Transform-based Compression for Quadratic Similarity Queries

Hanwei Wu and Markus Flierl
School of Electrical Engineering
KTH Royal Insititute of Technology
Stockholm, Sweden
Email: {hanwei, mflierl}@kth.se

***Abstract*—This paper considers the problem of compression for similarity queries [1] and discusses transform-based compression schemes. Here, the focus is on the tradeoff between the rate of the compressed data and the reliability of the answers to a given query. We consider compression schemes that do not allow false negatives when answering queries. Hence, classical compression techniques need to be modified. We propose transform-based compression schemes which decorrelate original data and regard each transform component as a distinct D-admissible system. Both compression and retrieval will be performed in the transform domain. The transform-based schemes show advantages in terms of encoding speed and the ability of handling high-dimensional correlated data. In particular, we discuss component-based and vector-based schemes. We use $P\{\text{maybe}\}$, a probability that is related to the occurrence of false positives to assess our scheme. Our experiments show that component-based schemes offer both good performance and low search complexity.**

I. Introduction

The problem of efficient data retrieval from large databases has become more relevant in recent years. Retrieval based on similarity queries is based on data items that are similar to a given query as defined by a similarity threshold. The notion of similarity is often defined by a specific metric measure, such as Euclidean distance and Hamming distance.

In case of a mobile scenario(see Fig. 1 for example), the capacity of communication between query and database side is often limited. This motivates us to construct retrieval schemes that reduce the communication between query and database side along with a proper computational load on both sides. Also, we are considering the *false negative* error type that may occur in the retrieval process and which is not permitted in some applications, such as security cameras. On the other hand, although false positive errors can be detected by further verification, they increases the computational cost, and hence, reduce the efficiency. Therefore the tradeoff between communication rate and the reliability of answers to the query is an interesting question to explore. Our setting is closely related to the problem of compression for similarity queries as introduced in [1] [2]. [2] studies the problem from an information-theoretic viewpoint and introduces the term *identification rate* of the source which characterizes the minimal compression rate that allows query answers with a vanishing false positive probability, when false negatives are not allowed. [2] [1] derives the identification rate for Gaussian sources with quadratic distortion and for binary sources with Hamming distance. For practical schemes, [3] constructs a scheme based on a Type Covering lemma (TC) using lossy compression as a building block. The results show that the compression rate is close to the fundamental limit for binary sources with Hamming similarity measure. In [4], the authors present a shape-gain quantizer for i.i.d. Gaussian sequences: scalar quantization is applied on the amplitude of data vector and the shape (the projection on the unit sphere) is quantized using a warped spherical code.

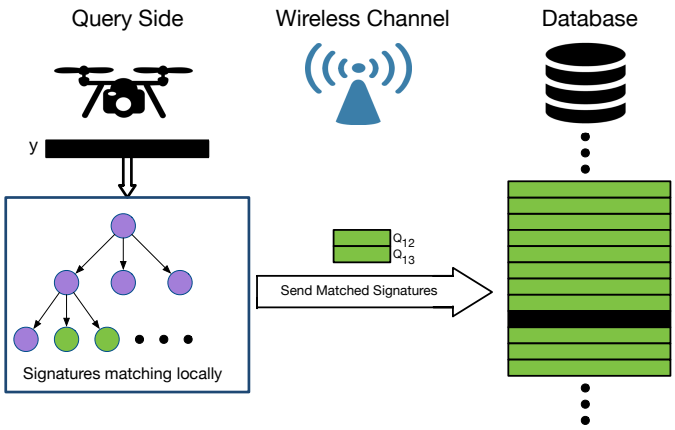

Fig. 1: Querying can be performed locally at the client. Only matched signatures are sent to the database that is stored in a different location.

In our previous work [5], we propose tree-structured vector quantizers that hierarchically cluster the data into sphere-shaped quantization cells, that the k-center clustering method is more suitable for similarity queries. However, vector quantization is inherently computational expensive. Hence, it is difficult to produce a sufficient number of centroids for a large volume of high dimensional data.

Therefore, in this paper, we propose transform-based compression schemes for similarity queries. On the database side, transform-based schemes compute a linear transformation of the data and use it to map the original data to uncorrelated coefficients. Then each coefficient is quantized separately and the indices are compressed by entropy coding. For the retrieval process, the queries will be mapped through the same linear transformation as the database and then compared with the

codebook stored in the database. The advantages of transform-based schemes are threefold. First, the transform packs energy to the first coefficients which is beneficial for bit allocation schemes. Second, the transform-based schemes avoid computing Euclidean distances between high dimensional vectors in the encoding process, and hence, require less computations compared to vector quantization. Third, the transform-based scheme can generate a large number of centroids from its uncorrelated coefficients and is less affected by the *curse of dimensionality*. These advantages are beneficial for practical applications.

The paper is organized as follows. In Section II, we give an introduction to compression for similarity queries. We include a brief description of orthonormal transforms and the preservation of Euclidean distance. In Section III and IV, we discuss two transform-based schemes, namely vector-based and component-based similarity queries. Section III discusses the decision rules of the two approaches to similarity queries in the transform domain. Section IV describes the two corresponding encoding-retrieval schemes. Simulation results are shown in Section 5.

II. Background

A. Quadratic Similarity Queries: Problem Statement

Given two n-dimensional real vectors $\mathbf{x}$ and $\mathbf{y}$, the *quadratic similarity* [1] between $\mathbf{x}$ and $\mathbf{y}$ is defined as $d(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|^2$, where $\|\cdot\|$ denotes the Euclidean norm. Consider a database, denoted by $\mathcal{F}$, which contains N independent n-dimensional real vectors $\mathbf{x}_{(i)}$, *i.e.,* $\mathcal{F} = \left\{\mathbf{x}_{(i)}\right\}_{i=1}^{N}$. The problem investigated is to design quantization schemes on the database, and algorithms which retrieve vectors from the database similar (measured in quadratic similarity) to a given query vector based on the quantized database.

We firstly describe the quantization scheme. The proposed quantization scheme $Q(\cdot)$ partitions the database $\mathcal{F}$ into K cells, denoted by $\mathcal{T} = \{C_k, k = 1, ..., K\}$, where the individual cells C_k satisfy $\cup_{k=1}^{K} C_k = \mathcal{F}$ and $C_k \cap C_{k'} = \emptyset$ for $k \neq k'$. Specifically, the quantization scheme $Q(\cdot)$ takes any vector $\mathbf{x}$ as input, and outputs the index k of the cell the vector belongs, *i.e.,* $Q(\mathbf{x}) = k$ if $\mathbf{x} \in C_k$. The index $Q(\mathbf{x})$ is called the *signature* of $\mathbf{x}$. Given a quantization cell, let $\hat{\mathbf{x}}$ be the $centroid$ of the cell, all vectors in the cell are represented by $\hat{\mathbf{x}}$.

Next, we describe the query and retrieval process. An n-dimensional real vector $\mathbf{y}$ is given as the query. For the retrieval process, the *query function* g takes the query $\mathbf{y}$ and the signature $Q(\mathbf{x})$ of each vector $\mathbf{x}$ from the database as input, and outputs the decision *no* or *maybe*. The data items $\mathbf{x}$ with output decision *maybe* should be retrieved by the scheme. The vectors $\mathbf{x}$ and $\mathbf{y}$ are called *D-similar* if $d(\mathbf{x},\mathbf{y}) \leq D$. A scheme is called D-admissible if we obtain $g(Q(\mathbf{x}),\mathbf{y}) =$ maybe for any pair of data item and query $(\mathbf{x},\mathbf{y})$ which satisfies $d(\mathbf{x},\mathbf{y}) \leq D$. In other words, a D-admissible scheme should retrieve all *D-similar* vectors in the database for a given query vector, such that no false negatives are produced. D is called the *similarity threshold*.

Consider a probabilistic model for database and query. Specifically, let $P_{\mathbf{X}}$ and $P_{\mathbf{Y}}$ denote the distribution of database vectors and query vectors respectively.The objective is to design quantization schemes that minimize the probability of the output *maybe* when averaging over database vectors $\mathbf{X}$ and query vectors $\mathbf{Y}$. According to [1], this probability is calculated as

$$\begin{aligned} P\{g(Q(\mathbf{X}),\mathbf{Y}) = \text{maybe}\} &= P\{g(Q(\mathbf{X}),\mathbf{Y}) = \text{maybe}|d(\mathbf{X},\mathbf{Y}) \leq D\}\, P\{d(\mathbf{X},\mathbf{Y}) \leq D\} \\ &\quad + P\{g(Q(\mathbf{X}),\mathbf{Y}) = \text{maybe}, d(\mathbf{X},\mathbf{Y} > D)\} \\ &= P\{d(\mathbf{X},\mathbf{Y}) \leq D\} + P(\varepsilon), \end{aligned} \tag{1}$$

where the second equality follows from $P\{g(Q(\mathbf{X}),\mathbf{Y}) = \text{maybe}|d(\mathbf{X},\mathbf{Y}) \leq D\} = 1$ by the requirement of D-admissible schemes. Hence, minimizing (1) is equivalent to minimize the probability of false positives $P(\varepsilon)$. That is, the probability $P\{g(Q(\mathbf{X})),\mathbf{Y}) = \text{maybe}\}$ can be used as a performance measure for the investigated schemes. In the following, we use the abbreviation $P\{\text{maybe}\}$.

Note that the concept of D-similarity allows us to define an expansion of a set by the distance D. Let A be a set of vectors. Further, let

$$\Gamma^D(A) \triangleq \{\mathbf{y} \in \mathbb{R}^n : \exists_{\mathbf{x}\in A} d(\mathbf{x},\mathbf{y}) \leq D\} \tag{2}$$

be the expansion of set A by the distance D [4]. Then the probability $P\{\text{maybe}\}$ can be written as

$$P\{\text{maybe}\} = \sum_{k=1}^{K} P\{Q(\mathbf{X}) = k\}\, P\left\{\mathbf{Y} \in \Gamma^D(C_k)\right\}. \tag{3}$$

B. Orthogonal Transforms

We seek a linear transformation to transform the data into uncorrelated coefficients. We desire an orthogonal transform because orthogonality preserves the Euclidean distance. We choose the Karhunen-Loève Transform (KLT) because it is the orthonormal transform that optimally decorrelates different dimensions of a data vector and compacts energy into the first few dimensions. In the following text, we give a brief description of the KLT and its Euclidean distance conservation property. We use bold characters $\mathbf{x}$ to denote vectors, and non-bold characters x to denote one component of the vector $\mathbf{x}$.

1) Karhunen-Loève Transform: The Karhunen-Loève transform (KLT) is a type of orthogonal transform that depends on the covariance matrix of the data. Let $\tilde{\mathbf{x}} \in \mathbf{R}^n$ be a real random vector of n dimensions, Φ be the eigenmatrix of the covariance matrix $R_{\tilde{\mathbf{x}}}$ of $\tilde{\mathbf{x}}$. Since the covariance matrix is symmetric, its eigenmatrix Φ is orthonormal $\Phi\Phi^T = I$. We use the transpose of the eigenmatrix Φ^T as the KLT of $\tilde{\mathbf{x}}$ which gives uncorrelated coefficients $\mathbb{E}\{\mathbf{x}\mathbf{x}^T\} = \mathbb{E}\{\Phi^T\tilde{\mathbf{x}}(\Phi^T\tilde{\mathbf{x}})^T\}$=$\Phi^T R_{\tilde{\mathbf{x}}}\Phi = \Lambda$, where Λ is a diagonal matrix with diagonal entries as the eigenvalues of the covariance matrix $R_{\tilde{\mathbf{x}}}$.

2) Euclidean Distance Conservation: The Euclidean distance between two vectors in the original vector space is $\tilde{\mathbf{x}}_1$and $\tilde{\mathbf{x}}_2$ is $\|\tilde{\mathbf{x}}_1 - \tilde{\mathbf{x}}_2\|_2^2 = \|\tilde{\mathbf{x}}_1\|^2 + \|\tilde{\mathbf{x}}_2\|^2 - 2\langle\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2\rangle$. Due to the energy conservation property of orthogonal transforms, the vector length in the transform domain is the same as in the original space. Then the Euclidean distance between two vectors is only determined by their inner product $\langle\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2\rangle$. Let

$\mathbf{x}_1$ and $\mathbf{x}_2$ be the corresponding vectors in transform domain. Eqn. 4 shows that the inner product between two vectors in the original vector space is the same as in the transform domain. Hence, the Euclidean distance between two vectors is preserved through the orthogonal transform.

$$\begin{aligned}\langle \mathbf{x}_1, \mathbf{x}_2 \rangle &= \mathbf{x}_1^T \mathbf{x}_2 \\ &= (\Phi \tilde{\mathbf{x}}_1)^T (\Phi \tilde{\mathbf{x}}_2) \\ &= \tilde{\mathbf{x}}_1^T \Phi^T \Phi \tilde{\mathbf{x}}_2 \\ &= \tilde{\mathbf{x}}_1^T \tilde{\mathbf{x}}_2 \\ &= \langle \tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2 \rangle \end{aligned} \tag{4}$$

III. Similarity Queries in Transform Domain

In the original space, let a query visit a quantization cell and the query $\mathbf{y}$ is compared to the centroid $\hat{\mathbf{x}}$ of the cell to decide whether $\mathbf{y}$ and the data items associated with the cell are similar or not. The Euclidean distance follows the triangle inequality and is used as the similarity measure. The decision rule Eqn. 5 guarantees that the system will not produce any false negatives as required for D-admissible systems

$$g(Q(\mathbf{x}), \mathbf{y}) = \begin{cases} \text{maybe} & d(\mathbf{y}, \hat{\mathbf{x}}) \le d^*(\mathbf{x}, \hat{\mathbf{x}}) + D; \\ \text{no} & \text{otherwise.} \end{cases} \tag{5}$$

Similarity queries in the transform domain should retrieve all the true positive data items that are retrieved in the original space under the same similarity threshold D. Due to the Euclidean distance conservation property of the KL transform, the original as well as the transform domain share the same similarity measure. Therefore, Eqn. 5 still applies in the transform domain. Specifically, we discuss two schemes that handle the similarity queries in the transform domain.

A. Vector-based Similarity Queries

For the vector-based approach, we directly compare query vectors against database centroid vectors of the cells in the transform domain. Since the KLT is an orthogonal transform, the transform partitions the space into parallelepipeds. To approximate the similarity of parallelepiped cells $d^*(\mathbf{x}, \hat{\mathbf{x}})$, we simply sum up the similarities $d^*(x^i, \hat{x}^i)$ of uncorrelated transform coefficients. This forms an upper bound for the true cell similarity, $d^*(\mathbf{x}, \hat{\mathbf{x}}) \le \sum_{i=1}^{n} d^*(x^i, \hat{x}^i)$. Then the vector-based decision rule is given as

$$g(Q(\mathbf{x}), \mathbf{y}) = \begin{cases} \text{maybe} & d(\mathbf{y}, \hat{\mathbf{x}}) \le \sum_{i=1}^{n} d^*(x^i, \hat{x}^i) + D; \\ \text{no} & \text{otherwise} \end{cases} \tag{6}$$

Fig. 2 shows that the similarity approximation of parallelepiped cells results in larger sphere-shaped cells. While the larger approximated cell similarity guarantees that the system remains D-admissible, the interstices created by packed spheres degrade the performance.

B. Component-based Similarity Queries

For the component-based approach, transform coefficients are quantized separately and each component forms a distinct

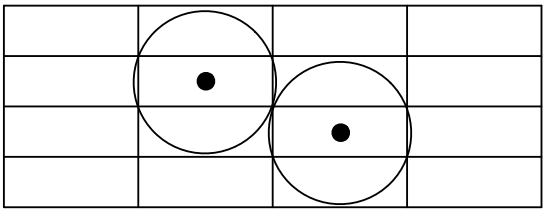

Fig. 2: Spherical cells approximated by the corresponding parallelepiped cells. The approximated cell similarity is much larger than the original cell similarity.

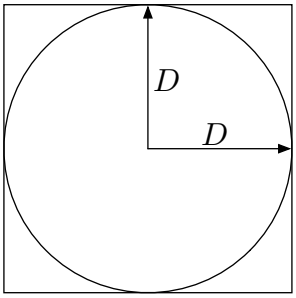

Fig. 3: Vector-based similarity query (sphere) and component-based similarity query (square).

D_i-admissible system, where $D_i \le D$. For multivariate Gaussian data, the transform coefficients are statistically independent. To guarantee that the whole system is D-admissible, each component should also constitute a D-admissible system, that is $D_i = D$. That is, the shape of the query changes from hypersphere in the origin domain to hypercube with edge length D_i in the transform domain. See Fig. 3.

In the query process, the i-th component of the query transform coefficient y_i searches the quantization intervals of the corresponding component under the similarity threshold D. Then we have the component-based decision rule as

$$g(q^i(x^i), y^i) = \begin{cases} \text{maybe} & d(y^i, \hat{x}^i) \le d^*(x^i, \hat{x}^i) + D; \\ \text{no} & \text{otherwise,} \end{cases} \tag{7}$$

where q^i is the quantizer and $\hat{x}^i$ the quantization centroid of the i-th component. The database vectors are labeled as *maybe* if and only if all its transform coefficients are determined as *maybe*.

IV. Proposed Encoding-Retrieval Schemes

The compression is performed in the transform domain. Assume that we have a database with N vectors $X = \{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots, \tilde{\mathbf{x}}_N\}$, where $\tilde{\mathbf{x}}_i \in \mathbf{R}^n$. After performing the KLT on the data set, the n transform coefficients are then quantized by n distinct scalar quantizers. Each transform coefficient generates a sub-codebook $\mathcal{C}^i$. Then one data vector is encoded as the concatenation of its encoded transform coefficients $Q(\mathbf{x}) = [q^1(x^1), q^2(x^2), ..., q^n(x^n)]$. The complete codebook is obtained by the Cartesian product of component codebooks $\mathcal{T} = \mathcal{C}^1 \times \mathcal{C}^2 \times \cdots \times \mathcal{C}^n$.

A. Vector-based Scheme

The vector-based retrieval scheme performs exhaustive search on the database. The codeword of a cell is the concatenation of the component centroids $\hat{\mathbf{x}} = [\hat{x}^1, \hat{x}^2, \ldots, \hat{x}^n]$. Guided by the decision rule Eqn. 6, the query retrieves the data items that are associated with the cells labeled as *maybe*.

The vector-based retrieval scheme requires $k_1 \times k_2 \ldots \times k_n$ distance calculations for one query, where k_i stands for the number of quantization intervals for the i-th component. Therefore, the computational costs of the vector-based scheme grows exponentially with the dimension of the data vector. Hence, the vector-based scheme is infeasible for applications with high dimensional data vectors.

B. Component-based Scheme

1) Bit allocation: The KLT compacts the signal energy into a few components. Hence it is reasonable to assign different rates to the components based on their corresponded component variance. For multivariate Gaussian signals, the components are statistically independent. We formulate the bit allocation problem to minimize the overall $P\{\text{maybe}\}$ as

$$\begin{aligned} \min \quad & \prod_{i=1}^{M} P_i \\ \text{s.t.} \quad & \frac{1}{M}\sum_{i=1}^{M} R_i \leq R, \end{aligned} \tag{8}$$

where R_i is the rate for i-th component, and P_i is $P\{\text{maybe}\}$ of the i-th component.

We use the theoretical *identification rate* for Gaussian sources to assign component rates. According to [1], the *identification rate* for the case that query and database have the same Gaussian distribution with finite second moment σ^2 is

$$R_{\text{ID}}(D, P_X, P_Y) = \begin{cases} \log(\frac{2\sigma^2}{2\sigma^2 - D}) & \text{for } 0 \leq D < 2\sigma^2 \\ \infty & \text{for } D \geq 2\sigma^2 \end{cases} \tag{9}$$

Eqn. 9 shows that if the similarity threshold D is larger than $2\sigma^2$, then $\mathbf{X}$ and $\mathbf{Y}$ are inherently D-similar. In this case, $P\{\text{maybe}\}$ converges to 1. Therefore, we assign 0 bits to the components which similarity threshold D is larger than twice of their variance. For the case $0 \leq D < 2\sigma_i^2$, we explore the relation between rate and component variance as shown in Eqn. 9. Specifically, we set a base rate R_{base} to the component with the largest eigenvalue λ_{base}. Then the rate assigned to the i-th component is depended on the ratio of two component identification rates. In this sense, the component rate is determined by both component variance and similarity threshold D. Since the component variances are equal to the corresponding eigenvalues of the covariance matrix, we can summarize the bit allocation scheme for transform coefficients as

$$R_i = \begin{cases} R_{base} - \log \frac{\lambda_{base}(2\lambda_i - D)}{\lambda_i(2\lambda_{base} - D)} & \text{for } 0 \leq D < 2\lambda_i \\ 0 & \text{for } D \geq 2\lambda_i \end{cases} \tag{10}$$

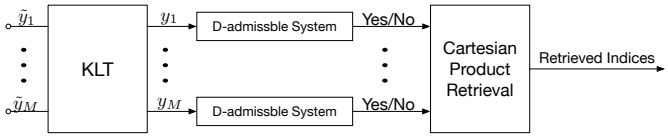

Fig. 4: Component-based retrieval scheme.

2) Component-based Retrieval: The component-based retrieval scheme is shown in Fig. 4. We use the KLT as obtained from the database data to transform the query into the transform domain. For each component, we follow the decision rule Eqn. 7 and retrieve indices that are labeled as *maybe*. Let $\mathcal{R}(y_i)$ denote the retrieved index set of the i-th component. In this way, the total indices that should be retrieved are obtained by the Cartesian product of the retrieved component index sets $\mathcal{R}(\mathbf{y}) = \mathcal{R}(y_1) \times \mathcal{R}(y_2) \times \cdots \times \mathcal{R}(y_n)$.

3) C-tree Structure: The component-based scheme requires only $k_1 + k_2 + \ldots + k_n$ one-dimension distance calculations. The major computational cost is the scanning of the combinations of the retrieved component index set. This computational cost is exponential in the number of bits. Hence, we propose a tree-structured codebook on the product codes, namely C-trees, to avoid the scanning of codewords that are not assigned to database items. C-trees iteratively split the data space based on the vector components.

The construction of C-trees involves *training* and *enroll* steps. In the *training* step, the component codebooks of the first m principle components are generated independently. The size of each component codebook $k_i = 2^{R_i}$ is determined by the bit allocation. After the training step, we *enroll* the database items into the C-tree with trained component codebooks. We start from the first principle component with the highest eigenvalue. That is, the data items are assigned to their corresponding cells by the nearest neighbor selection on their first principle component. Then the process is recursively applied to each cell of the remaining components that are ordered by their corresponding eigenvalues. The descending divisions are only defined by the data items that belong to the parent cell. The enroll step is completed after all the database items are assigned to the leaf nodes. In this sense, the data that is labeled with the associated leaf node index can be stored separately from the tree.

V. Experimental Results

A. Experimental Setup

We test our proposed schemes on both synthetic data and real data from image descriptors. For the synthetic data, we use $N = 10^5$ random vectors generated from a 2-dimensional Gaussian source with correlation coefficient $\rho = 0.6$. For the real data, we extract the first 24 dimensions of one million SIFT image descriptors from the SIFT1M dataset [6].

To make sure that we have the ground truth for the experiment, we sample M vectors directly from the database and use them as queries. $P\{\text{maybe}\}$ is regarded as the performance

measure of the scheme. The lower $P\{\text{maybe}\}$, the better. Given a single query, $P\{\text{maybe}|\mathbf{y}\}$ is estimated by

$$\hat{P}\{\text{maybe}|\mathbf{y}\} = \frac{\sum_{k\in R(\mathbf{y})} |C_k|}{N}, \tag{11}$$

where the retrieval process $R(\mathbf{y})$ outputs the indices of the retrieved nodes. $|C_k|$ is the number of data items associated with the node k. Finally, $P\{\text{maybe}\}$ is obtained by averaging over all M queries

$$\hat{P}\{\text{maybe}\} = \frac{1}{M}\sum_{i=1}^{M} \hat{P}\{\text{maybe}|\mathbf{y}\}. \tag{12}$$

B. Results

We compare $P\{\text{maybe}\}$ for the two transform-based schemes and the non-transform product codes. Full-search vector quantization serves as the baseline method. We test all four schemes on the synthetic data. We omit the test of the vector-based scheme on the SIFT1M data due to the extremely high computational cost of its exhaustive search. Further, as shown for the synthetic data, its performance is worse than the component-based scheme.

Figs. 5 and 6 show the results of two-dimensional Gaussian and SIFT1M data, respectively. We can make three observations from the results. First, full-complexity unstructured clustering has the best performance for two-dimensional Gaussian data due to its space-filling advantages [7]. However, the complexity for full search is $O(N^{nK+1}\log N)$ with K clusters, N data samples, and n-dimensional vectors. The complexity grows exponentially as data rate and volume increase. On the other hand, the product codes use scalar components and the complete codebook is generated by the Cartesian product of component codebooks. Hence, the computational complexity is limited and product codes are more practical. Second, among the transform-based schemes, the component-based scheme performs better than the vector-based scheme. The reason is that the vector-based scheme uses an approximate cell similarity which can be much higher than the true cell similarity in the decision rule. This increases the false positive errors of the scheme significantly. Third, the component-based scheme with KLT has better performance than the component-based scheme without KLT (non-transform scheme). This is due to the efficient bit allocation among individual component systems. The advantage of the component-based scheme with KLT is more obvious for the SIFT1M dataset.

VI. Conclusions

We explore transform-based compression schemes for quadratic similarity queries. For that, we consider orthonormal transforms that preserve the Euclidean distance. We compare vector-based and component-based approaches. We give component-based decision rules for quadratic similarity queries. We also propose a bit allocation solution for our component systems. Our experiments show that the component-based scheme with KLT shows good performance for handling high-dimensional data while maintaining a low complexity.

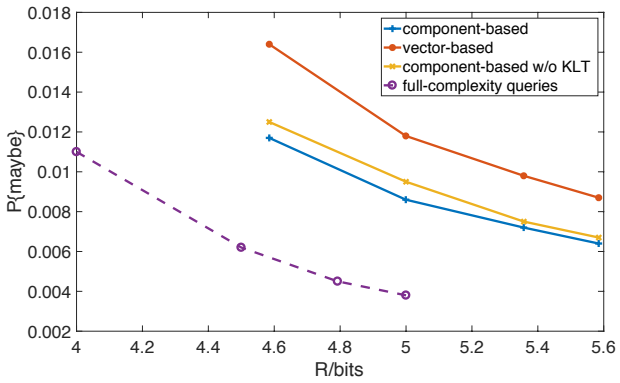

Fig. 5: Results for two-dimensional Gaussian data with $\sigma^2 = 1$ and ρ = 0.6.

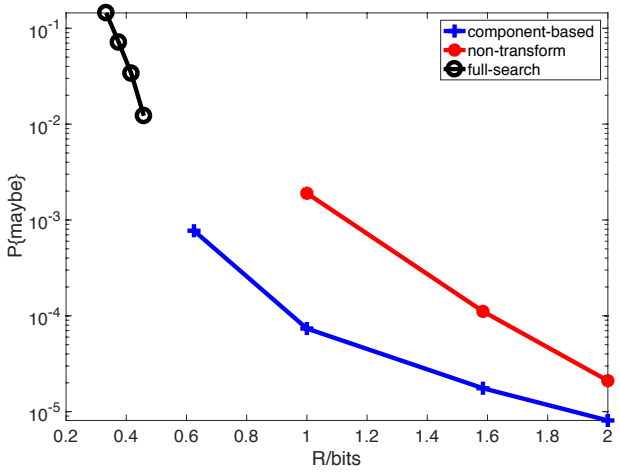

Fig. 6: Results for SIFT1M data.

References

[1] A. Ingber, T. Courtade, and T. Weissman, "Compression for quadratic similarity queries," *IEEE Trans. on Information Theory*, vol. 61, no. 5, pp. 2729 –2747, May 2015.

[2] A. Ingber and T. Weissman, "The minimal compression rate for similarity identification," 2013, [Online]. Available: http://arxiv.org/abs/1312.2063.

[3] I. Ochoa, A. Ingber, and T.Weissman, "Compression schemes for similarity queries," in *Proc. of the IEEE Data Compression Conference*, Mar. 2014.

[4] F. Steiner, S. Dempfle, A. Ingber, and T. Weissman, "Compression for quadratic similarity queries: finite blocklength and practical schemes," *IEEE Trans. on Information Theory*, vol. 62, no. 5, pp. 2737 –2747, May 2016.

[5] H. Wu, Q. Wang, and M. Flierl, "Tree-structured vector quantization for similarity queries," in *2017 Data Compression Conference (DCC)*, April 2017.

[6] H. Jegou, M. Douje, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615 –1630, 2011.

[7] T. Lookabaugh and R. Gray, "High-resolution quantization theory and the vector quantizer advantage," *IEEE Trans. on Information Theory*, vol. 35, no. 5, pp. 1020–1033, Sep. 1989.