Ptolemaic Indexing of the Signature Quadratic Form Distance

Jakub Lokoč
Charles University in Prague, Faculty of Mathematics and Physics, SIRET research group
lokoc@ksi.mff.cuni.cz

Magnus Lie Hetland
Norwegian University of Science and Technology, Dept. of Computer Science
mlh@idi.ntnu.no

Tomáš Skopal
Charles University in Prague, Faculty of Mathematics and Physics, SIRET research group
skopal@ksi.mff.cuni.cz

Christian Beecks
RWTH Aachen University in Germany, Data Management and Data Exploration Group
beecks@cs.rwth-aachen.de

ABSTRACT

The *signature quadratic form distance* has been introduced as an adaptive similarity measure coping with flexible content representations of multimedia data. While this distance has shown high retrieval quality, its high computational complexity underscores the need for efficient search methods. Recent research has shown that a huge improvement in search efficiency is achieved when using metric indexing. In this paper, we analyze the applicability of *Ptolemaic indexing* to the signature quadratic form distance. We show that it is a *Ptolemaic metric* and present an application of Ptolemaic pivot tables to image databases, resolving queries nearly four times as fast as the state-of-the-art metric solution, and up to 300 times as fast as sequential scan.

Categories and Subject Descriptors

H.3.1 [**content analysis and indexing**]: indexing methods

General Terms

Theory, Experimentation, Performance

1. INTRODUCTION

The explosive growth of complex multimedia data including images, videos, and music challenges the effectiveness and efficiency of today's multimedia databases. Supposed to provide users access and insight into these inevitably increasing masses, multimedia databases have to manage data objects effectively and appropriately with respect to contents access. When searching multimedia databases in a content-based way, users issue similarity queries by selecting multimedia objects or by sketching the intended object contents. Given an example multimedia object or sketch, the multimedia database searches for the most related objects with respect to the query by measuring the similarity between the query and each database object by means of a distance function. As a result, the multimedia objects with the lowest distance to the query are returned to the user.

In fact, when determining content-based similarity between two multimedia objects, the distance is evaluated on *feature representations* which aggregate the inherent properties of the multimedia objects. The conventional feature representations aggregate and store these properties in *feature histograms*, which can be compared by vectorial distances [16, 23]. Recent feature representations adaptively aggregate and store individual object properties in more flexible *feature signatures*, which can be compared by adaptive similarity measures [3]. It has been shown that the *signature quadratic form distance* (SQFD) [2, 5] yields high retrieval effectiveness [2, 5, 3]. However, the distance *indexability* (or search efficiency) remains a challenging issue. While previous solutions [4, 6] relying on the *sequential scan* of the database provide approximate and exact search results with a comparatively low speed-up, the recently introduced approach of Beecks et al. [1] using *metric access methods* [9, 28] improves the efficiency of accessing multimedia databases with a speed-up factor of up to 170.

1.1 Paper Contributions

In this paper, we use *Ptolemaic pivot tables* (PPT), originally described by Hetland [15], for efficient content-based similarity search in large multimedia databases. Ptolemaic indexing has been shown to be particularly efficient for quadratic form distances (QFDs). Unfortunately, as the Ptolemaic approach suffers from an increased internal complexity, it is mainly suitable for expensive distances, where this extra complexity becomes insignificant. Hence, it may not be a feasible solution for one particular kind of QFD: the cheap (weighted) Euclidean distance. Moreover, it has recently been shown that for indexing purposes, all *static* QFDs can be mapped to the Euclidean case [26], so Ptolemaic indexing may not be viable even for them.

However, the mapping to Euclidean case does not effectively apply to the more expressive family of *signature* quadratic form distances. In this paper we show both that these distances are Ptolemaic, and that Ptolemaic indexing

SISAP '11, June 30-July 1, 2011, Lipari, Italy.

is a clear improvement on the state of the art [1] for indexing them. The main contributions of this paper are:

- A proof sketch that the SQFD is a Ptolemaic metric.
- New heuristics for efficiently performing the Ptolemaic filtering which lead to an improvement in the real-time efficiency of the PPT method.
- Empirical evidence that PPT is an efficient index for the SQFD, also when combined with metric pivoting.

The structure of this paper is as follows: in Section 2 we briefly describe the SQFD. In Section 3 we outline related work. In Sections 4 and 5 we investigate Ptolemaic indexing of the SQFD. We report the experimental results in Section 6 before we conclude our paper in Section 7.

2. SQFD AND FEATURE SIGNATURES

In this section, we briefly review the SQFD as an adaptive similarity measure between feature signatures which represent objects by individually aggregating their properties in a compact way. Unlike conventional feature histograms, feature signatures are frequently obtained by clustering the objects' properties, such as color, texture, or other more complex features [10, 21], within some feature space and storing the cluster representatives and weights. Thus, given a feature space $\mathbb{F}$, the *feature signature* S^o of a multimedia object o is defined as a set of tuples from $\mathbb{F} \times \mathbb{R}^+$ consisting of representatives $r^o \in \mathbb{F}$ and weights $w^o \in \mathbb{R}^+$.

We depict an example of image feature signatures according to a feature space comprising position and color information, i.e. $\mathbb{F} \subseteq \mathbb{R}^5$, in Figure 1. For this purpose we applied a k-means clustering algorithm where each representative $r_i^o \in \mathbb{F}$ corresponds to the centroid of the cluster $\mathcal{C}_i^o \subseteq \mathbb{F}$, i.e., $r_i^o = \frac{\sum_{f \in \mathcal{C}_i^o} f}{|\mathcal{C}_i^o|}$, with relative frequency $w_i^o = \frac{|\mathcal{C}_i^o|}{\sum_i |\mathcal{C}_i^o|}$. We depict the feature signatures' representatives by circles in the corresponding color. The weights are reflected by the diameter of the circles. As can be seen in this example, feature signatures adjust to individual image contents by aggregating the features according to their appearance in the underlying feature space.

To compare feature signatures we make use of the SQFD which is a generalization of the conventional *quadratic form distance* (QFD) [13]. It is defined for the comparison of two feature signatures of different structure and size as follows.

Definition 1 (SQFD). *Given two feature signatures* $S^q = \{\langle r_i^q, w_i^q \rangle\}_{i=1}^n$ *and* $S^p = \{\langle r_i^p, w_i^p \rangle\}_{i=1}^m$ *and a similarity function* $f_s : \mathbb{F} \times \mathbb{F} \to \mathbb{R}$ *over some feature space* $\mathbb{F}$*, the signature quadratic form distance* SQFD_{f_s} *between* S^q *and* S^p *is defined as:*

$$\mathrm{SQFD}_{f_s}(S^q, S^p) = \sqrt{(w_q \mid -w_p) \cdot A_{f_s} \cdot (w_q \mid -w_p)^T},$$

where $A_{f_s} \in \mathbb{R}^{(n+m)\times(n+m)}$ *is the similarity matrix arising from applying the similarity function* f_s *to the corresponding representatives, i.e.,* $a_{ij} = f_s(r_i, r_j)$*. Furthermore,* $w_q = (w_1^q, \ldots, w_n^q)$ *and* $w_p = (w_1^p, \ldots, w_m^p)$ *form weight vectors, and* $(w_q \mid -w_p) = (w_1^q, \ldots, w_n^q, -w_1^p, \ldots, -w_m^p)$ *denotes the concatenation of weights* w_q *and* $-w_p$*.*

As can be seen in Definition 1, the signature quadratic form distance takes into account the similarity values between any two representatives by making use of a similarity function f_s. This similarity relationship is reflected within the similarity matrix A_{f_s} which has to be determined for each distance computation individually. Thus, the complexity of a single distance computation is in $\mathcal{O}\big((n+m)^2 \cdot \phi\big)$ where n and m denote the size of feature signatures S^q and S^p, respectively, and ϕ denotes the complexity of the similarity function f_s over some feature space $\mathbb{F}$. Beecks et al. [5] proposed three example similarity functions:

- *Minus function:* $f_-(r_i, r_j) = -d(r_i, r_j)$
- *Heuristic function:* $f_h(r_i, r_j) = \frac{1}{\alpha + d(r_i, r_j)}$
- *Gaussian function:* $f_g(r_i, r_j) = e^{-\alpha \cdot d^2(r_i, r_j)}$

It turns out that the Gaussian function with the parameter $\alpha \in \mathbb{R}^+$ adapted to the current multimedia database exhibits the highest retrieval performance in terms of effectiveness, while the minus function results in the lowest computation time [3].

Figure 1: Three example images with their corresponding feature signature visualizations.

Beyond images, feature signatures can also be applied to other kinds of multimedia data which fit into this flexible feature representation form. In general, there are no obstacles to using a non-vectorial feature space $\mathbb{F}$, so the SQFD has prospects of becoming a universal distance for measuring locality-sensitive similarity between any complex signatures consisting of local features.

3. INDEXING SQFD – RELATED WORK

In this section, we first briefly explain the fundamentals of metric indexing with a particular focus on the simplest and most intuitive metric access method: *pivot tables*. We then outline related work regarding metric indexing of the SQFD.

3.1 Metric Access Methods

A *metric space* $(\mathbb{U}, \delta)$ consists of a feature representation domain $\mathbb{U}$ (in this paper, the set of all possible signatures) and a distance function δ which has to satisfy the metric postulates: *identity*, *non-negativity*, *symmetry*, and *triangle inequality*. In this way, metric spaces allow domain experts to model their notion of content-based similarity by an appropriate feature representation and distance function serving as similarity measure. At the same time, this approach allows database experts to design index structures, so-called *metric access methods* (or metric indexes) [9, 28, 24, 14], for efficient query processing of content-based similarity queries in a database $\mathbb{S} \subset \mathbb{U}$. These methods rely on the distance function δ only, i.e., they do not necessarily know the structure of the feature representation of the objects.

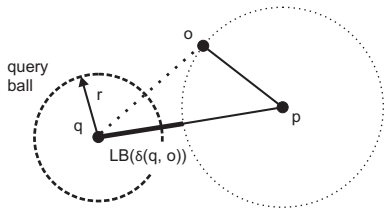

Figure 2: The lower-bounding principle.

Metric access methods organize database objects $o_i \in \mathbb{S}$ by grouping them based on their distances, with the aim of minimizing not only traditional database costs like I/O but also the number of costly distance function evaluations. For this purpose, nearly all metric access methods apply some form of filtering based on cheap lower bounds. For the case of pivoting, these bounds are based on the fact that exact pivot–object distances are pre-computed.

We illustrate this fundamental principle in Figure 2 where we depict the query object $q \in \mathbb{U}$, some pivot element $p \in \mathbb{S}$, and a database object $o \in \mathbb{S}$ in some metric space. Given a range query (q, r), we wish to estimate the distance $\delta(q, o)$ by making use of $\delta(q, p)$ and $\delta(o, p)$, with the latter already stored in the metric index. Because of the triangle inequality, we can safely filter object o without needing to compute the (costly) distance $\delta(q, o)$ if the triangular lower bound

$$\delta_{\mathrm{T}}(q, o) = |\delta(q, p) - \delta(o, p)|\,, \tag{1}$$

also known as the *inverse triangle inequality*, is greater than the query radius r.

3.1.1 Pivot Tables

One of the most efficient (yet simple) metric indexes is the *pivot table* [22], originally introduced as LAESA [20]. Basically, the structure of a pivot table is a simple matrix of distances $\delta(o_i, p_j)$ between the database objects $o_i \in \mathbb{S}$ and a pre-selected static set of m pivots $p_j \in \mathbb{P} \subset \mathbb{S}$. For querying, pivot tables allow us to perform cheap lower-bound filtering by computing the maximum lower bound (1) to $\delta(q, o)$ using all the pivots.

From a more intuitive perspective, pivot tables index the database objects as m-dimensional vectors in a pivot space. When querying, the range query ball (q, r) (or kNN ball with the current radius) is mapped into the pivot space, such that its center is $(\delta(q, p_1), \delta(q, p_2), \ldots, \delta(q, p_m))$. An important property of the mapping is that δ in the original space is lower-bounded by L_∞ distance in the pivot space (i.e., it is a non-expansive mapping). The query ball in the pivot space (i.e., the L_∞-ball of radius r) can therefore be used to retrieve all the objects inside the query ball in the original space, possibly with some false positives that must be filtered out by δ in a refinement step. See Figure 3 for an illustration of the pivot-based mapping from the original space into the pivot space, and the respective query balls.

3.2 Metric indexing of the SQFD

Unlike previous approaches which focus on improving the efficiency of a sequential scan by making use of *maximum component feature signatures* [4] or *similarity matrix compression* [6], a recently introduced approach exploits the indexability of the SQFD by investigating the parameters of

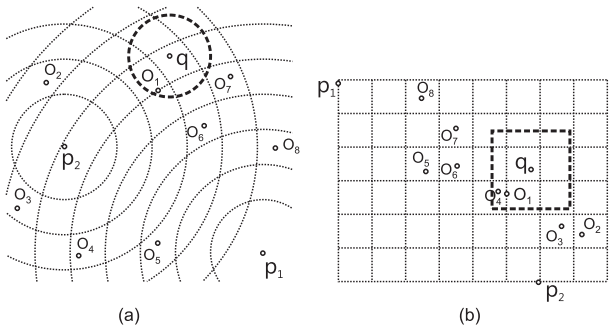

Figure 3: (a) Original space (b) Pivot space

the similarity function and indexing the data through simple pivot tables [1]. Although the approaches that improve sequential scan can be used when adapting the similarity function to user preferences, they only reach a speed-up factor of up to 13 and 9, respectively. In contrast, using pivot tables yields a speed-up factor of up to 170 when modifying the inherent parameter α used by the similarity function f_s. In fact, Beecks et al. [1] show that the similarity function f_s applied inside SQFD not only determines the retrieval quality but also the indexability. Thus, changing the parameters of the similarity function will result in better indexability.

However, in this paper we investigate the indexability of the SQFD beyond the lower-bounding principle based on triangle inequality. We propose to use the Ptolemaic indexing approach introduced by Hetland [15] to improve the efficiency of processing content-based similarity queries even further. Our approach is detailed in the next section.

4. PTOLEMAIC INDEXING OF THE SQFD

This section first summarizes the principles of Ptolemaic indexing, and then shows that they apply to the SQFD.

4.1 The Principles of Ptolemaic Indexing

In metric indexes, the triangle inequality is used to construct lower bounds for the distance. Analogously, in *Ptolemaic indexing* [15], *Ptolemy's inequality* is used to construct such lower bounds as well. A distance function is called a *Ptolemaic distance* if it has the properties of *identity*, *non-negativity*, and *symmetry*, and satisfies *Ptolemy's inequality*. If a Ptolemaic distance also satisfies the *triangle inequality*, it is a *Ptolemaic metric*.

Ptolemy's inequality states that for any quadrilateral, the pairwise products of opposing sides sum to more than the product of the diagonals. In other words, for any four points x, y, u, $v \in \mathbb{U}$, we have the following:

$$\delta(x, v) \cdot \delta(y, u) \leq \delta(x, y) \cdot \delta(u, v) + \delta(x, u) \cdot \delta(y, v) \tag{2}$$

One of the ways the inequality can be used for indexing is in constructing a pivot-based lower bound. For a query q, object o, and pivots p and s, we get the *candidate bound*:

$$\delta_{\mathrm{C}}(q, o, p, s) = \frac{|\delta(q, p) \cdot \delta(o, s) - \delta(q, s) \cdot \delta(o, p)|}{\delta(p, s)} \tag{3}$$

For simplicity, we let $\delta_{\mathrm{C}}(q, o, p, s) = 0$ if $\delta(p, s) = 0$. As for triangular lower-bounding, one would normally have a set of pivots $\mathbb{P}$, and the bound can then be maximized over

all (ordered)[1] pairs of distinct pivots drawn from this set, giving us the final Ptolemaic bound [15]:

$$\delta(q,o) \geq \delta_{\mathrm{P}}(q,o) = \max_{p,s \in \mathbb{P}} \delta_{\mathrm{C}}(q,o,p,s) \qquad (4)$$

As for the triangular case, the Ptolemaic lower bound δ_{P} could be used to filter objects not contained in the query ball, i.e., exclude those $o_i \in \mathbb{S}$ from search for which $\delta_{\mathrm{P}}(q,o_i) > r$.

4.2 The SQFD is a Ptolemaic Metric

This section outlines a proof sketch justifying the use of Ptolemaic and metric indexing for the SQFD. The idea is simple: the SQFD is basically an efficient way of calculating the QFD between extremely sparse weight vectors. Consider an ordinary QFD with vectors in $\mathbb{R}^N$. Assume that for any dimension, at most one vector in the space has a nonzero value. By assuming a representative for each dimension, the similarity matrix for this space can be computed exactly as in the SQFD. It should now be clear that for the QFD over these vectors, most dimensions will be irrelevant. Any dimension where both of the vectors have a value of 0 will be eliminated from the inner product, and we end up actually using a tiny portion of the similarity matrix. This is exactly what is done in the SQFD.

We can look at this the other way around, starting with the SQFD signatures. We first embed the weight vectors of feature signatures into $\mathbb{R}^N$, where each dimension of each weight vector in the original feature space is assigned to a separate dimension in $\mathbb{R}^N$. The number of dimensions, N, is thus the sum of all feature signature sizes in our original distance space. Embedding a weight vector in $\mathbb{R}^N$ simply involves padding it with zeros on either end, so that its weights end up in the correct dimensions. Let $w' \in \mathbb{R}^N$ be the embedded version of any weight vector w. We then have:

$$(w_a \mid -w_b)' = w'_a - w'_b$$

Here, the embedding $(w_a \mid -w_b)' \in \mathbb{R}^N$ is taken to mean the embedding that assigns the values from w_a and w_b to the same dimensions as the individual embeddings w'_a and w'_b.

We can now define a global similarity matrix A, using the same similarity function as for the local matrices A_{f_s}. We can then calculate the QFD between w'_a and w'_b as follows:

$$\begin{aligned}\mathrm{QFD}(w'_a, w'_b) &= \sqrt{(w'_a - w'_b) \cdot A \cdot (w'_a - w'_b)^T} \\ &= \sqrt{(w_a \mid -w_b)' \cdot A \cdot (w_a \mid -w_b)'^T}\end{aligned}$$

In the matrix product $x' A x'^T$, the extra dimensions involved in the embedding are all zero, and don't contribute at all (see Figure 4). Only the original dimensions, and the corresponding entries in A (which together form A_{f_s}), are used in computing the distance. Thus we have $\mathrm{QFD}(w'_a, w'_b) = \mathrm{SQFD}(a,b)$. If A is well-behaved (symmetric positive definite), QFD is a Ptolemaic metric [15]. Since the mapping $x \mapsto x'$ is a distance-preserving isomorphism, the same holds for SQFD. (This presentation assumes unique representatives, but this is not a requirement of the proof. The proof can also be extended to infinite distance spaces, using infinite-dimensional inner products.)

[1]Computing the absolute value is redundant when examining all ordered pairs. It is, however, useful when only *some* pairs are examined, as explained later.

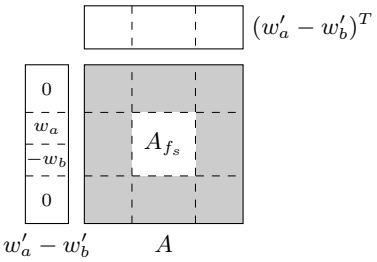

Figure 4: Embedding SQFD vectors in a QFD space. Because of the zero components of $w'_a - w'_b$, the gray areas of A won't affect the inner product.

5. PTOLEMAIC PIVOT TABLES

As mentioned in Section 3.1.1, pivot tables are a simple, yet efficient and extensible metric access method. Moreover, it was recently shown [15] that pivot tables could also be used as a Ptolemaic access method. In this paper we call this method *Ptolemaic pivot tables*, or PPT. Note that in addition to the Ptolemaic lower bounds, the PPT can also employ the lower bound provided by the triangle inequality (thus becoming a Ptolemaic *and/or* metric access method). The following section describes how some simple heuristics can be used to compute the lower bound more efficiently. The data structure needed in order to accommodate these new heuristics is described in Section 5.2.

5.1 Computing the Bound Efficiently

One of the most important sources of filtering power for Ptolemaic lower-bounding is the increased number of candidate bounds – quadratic in the number of pivots. Although this is certainly an advantage for filtering, it can significantly increase the computation time of the final bound. Unless the distance calculation itself is very slow, not many pivots are needed before the computation of the bound becomes prohibitively expensive. One of the contributions of this paper is a specific procedure for computing the bound, which takes advantage of the Ptolemaic filtering power and that is efficient (fast) in practice.

The idea is to perform *online pivot selection*, as used in the original pivot table methods, AESA [27] and LAESA [20], as well as in some more recent methods [11, 7]. It is a fair assumption that the candidate bounds will differ significantly – this is the motivation for using multiple pivots, after all. Our task is to find a good pivot pair, a pair that will let us exclude an irrelevant object. Rather than trying out every pivot pair in an arbitrary order (which we refer to as the *naïve approach*), we can perform a heuristic search for the best ones. As long as the search radius (or the *current* search radius, in a kNN search) is available to us when computing the bound, we can terminate as soon as the radius has been exceeded. With a good heuristic ordering of the pairs, this will usually allow us to terminate early.

A high-quality heuristic will not only allow us to terminate the bound computation early when we are *able* to discard an object; it will give us the confidence needed to end our computation early *by fiat*. That is, if we know the best candidate bounds are probably computed first, and we are unable to eliminate an object early on, we should probably give up. This "giving up point" can either be set to a fixed number of candidate bounds, or it can be based on lack of

improvement over a series of candidate bounds.

5.1.1 Pivot Permutations

Given the structure of the Ptolemaic lower bound (3), it would seem reasonable to expect a good pivot pair to consist of one object-like pivot p and one query-like pivot s.[2] We therefore decided to use low values for $\delta(q,s)$ and $\delta(o,p)$ as the heuristic guidelines in our search. In the heuristics, we require an ordering of the pivots based on distance to a particular object o (either a database or a query object). For an object o we define a *pivot permutation* [25, 8], as follows.

Having a fixed set of m pivots $\mathbb{P} = \{p_1, p_2, \ldots, p_m\}$ and an object $o \in \mathbb{U}$, let $()_o : \{1,2,\ldots,m\} \mapsto \{1,2,\ldots,m\}$ be a permutation such that $\forall i,j \in 1,\ldots,m : (i)_o \leq (j)_o \leftrightarrow \delta(p_{(i)_o}, o) \leq \delta(p_{(j)_o}, o) \vee (\delta(p_{(i)_o}, o) = \delta(p_{(j)_o}, o) \wedge i < j)$. Hence, the sequence $p_{(1)_o}, p_{(2)_o}, \ldots, p_{(m)_o}$ is ordered with respect to distances between the pivots and object o.

In order to efficiently look for pivots that are close to the query or to a given object, we need to precompute and store the pivot permutations $(\cdot)_o$ for every object o. We also compute $(\cdot)_q$ at the beginning of the search. Using these permutations, we generate a sequence of pivot pairs (p,s), and these pivot pairs are used to create candidate bounds.

5.1.2 An Unbalanced Heuristic

The simplest way of using these permutations is simply to use two nested loops, each iterating over one of the permutations (see Algorithm 5.1). Using this *unbalanced* heuristic, either the q-like pivots or the o-like pivots are preferred. In each iteration, the algorithm checks whether it should terminate early, that is, before all pivot pairs have been examined. This happens if either the bound is large enough ($\delta_\mathrm{P} > r$), or if we have already tried κ pivot pairs, where κ is a cut-off parameter.

Algorithm 5.1: UnbalancedFilter$(q, o, r, \kappa) \mapsto \delta_\mathrm{P}$

```
1: δP = c = 0
2: for i = 1 to m do
3:     for j = 1 to m do
4:         δP ← max{δP, δC(q, o, p_(i)o, p_(j)q)}
5:         c ← c + 1
6:         if δP > r or c = κ then
7:             return
```

5.1.3 A Balanced Heuristic

To avoid giving preference to one of the permutations as the unbalanced heuristic does, we propose the *balanced* traversal heuristic (see Algorithm 5.2). This explores cartesian product of $(\cdot)_o$ and $(\cdot)_q$ in a breadth-first fashion, starting at $((1)_o, (1)_q)$. Every o-like pivot (in order of distance from o) is combined with every q-like pivot that is at least as good, i.e., with at most the same rank in distance ordering from q, and vice versa. The checks for early termination work like in the unbalanced heuristic.

Figure 5 shows a comparison of the naïve approach, and both heuristics in a Euclidean space (using 20 pivots drawn from a 10D uniform unit cube distribution). As expected, the balanced heuristic achieves a tight bound the fastest:

[2]That is, p and s are close to object and query, respectively.

Algorithm 5.2: BalancedFilter$(q, o, r, \kappa) \mapsto \delta_\mathrm{P}$

```
1: δP = c = 0
2: for i = 1 to m do
3:     for j = 1 to i do
4:         δP ← max{δP, δC(q, o, p_(i)o, p_(j)q)}
5:         c ← c + 1
6:         if i ≠ j then
7:             δP ← max{δP, δC(q, o, p_(j)o, p_(i)q)}
8:             c ← c + 1
9:         if δP > r or c ≥ κ then
10:            return
```

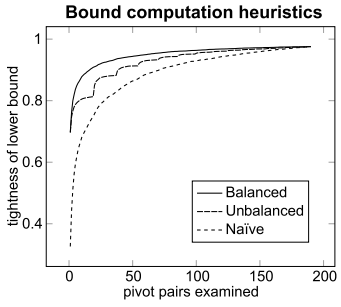

Figure 5: Pivot pair selection heuristics.

after examining 15 pivot pairs the bound gets to 90% of the value obtained by examining all 190 unique pivot pairs.

5.2 The PPT Index Structure

The index structure of PPT consists of two components:

— The pivot file, storing the set $\mathbb{P}$ of m pivots, and a pivot distance matrix, storing the distances between all distinct pairs of pivots from $\mathbb{P}$.

— The index file, storing the distances between each database object $o \in \mathbb{S}$ and all the pivots, the pivot permutation for each object o, and the object o itself. Formally, the index file consists of $|\mathbb{S}|$ entries, each belonging to a database object o, as $[o, (\cdot)_o, \delta(o, p_{(1)_o}), \delta(o, p_{(2)_o}), \ldots, \delta(o, p_{(m)_o})]$.

The difference between regular pivot tables and the PPT is thus an extra information stored: the pivot distance matrix and the pivot permutation for each object o (used by the proposed heuristics).

5.3 Querying the PPT

A similarity query is processed by traversing the index file sequentially. However, for each database object o, we try to avoid computing $\delta(q,o)$ by applying either triangle or Ptolemaic lower-bounding (or both). For the implementation of the PPT range queries, see Algorithm 5.3.

Note that TriangleFilter and PtolemaicFilter compute the lower bound to $\delta(q,o)$, including possible early termination – either due to the current bound exceeding the query radius r, or the number of examined pivot pairs exceeding the limit κ. While the PtolemaicFilter refers to either UnbalancedFilter or BalancedFilter (see Algorithms 5.1, 5.2), the TriangleFilter just applies the triangle inequality test for each pivot $p \in \mathbb{P}$, as usual in the regular pivot tables.

As mentioned, the parameter κ refers to the maximum number of pivot pairs to be examined before giving up on

Algorithm 5.3: RangeQuery($q, r, mode, \kappa$) $\mapsto$ *Result*

```
1: Result = ∅
2: for each o in 𝕊 do
3:     if mode = triangle or mode = triangle+pto then
4:         if TriangleFilter(q, o, r) > r then
5:             continue for
6:     if mode = ptolemaic or mode = triangle+pto then
7:         if PtolemaicFilter(q, o, r, κ) > r then
8:             continue for
9:     compute δ(q, o)
10:    if δ(q, o) ≤ r then
11:        add o to Result
```

computing the bound. Although this parameter could be left to the user, by setting $\kappa = |\mathbb{P}|$ we obtain Ptolemaic lower-bounding of the same time complexity as the triangle one (i.e., $\mathcal{O}(|\mathbb{P}|)$).

6. EXPERIMENTAL EVALUATION

In this section, we evaluate the efficiency of search under SQFD. We compare the PPT with the original pivot tables, so far the state-of-the-art metric index applied to SQFD.

6.1 The Testbed

We make use of the *MIR Flickr* database [17] including 25,000 web-images with textual annotations, and the *ALOI* database [12] comprising 72,000 images. The selected databases are not very large but they provide a ground truth for evaluating the search effectiveness.

We extracted feature signatures from the aforementioned databases, based on seven-dimensional features $(L, a, b, x, y, \chi, \eta) \in \mathbb{F}$ including color (L, a, b), position (x, y), contrast χ, and coarseness η information. These features were extracted for a randomly selected subset of pixels for each image and then aggregated by applying an adaptive variant of the k-means clustering algorithm described by Leow and Li [18]. Thus, we obtain one feature signature for each single image. These signatures vary in size between 5 and 115 feature representatives. On average, a feature signature consists of 54 representatives (i.e., 432 numbers per signature). The remaining settings in our experiments were the same as those used by Beecks et al. [1]. The tests ran on a workstation 2x Intel Xeon X5660 2.8 Ghz, 24GB RAM, Windows Server 2008 R2 64bit (non-virtualized).

Label	Description
Tri(n)	PPT using triangle mode
Pto(n, U|B|N, κ)	PPT using Ptolemaic mode
TriPto(n, U|B|N, κ)	PPT using triangle+Ptolemaic mode n is the number of pivots used U|B|N stands for Unbalanced, Balanced or Naïve heuristics κ is the max. number of pivot pairs used
% of Tri(n)	performance related to query realtime achieved by Tri(n)

Table 1. Labels used in the figures.

In Table 1, we summarize the description of labels used within the following figures. Note that Tri($\cdots$) denotes the original pivot tables used as a referential metric access method, while the Pto($\cdots$) and TriPto($\cdots$) labels refer to specific filtering modes of PPT (see Section 5.3).

6.2 The Results

We first present the SQFD properties in terms of *intrinsic dimensionality* (iDim) [9] and *mean average precision* (MAP) values [19], as they indicate whether the SQFD allows for efficient and effective indexing (see Figure 6). The crucial parameter alpha (denoted α in Section 2) is applied inside the Gaussian similarity function f_s when computing the SQFD matrix, and it has significant impact on iDim and MAP. For more about tuning iDim and MAP using alpha, see the paper by Beecks et al. [1].

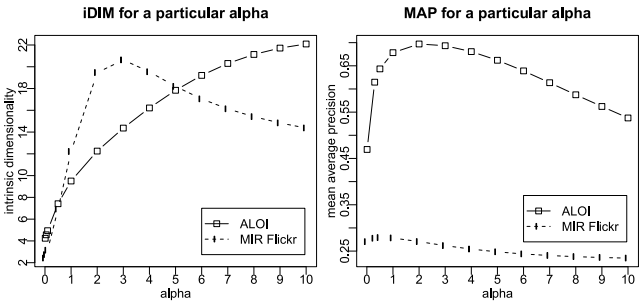

Figure 6: intrinsic dim. vs. mean average precision.

Next, we study the increase in efficiency in terms of kNN query response times. As the numbers of distance computations perfectly correlate with the real response times (because of SQFD's computational cost), we present only the real times in the figures. In general, a single SQFD computation takes on average 0.65 ms for the ALOI database, and 0.79 ms for the MIR Flickr database. The number of distance computations could be reconstructed using these values, as well as the real response times of the sequential scan. The query times were averaged for 100 different queries, while the query signatures were not indexed.

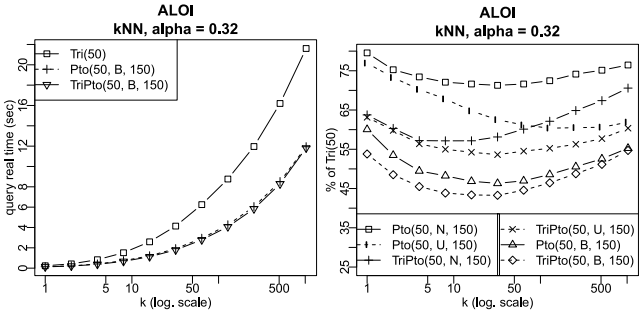

Figure 7: kNN queries on the ALOI database.

See Figure 7 for the performance of kNN queries on the ALOI database, 50 pivots, and alpha=0.32. The PPT is a clear winner in all configurations, while the Balanced heuristic always works best. Also note that the maximum relative speed-up with respect to Tri(50) is achieved for k=10–50.

The impact of the maximum number of pivot pairs used in the Ptolemaic bound computation is presented in Figure 8. It shows that the Ptolemaic filtering is effective even for a small number of pivot pairs (Pto), while it is further improved when combined with the triangle filtering (TriPto).

Figure 9 shows the relative performance w.r.t. Tri(10) for varying alpha. Note that for very small alpha also the iDim

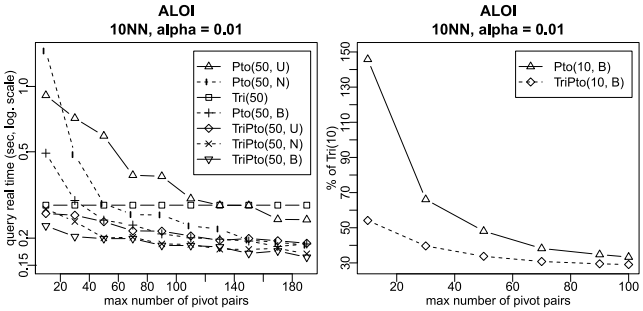

Figure 8: Number of pivot pairs in PPT (ALOI).

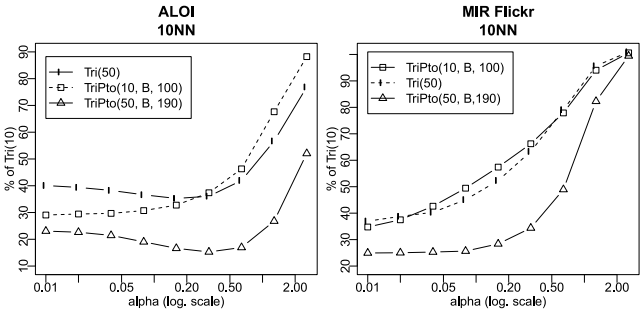

Figure 9: Impact of varying alpha on the relative performance.

is very small (≈ 2), so that the triangle filtering using 50 pivots is efficient enough and leaves a relatively smaller room for improvement. On the other hand, large alpha leads to high iDim (≈ 20), as well as smaller differences between triangular and Ptolemaic filtering – the space is hard to index, whichever filtering is used. Considering 50 pivots, the best speed-up of PPT is achieved for alpha=0.4 and ALOI (iDim ≈ 4), alpha=0.1 and MIR Flickr (iDim ≈ 3).

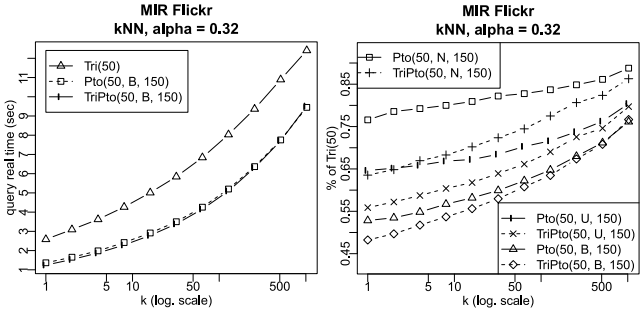

Figure 10: kNN queries on the MIR Flickr database.

The results for kNN queries on the MIR Flickr (see Figure 10) are similar to those for ALOI (see Figure 7), however, the largest speed-up of PPT was achieved for 1NN queries.

The results showing the varying number of pivot pairs on the MIR Flickr (see Figure 11) are also similar to those for ALOI (see Figure 8). Note that for a large enough number of pivot pairs (> 60) the Ptolemaic filtering works equally well using only 10 pivots as does triangle filtering with 50 pivots.

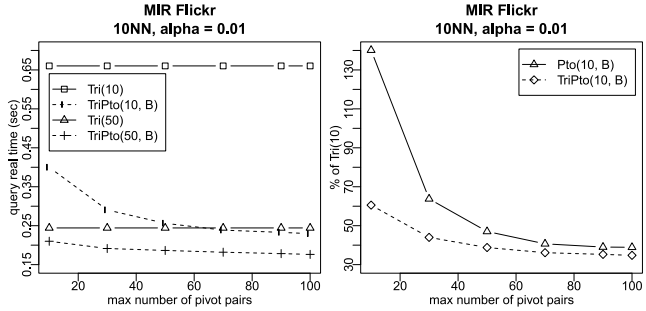

Figure 11: Number of pivot pairs in PPT (MIR Flickr).

6.3 Discussion

In summary, the PPT has proven to be a superior solution for indexing SQFD, beating the state-of-the-art metric index by responding almost four times as fast. When compared to the sequential scan, the speed-up factor of almost 300 is even more remarkable.

From another point of view, we have shown that the Ptolemaic filtering in PPT achieves the same filtering power as the triangle filtering in regular pivot tables using only 20% of the pivots. This suggests that PPT can be used as an economical solution that needs only small time and space to construct an index, while achieving the same query efficiency as a metric index that is five times as large and five times as slow to construct.

Moreover, our method uses a two-pivot online pivot selection technique where the best pair of pivots is heuristically chosen for every single query and database object pair. In the metric indexing area this is the "holy grail" of pivot selection techniques, as a good pivot should be either close to the query or close to the database object. Instead of selecting pivots from the database like the offline techniques do, the pivot pairs are picked from an pre-selected set of pivots. It seems that the effort usually spent on the offline pivot selection process does not play such an important role here, as the main pivot pair selection procedure is performed online. The number of candidate pairs grows quadratically with the number of pivots, so even a pivot set that is bad from the metric indexing point of view could provide good pivot pairs for Ptolemaic filtering. However, additional analysis of this hypothesis has to be done in the future.

7. CONCLUSIONS

In this paper we have applied the principles of Ptolemaic indexing to the signature quadratic form distance (SQFD), and found significant speed-up compared to the state-of-the-art metric indexing approach. Overall, our results have the immediate benefit of making similarity search more efficient for applications using the SQFD, but they are also important for Ptolemaic indexing in general. At present, the main family of Ptolemaic distances of practical importance is quadratic form distances (QFDs) [15]. Because QFDs are expensive to compute, the internal overhead of Ptolemaic indexing is not an obstacle. However, as has been recently shown [26], static QFDs can be mapped to Euclidean distance for the purpose of indexing, making Ptolemaic indexing seemingly unfeasible. In this paper we show that the

more expressive SQFDs are, in fact, equivalent to QFDs, except that their dynamic nature precludes this form of preprocessing. This, together with the increased performance over metric indexing, means that the matching of Ptolemaic indexing and SQFDs is a very natural and fruitful one.

Acknowledgments

This research has been supported by Czech Science Foundation projects GAČR 201/09/0683, 202/11/0968 (first and third author), by the Research Council of Norway project iAd (second author) and by the Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 686 (fourth author).

8. REFERENCES

[1] C. Beecks, J. Lokoč, T. Seidl, and T. Skopal. Indexing the signature quadratic form distance for efficient content-based multimedia retrieval. In *Proc. ACM International Conference on Multimedia Retrieval*, 2011.

[2] C. Beecks, M. S. Uysal, and T. Seidl. Signature quadratic form distances for content-based similarity. In *Proc. ACM International Conference on Multimedia*, pages 697–700, 2009.

[3] C. Beecks, M. S. Uysal, and T. Seidl. A comparative study of similarity measures for content-based multimedia retrieval. In *Proc. IEEE International Conference on Multimedia & Expo*, pages 1552–1557, 2010.

[4] C. Beecks, M. S. Uysal, and T. Seidl. Efficient k-nearest neighbor queries with the signature quadratic form distance. In *Proc. IEEE International Conference on Data Engineering Workshops*, pages 10 – 15, 2010.

[5] C. Beecks, M. S. Uysal, and T. Seidl. Signature quadratic form distance. In *Proc. ACM International Conference on Image and Video Retrieval*, pages 438–445, 2010.

[6] C. Beecks, M. S. Uysal, and T. Seidl. Similarity matrix compression for efficient signature quadratic form distance computation. In *Proc. International Conference on Similarity Search and Applications*, pages 109–114, 2010.

[7] S. E. Bratsberg and M. L. Hetland. Dynamic optimization of queries in pivot-based indexing. *Multimedia Tools and Applications*, 2010.

[8] E. Chávez, K. Figueroa, and G. Navarro. Effective proximity retrieval by ordering permutations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(9):1647–1658, 2008.

[9] E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroquín. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001.

[10] T. Deselaers, D. Keysers, and H. Ney. Features for image retrieval: an experimental comparison. *Information Retrieval*, 11(2):77–107, 2008.

[11] K. Figueroa, E. Chávez, G. Navarro, and R. Paredes. On the least cost for proximity searching in metric spaces. In *Proc. of WEA, LNCS 4007*. Springer, 2006.

[12] J.-M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders. The Amsterdam Library of Object Images. *International Journal of Computer Vision*, 61(1):103–112, 2005.

[13] J. Hafner, H. S. Sawhney, W. Equitz, M. Flickner, and W. Niblack. Efficient color histogram indexing for quadratic form distance functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:729–736, 1995.

[14] M. L. Hetland. The basic principles of metric indexing. In C. A. C. Coello, S. Dehuri, and S. Ghosh, editors, *Swarm Intelligence for Multi-objective Problems in Data Mining*, volume 242 of *Studies in Computational Intelligence*. Springer, 2009.

[15] M. L. Hetland. Ptolemaic indexing. `arXiv:0911.4384 [cs.DS]`, 2009.

[16] R. Hu, S. Rüger, D. Song, H. Liu, and Z. Huang. Dissimilarity measures for content-based image retrieval. In *Proc. IEEE International Conference on Multimedia & Expo*, pages 1365 –1368, 2008.

[17] M. J. Huiskes and M. S. Lew. The mir flickr retrieval evaluation. In *Proceedings of the 1st ACM International Conference on Multimedia Information Retrieval*, pages 39–43, 2008.

[18] W. K. Leow and R. Li. The analysis and applications of adaptive-binning color histograms. *Computer Vision and Image Understanding*, 94(1-3):67–91, 2004.

[19] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[20] M. L. Mico, J. Oncina, and E. Vidal. A new version of the nearest-neighbour approximating and eliminating search algorithm (aesa) with linear preprocessing time and memory requirements. *Pattern Recogn. Lett.*, 15(1):9–17, 1994.

[21] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[22] G. Navarro. Analyzing metric space indexes: What for? In *IEEE SISAP 2009*, pages 3–10, 2009.

[23] J. Puzicha, J. Buhmann, Y. Rubner, and C. Tomasi. Empirical evaluation of dissimilarity measures for color and texture. In *Proc. IEEE International Conference on Computer Vision*, volume 2, pages 1165 –1172, 1999.

[24] H. Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann, 2006.

[25] M. Skala. Counting distance permutations. *J. of Discrete Algorithms*, 7:49–61, March 2009.

[26] T. Skopal, T. Bartoš, and J. Lokoč. On (not) indexing quadratic form distance by metric access methods. In *Proc. Extending Database Technology (EDBT), ACM*, 2011.

[27] E. Vidal. New formulation and improvements of the nearest-neighbour approximating and eliminating search algorithm (AESA). *Pattern Recognition Letters*, 15(1):1–7, January 1994.

[28] P. Zezula, G. Amato, V. Dohnal, and M. Batko. *Similarity Search: The Metric Space Approach*. Springer, 2005.