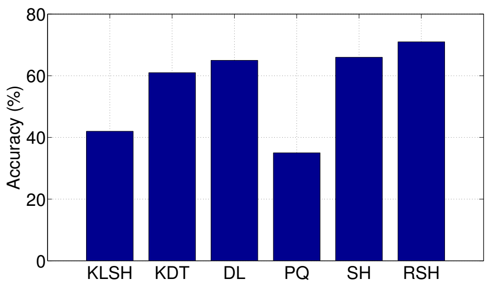
E. Uncertainty Matrix Estimation

To decide the uncertainty ellipsoid matrix, we randomly selected a set of 20K points from the SIFT eight-category dataset and computed their one-nearest neighbor using a linear scan with Euclidean distance. Later, the difference of *matching* descriptors were used to find the Löwner-Jones ellipsoid using the algorithm mentioned before. Even though this step is computationally expensive, it has to be done only once. For the MNIST digits dataset, we used 200 images per digit class to build the uncertainty matrix.

[3]http://www.robots.ox.ac.uk/~vgg/research/affine/index.html
[4]http://www.cs.nyu.edu/~roweis/data.html
[5]http://lear.inrialpes.fr/~jegou/data.php



(a) SIFT



(b) MNIST

Fig. 6. A comparison of NN performance of RSH against several other state-of-the-art NN methods on (a) SIFT eight category dataset and (b) MNIST digits dataset. We used a single data uncertainty model for all the data classes in both the datasets.

F. Nearest Neighbor Performance

In this subsection, we provide experimental results on one-nearest neighbor classification using our algorithm, along with comparison to several state-of-the-art methods in NN retrieval. As we alluded to earlier, our main goal in this section is to demonstrate the robustness achieved by our algorithm using a single worst-case uncertainty model for different data distortion categories. In the next section, we will showcase performance results on K-NN retrieval on large scale data.

We chose to compare our method against the following other state-of-the-art NN algorithms, namely (i) KDT (Best-Bin-First based on k-d trees), (ii) Kernelized LSH [23], (iii) LASSO based NN (DL), (v) Product Quantization (PQ) [19], and Spectral Hashing (SH) [38]. We used the publicly available implementations of these algorithms. For SH and KLSH, we used 64 bit hash codes, while for PQ, we used the IVFADC variant of the algorithm (using 64 bits and with single cell visits) using an inverted file system for retrieval.

In Figure 6, we show the accuracy of NN retrieval for the eight distortion category SIFT dataset and the 10 category MNIST digits dataset. The bar plots clearly demonstrate the superior performance of RSH against the state of the art. We found that adhering to Figure 5, RSH produced better BO compared to non-robust DL. This improvement over DL was more prominent in the SIFT dataset, probably because this dataset underwent more perturbations compared to the MNIST digits. Among the state of the art, the PQ method, that shows promising results on the SIFT dataset, is seen to perform poorly on the MNIST dataset.

G. Scalability of RSH

The previous experiments used relatively small datasets for performance evaluation. Contemporary datasets are often very large. In this section, we evaluate the performance of RSH for the task of retrieving K nearest neighbors using the methodology proposed in [19] using 1M SIFT descriptors from the INRIA ANN-SIFT1M dataset. [6]. To make our evaluation comparable to the results reported in [19], we decided to use a hash code of length 64-bits by learning a dictionary of 256 basis and using eight active indices for every sparse code. To learn this dictionary, we used the 100K training descriptors associated with this dataset, while we reused the SIFT uncertainty model from our experiments in Section VI-E for computing the robust directions.

For evaluation, we use the standard Recall@K metric (as in [19]), which is defined as the proportion of the query vectors for which the ground truth NN is ranked in the first K retrieved points. Note that when K=1, this measure corresponds to the standard precision measure used in papers such as [27] for ANN evaluation. We also include two other popular hashing algorithms, viz. Euclidean LSH (E2LSH) [13] and Shift Invariant Kernel Hashing (SIKH) [28] in our comparisons, using 64-bit hash codes. For the product quantization method, we used the IVFADC variant of the algorithm using single cell visits (defined as $w = 1$ in [19]), as it is the most similar variant of this technique to our method. In Figure 7, we plot the Recall@K performance on 10K SIFT test points provided in the dataset. The plot clearly shows that RSH outperforms the state-of-the-art ANN methods, especially when K is small. We achieve an improvement of 5.9% and 6.4% over the PQ-IVFADC method at Recall@1 and Recall@100 respectively. We outperform all other methods significantly when K is low.

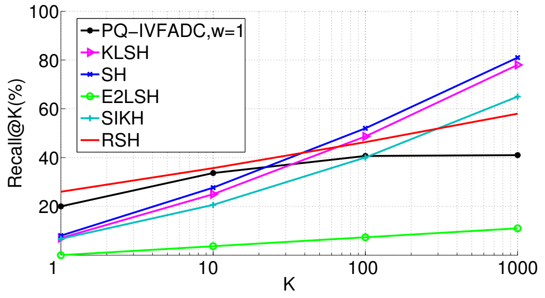
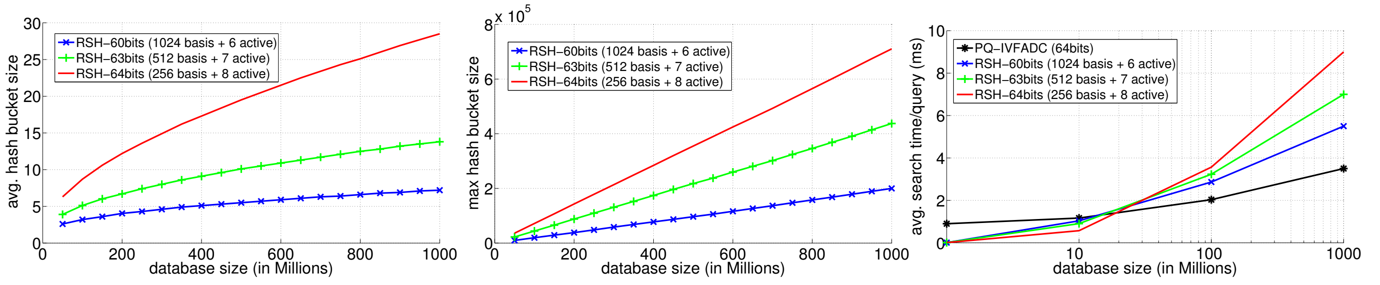


Fig. 7. SIFT dataset: comparison of K-NN recall performance against several state-of-the-art algorithms. The evaluation used 1M SIFT descriptors from the INRIA Copydays dataset. Our algorithm (RSH) uses a dictionary with 256 atoms and 8 active coefficients so that each data point is encoded using a 64-bit hash code. All other methods also use a 64-bit hash code.

H. Computational and Memory Requirements

Our algorithms were implemented mainly in MATLAB, while hash table and the query software were implemented in C++ with MATLAB interfaces. Using a single core CPU, it took on an average 9ms (in MATLAB) for computing

[6]http://corpus-texmex.irisa.fr/



(a) Avg. bucket size (b) Max bucket size (c) Query time taken

Fig. 8. Hash table statistics and average query time taken when the SIFT database size is increased to 1 billion points. We evaluate three different dictionary size/active basis combinations, namely 256/8, 512/7, and 1024/6 leading to 64-bit, 63-bit, and 60-bit hash codes.

the robust directions (u) for a SIFT descriptor and less than 100μs for sparse coding a SIFT vector using the SPAMS toolbox [26],[7]. As for the MNIST dataset, it took approximately 23ms for computing the robust directions, around 3.11ms for sparse coding. Below, we evaluate the performance of our hashing scheme and query retrieval when the database size increases to billions of points.

a) Hashtable Statistics: In this section, we explicitly evaluate the performance of hashing introduced by RSH. To this end, our evaluation criteria is the distribution of hash codes produced by RSH for an increasing number of data points. For this experiment, we used the INRIA BIGANN dataset introduced in [20] consisting of 1 billion SIFT descriptors. We decided to use a hash code of approximately 64-bits in our experiments so that our timing comparisons (discussed next) is comparable to those reported in [19]. To this end, we used three different dictionary basis/active-set combinations, namely 256/8, 512/7, and 1024/6 leading to hash codes of length 64-bits, 63-bits, and 60-bits respectively. Recall that we need to use only $\lceil \log(n) \rceil$ bits to represent the indices of a dictionary with n basis. The data points belonging to a hash bucket are stored in a linked list (possibly on the disk). Note that we need to store only the coefficients corresponding to the active basis for every descriptor, and thus our memory needs are much less compared to [27]. Also, we do not need to store the cluster centroids as in PQ [19], which are often non-sparse.

In Figures 8(a) and 8(b), we plot the average hash bucket length and the maximum number of data points in any hash bucket respectively when the hashed database size is increased from 50M to 1000M SIFT descriptors. As is expected, using a dictionary with large number of basis provides more diversity in hashing and leads to fewer hash collisions under approximately same hash code lengths. Further, the plots show that the average hash bucket size increases marginally, while the maximum number of elements increases linearly, but remains in the range of a few thousands (especially for larger dictionaries). This substantiates our assumption that sparse code combinations provide unique representations for dissimilar data points, while the recall results in Figures 6 and 7 show that similar data points get hashed to the same hash bucket.

[7]http://www.di.ens.fr/willow/SPAMS/

b) Query Retrieval Performance: In Figure 8(c), we evaluate the average query time for the three different dictionary size/active basis combinations described in the last section. We compare the performance against PQ as reported in [19] on the 1 billion SIFT dataset. As this plot shows, our performance is comparable, albeit slightly inferior when the database size is very large while using smaller dictionaries.

VII. Conclusion

In this paper, we proposed a model of the sparse coding framework for aiding NN searches. A novel formulation of dictionary learning problem was introduced based on the ellipsoidal data uncertainty, followed by a scalable and efficient algorithm to solve the resulting robust objective. Our experiments demonstrated the superior performance of our scheme over the state of the art. Going forward, we would like to investigate the effects of other perturbation models for the L1 ball, polyhedral uncertainty, etc.

Acknowledgements

This research has been supported in part by the National Science Foundation through grants #IIP-0443945, #CNS-0821474, #IIP-0934327, #CNS-1039741, #IIS-1017344, #IIP-1032018, #SMA-1028076, and #CNS-1338042. We are grateful for resources from the University of Minnesota Supercomputing Institute.

References

[1] A. Babenko and V. Lempitsky. The inverted multi-index. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3069–3076. IEEE, 2012.
[2] L. Bar and G. Sapiro. Hierarchical dictionary learning for invariant classification. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 3578–3581. IEEE, 2010.
[3] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust optimization*. Princeton University Press, 2009.
[4] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25(1):1–14, 1999.
[5] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is nearest neighbor meaningful? *Database Theory*, pages 217–235, 1999.
[6] S. P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.
[7] E. J. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006.
[8] C. Caraminis, H. Xu, and S. Mannor. *Optmization for Machine Learning*, chapter Robust Optimization in Machine Learning. MIT Press, 2011.

[9] H. Cheng, Z. Liu, L. Hou, and J. Yang. Sparsity induced similarity measure and its applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 2012.

[10] A. Cherian, V. Morellas, and N. Papanikolopoulos. Efficient Similarity Search via Sparse Coding. Technical report, University of Minnesota, 2011.

[11] A. Cherian, S. Sra, and N. Papanikolopoulos. Denoising Sparse Noise via Online Dictionary Learning. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2011.

[12] T. H. Cormen. *Introduction to algorithms*. The MIT press, 2001.

[13] M. Datar, N. Immorlica, P. Indyk, and V.S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, pages 253–262, 2004.

[14] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–451, 2004.

[15] M. Elad and M. Aharon. Image denoising via learned dictionaries and sparse representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 895–900, 2006.

[16] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. *Theory of Computing*, pages 604–613, 1998.

[17] P. Jaccard. Etude comparative de la distribution florale dans une portion des Alpes et du Jura. *Bulletin de la Socit vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[18] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *European Conference on Computer Vision*, pages 304–317, 2008.

[19] H Jegou, M Douze, and C Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

[20] H. Jégou, R. Tavenard, M. Douze, and L. Amsaleg. Searching in one billion vectors: re-rank with source coding. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 861–864, 2011.

[21] L. G. Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

[22] D. E. Knuth. *The art of computer programming. Vol. 3, Sorting and Searching*. Addison-Wesley Reading, MA, 1973.

[23] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2130–2137, 2009.

[24] W. Liu, J. Wang, S. Kumar, and S. Chang. Hashing with graphs. In *Proceedings of the International Conference on Machine Learning*, pages 1–8, 2011.

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

[26] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60, 2010.

[27] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Applications*, pages 331–340. INSTICC Press, 2009.

[28] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *Advances in Neural Information Processing Systems*, 2009.

[29] I. Ramirez, F. Lecumberry, and G. Sapiro. Universal priors for sparse modeling. In *Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, pages 197–200, 2009.

[30] M. Rojas, S. A. Santos, and D. C. Sorensen. LSTRS: Matlab software for large-scale trust-region subproblems and regularization. *ACM Transactions on Mathematical Software*, 34(2):11, 2008.

[31] S. Roth and M. J. Black. A framework for learning image priors. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[32] G. Heidemann S. Klenk. A sparse coding based similarity measure. In *International Conference on Data Mining*, pages 512–516, 2009.

[33] R. Salakhutdinov and G. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

[34] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 750–757. IEEE, 2003.

[35] X. Shu and N. Ahuja. Hybrid Compressive Sampling via a New Total Variation TVL1. *European Conference on Computer Vision*, pages 393–404, 2010.

[36] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[37] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2010.

[38] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems*, pages 1753–1760, 2009.

[39] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009.

[40] J. Zepeda, E. Kijak, and C. Guillemot. SIFT-based local image description using sparse representations. In *IEEE Conference on Multimedia Signal Processing*. IEEE, 2009.

Anoop Cherian is a Postdoctoral Researcher in the LEAR project team at INRIA Rhone-Alpes, France. He received his B.Tech (honors) degree in computer science and Engineering from the National Institute of Technology, Calicut, India in 2002, his M.S. and Ph.D. degrees in computer science from the University of Minnesota, Minneapolis in 2010 and 2013 respectively. From 2002–2007, he worked as a software design engineer at Microsoft. His research interests lie in the areas of computer vision and machine learning. He is the recipient of the Best Student Paper Award at Intl. Conf. on Image Processing in 2012.

Suvrit Sra is a Senior Research Scientist at the Max Planck Institute for Intelligent Systems in Tübingen, Germany. He received a Ph.D. in Computer Science from the University of Texas at Austin in 2007. His research focuses on large-scale data analysis and optimization. Beyond optimization, he has interests in numerous subareas within mathematics; most notably in matrix analysis. His research has won awards at several international venues; the most recent being the ”SIAM Outstanding Paper Prize (2011)” for his work on metric nearness. He regularly organizes the Neural Information Processing Systems (NIPS) workshops on ”Optimization for Machine Learning” and has recently edited a book of the same title.

Vassilios Morellas received his diploma degree in Mechanical Engineering from the National Technical University of Athens, Greece, his MSME degree from Columbia University, NY and his PhD degree from the department of Mechanical Engineering at the University of Minnesota. His research interests are in the area of geometric image processing, machine learning, robotics, and sensor integration. He is a Program Director in the department of Computer Science & Engineering and Executive Director of the NSF Center for Safety Security and Rescue. Prior to his current position, he was a Senior Principal Research Scientist at Honeywell Laboratories where he developed technologies in the general areas of access control, surveillance, and biometrics.

Nikolaos Papanikolopoulos received his Diploma of Engineering in Electrical and Computer Engineering, from the National Technical University of Athens in 1987. He received his M.S. in 1988 and PhD in 1992 in Electrical and Computer Engineering from Carnegie Mellon University. Currently, Dr. Papanikolopoulos is a Distinguished McKnight University Professor in the Department of Computer Science at the University of Minnesota and Director of the Center for Distributed Robotics and SECTTRA. His research interests include robotics, computer vision, sensors for transportation applications, and control. His transportation research has included projects involving vision-based sensing and classification of vehicles, and the human activity recognition.