Pacific Graphics 2011
Bing-Yu Chen, Jan Kautz, Tong-Yee Lee, and Ming C. Lin
(Guest Editors)

Volume 30 (2011), Number 7

Improving Performance and Accuracy of Local PCA

Václav Gassenbauer[1,2] Jaroslav Křivánek[3] Kadi Bouatouch[1] Christian Bouville[1] Mickaël Ribardière[1]

[1]IRISA / INRIA, Rennes
[2]Faculty of Electrical Engineering, Czech Technical University in Prague
[3]Charles University, Prague

Abstract

Local Principal Component Analysis (LPCA) is one of the popular techniques for dimensionality reduction and data compression of large data sets encountered in computer graphics. The LPCA algorithm is a variant of k-means clustering where the repetitive classification of high dimensional data points to their nearest cluster leads to long execution times. The focus of this paper is on improving the efficiency and accuracy of LPCA. We propose a novel SortCluster LPCA algorithm that significantly reduces the cost of the point-cluster classification stage, achieving a speed-up of up to 20. To improve the approximation accuracy, we investigate different initialization schemes for LPCA and find that the k-means++ algorithm [AV07] yields best results, however at a high computation cost. We show that similar ideas that lead to the efficiency of our SortCluster LPCA algorithm can be used to accelerate k-means++. The resulting initialization algorithm is faster than purely random seeding while producing substantially more accurate data approximation.

Categories and Subject Descriptors (according to ACM CCS): I.5.3 [Pattern Recognition]: Clustering—Algorithms; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1. Introduction

Precomputation-based and data-driven approaches have emerged in computer graphics as important tools for improving rendering performance and increasing image fidelity [Ram09, FH09]. Common to these techniques is the need to compress large data sets consisting of high-dimensional data points. One of the successful data compression approaches has been so called *local principal component analysis* (LPCA), also known as *clustered PCA* (CPCA) [KL97]. LPCA was shown to be effective for compression of transfer matrices in pre-computed radiance transfer [SKS02, SHHS03, HR10] or of Bidirectional texture function (BTF) data sets [FH09, MMK03]. However, slow performance of the data compression (often several hours for a single data set) is a serious bottleneck in the data processing pipeline. For instance, though PRT may be a useful tool for scene relighting, the long pre-computation times (including the data compression) may hinder its practical applicability. Quite surprisingly, performance of data compression has been largely unaddressed in previous computer graphics research. Huang and Ramamoorthi [HR10] do accelerate the transfer matrix pre-computation, however, they still rely on the slow LPCA algorithm to compress the resulting data set.

In this paper we improve both performance and data approximation accuracy of the LPCA algorithm. Being a variant of *k*-means clustering, the bottleneck of LPCA consists in the repeated classification of data points to the nearest clusters. Due to the high data dimensionality, simple approaches such as spatial data indexes [PM99, KMN*02] are ineffective at accelerating this process. Nonetheless, we found that a significant speed-up can be achieved by taking advantage of the information gathered as the algorithm progresses to eliminate some point-cluster distance calculations that provably cannot change current point-cluster assignment. Though this simple and effective idea (known as SortMeans) has been previously used to accelerate *k*-means clustering [Phi02, Elk03], to our knowledge our work is the first to investigate its use in computer graphics. Our main contribution, however, consists in extending the SortMeans algorithm [Phi02] from the simple *k*-means problem (where each cluster is represented by one mean point) to the LPCA problem (where a cluster is represented by an affine subspace). In addition, we improve the approximation accuracy of LPCA by seeding the clusters using the *k*-means++ algorithm [AV07]. To minimize the performance impact of this advanced initialization, we apply a variant of SortMeans to cut some of the unnecessary distance calculations. We

present extensive measurements of performance and approximation accuracy of the presented algorithms for four radiance transfer matrices (similar to [HR10]) and three BTF data sets from the University of Bonn database. The speed-up of classification is 5 to 20 for the coherent transfer matrices and 1.2 to 1.6 for the complex BTF data. We show that the k-means++ algorithm substantially improves the approximation accuracy for the investigated data sets. Our results could be useful both in and outside of computer graphics.

2. Related Work

LPCA in computer graphics. Local principal component analysis (LPCA) was devised by Kambhatla and Leen [KL97] in the machine learning community. It was introduced to computer graphics (under the name clustered PCA) by Sloan et al. [SHHS03] who used it for the compression of low-frequency light transfer matrices in precomputed radiance transfer [SKS02]. LPCA was later used for all-frequency light transfer on glossy surfaces [LSSS04, MSRB07, XJF*08, HR10]. Mahajan et al. [MSRB07] adapt the clusters to optimize a rendering performance metric, and use a hierarchical splitting algorithm instead of k-means-based clustering. Huang and Ramamoorthi [HR10] achieve a substantial speed-up of LPCA compression by reducing the size of the input data set. Our work, on the other hand, focuses on the acceleration of the LPCA algorithm itself and is complementary to their work. LPCA is also one of the most widely used algorithms for BTF data compression because it provides both high compression ratio and low approximation error [FH09, MMK03].

Acceleration of k-means and related methods. Despite a wide range of LPCA applications, we are not aware of any work on improving the algorithm's performance. Most of the previous research has focused on the acceleration of the simpler k-means clustering. The idea is to reduce the number of distance calculations when classifying data points to their nearest cluster. Some works organize the data points in a spatial data structure such as kd-tree [PM99, KMN*02, Moo00]. With the exception of [Moo00] these approaches are effective only for low-dimensional data points. Other works are based on using the triangle inequality to avoid unnecessary point-cluster distance calculation. Works of Hodgson [Hod88] and Phillips [Phi02] propose the use of an upper bound on the distance between a data point and a cluster. Our algorithm is based on their ideas, specifically on Phillips' SortMeans algorithm. We generalize the algorithm to the more complex LPCA problem. More recent works [Elk03, Ham10] add the use of a lower bound on the point-cluster distance to make k-means clustering even faster. The computation of the lower bound is, however, not trivial in the case of LPCA, and we do not use it.

Initialization of k-means. The k-means algorithm is prone to getting stuck in local minima of the objective function thereby producing suboptimal clustering results. Random restarts [KMN*02] partially alleviate the problem at a high computational cost. More advanced initialization algorithms attempt to seed the centroids close to the target clustering. The farthest-first heuristic [HS85] tends to place the initial centroids on the outer hull of the space subtended by the data points. The state-of-the-art k-means++ algorithm [AV07] achieves better results by randomizing this process. In spite of using a more advanced heuristic for choosing the first two centroids, the variant of k-means++ described by Ostrovsky et al. [ORSS06] did not significantly improve the results in our experiments.

3. Preliminaries

3.1. Problem definition

We start by formally defining the data approximation problem that we address in this work. Let the input data set $\mathcal{X} \subset V^d$ be a subset of a d-dimensional linear space V^d with the dot product $\langle \cdot, \cdot \rangle$, and let k and l be integers. The goal is to choose k affine subspaces $\mathbf{a}_1, \ldots, \mathbf{a}_k \subset V^d$ of dimension up to l, $l < d$ such that the following objective function is minimized:

$$\phi = \sum_{\mathbf{x} \in \mathcal{X}} \min_{j=1}^{k} d(\mathbf{x}, \mathbf{a}_j),$$

where $d(\mathbf{x}, \mathbf{a})$ is the distance of data point $\mathbf{x} \in \mathcal{X}$ from affine subspace $\mathbf{a}$. The assignment of $\mathbf{x}$ to their nearest affine subspaces induces a clustering (partition) of the data set $\mathcal{X}$.

The distance $d(\mathbf{x}, \mathbf{a})$ is defined as [KL97]:

$$d(\mathbf{x}, \mathbf{a}) = \|(\mathbf{x} - \mu(\mathbf{a})) - \mathbf{x}_s\|, \tag{1}$$

where $\mu(\mathbf{a})$ is the origin of $\mathbf{a}$, $\mathrm{dir}(\mathbf{a})$ is its basis, and $\mathbf{x}_s$ is the orthogonal projection of $(\mathbf{x} - \mu(\mathbf{a}))$ onto $\mathbf{a}$:

$$\mathbf{x}_s = \sum_{i=1}^{l} \langle \mathbf{x} - \mu(\mathbf{a}), \mathrm{dir}_i(\mathbf{a}) \rangle \cdot \mathrm{dir}_i(\mathbf{a}). \tag{2}$$

3.2. Local Principal Component Analysis (LPCA)

Finding the optimal solution of the above problem is NP-hard, even just for two clusters of zero dimension [DFK*04]. To find an approximate solution, Kambhatla and Leen [KL97] proposed the *local principal component analysis* (LPCA) algorithm as a modification of the classical k-means algorithm (which solves the problem defined above in affine subspaces with dimension of $l = 0$). The LPCA algorithm proceeds as follows. In the first step, initial centroids for the clusters are selected. Each data point $\mathbf{x} \in \mathcal{X}$ is assigned to the nearest (in terms of Euclidean distance) cluster. In the second step, affine subspaces are found within each cluster and are used as a low-dimensional approximation of the data points assigned to it. Both steps are repeated several times. Then the dimension of the affine subspaces is increased and the whole previous computation is performed

again until the desired dimension of the subspaces is reached [SHHS03].

Thanks to its simplicity, LPCA is one of the most widely used algorithms in computer graphics for approximation of high-dimensional signals. However, LPCA is computationally demanding when used for classification of high-dimensional points into a high number of clusters. The inefficiency comes from the fact that no information is passed from one stage to another: the point classification computes distances to *all* PCA subspaces for each data point. Our goal is to avoid unnecessary distance calculations by exploiting the information computed in the previous iteration. Our accelerated algorithm produces exactly the same clustering as the original LPCA algorithm, but more quickly.

3.3. SortMeans: Acceleration of k-means Clustering

Our algorithm is based on the SortMeans algorithm proposed by Phillips [Phi02] for accelerating the k-means problem, which is an instance of the data approximation problem defined above, where the dimension of the affine subspaces is $l = 0$. Phillips uses the triangular inequality to avoid unnecessary point-cluster distance computations in the classification stage. For an arbitrary point $\mathbf{x} \in \mathcal{X}$ and two clusters c_a and c_b represented by their centroids $\mu(\mathrm{c}_a)$ and $\mu(\mathrm{c}_b)$, respectively, the triangle inequality says

$$d(\mu(\mathrm{c}_a),\mu(\mathrm{c}_b)) \leq d(\mathbf{x},\mu(\mathrm{c}_a)) + d(\mathbf{x},\mu(\mathrm{c}_b))$$

or, equivalently

$$d(\mathbf{x},\mu(\mathrm{c}_b)) \geq d(\mu(\mathrm{c}_a),\mu(\mathrm{c}_b)) - d(\mathbf{x},\mu(\mathrm{c}_a)),$$

where $d(\cdot,\cdot)$ is the Euclidean distance between two data points in V^d. Therefore if one knows that $d(\mu(\mathrm{c}_a),\mu(\mathrm{c}_b)) \geq 2d(\mathbf{x},\mu(\mathrm{c}_a))$, then $d(\mathbf{x},\mu(\mathrm{c}_b)) \geq d(\mathbf{x},\mu(\mathrm{c}_a))$, e.g. the distance of $\mathbf{x}$ to c_b cannot be lower than the one to c_a and the computation of $d(\mathbf{x},\mu(\mathrm{c}_b))$ can be safely avoided.

We briefly summarize the SortMeans algorithm [Phi02] built around this idea. At the beginning of each iteration SortMeans precomputes two $k \times k$ matrices (k is the number of clusters), a distance matrix $\mathbf{D}$ and a permutation matrix $\mathbf{M}$. Elements of $\mathbf{D}$ are defined as $\mathbf{D}(i,j) = d(\mu(\mathrm{c}_i),\mu(\mathrm{c}_j))$. Rows of $\mathbf{M}$ represent permutations on the set of cluster indices such that $\mu(\mathrm{c}_{\mathbf{M}(i,j)})$ is the j-th nearest centroid from $\mu(\mathrm{c}_i)$.

For each data point $\mathbf{x} \in \mathcal{X}$, which was assigned to c_i in the previous iteration, the algorithm iterates over all other clusters, keeping the minimal distance found so far, denoted $d_{\min}$ (which is initially set to $d(\mathbf{x},\mu(\mathrm{c}_i))$). Clusters are iteratively processed in increasing order of their distances from $\mu(\mathrm{c}_i)$ using $\mathbf{M}$. If $\mathbf{D}(i,\mathbf{M}(i,j)) \geq d(\mathbf{x},\mu(\mathrm{c}_i)) + d_{\min}$ the nearest cluster has been already found and the iteration is stopped. Otherwise the actual distance $d(\mathbf{x},\mu(\mathrm{c}_{\mathbf{M}(i,j)}))$ is computed. If $d(\mathbf{x},\mu(\mathrm{c}_{\mathbf{M}(i,j)})) < d_{\min}$, $\mathrm{c}_{\mathbf{M}(i,j)}$ becomes the new nearest cluster and $d_{\min} = d(\mathbf{x},\mu(\mathrm{c}_{\mathbf{M}(i,j)}))$ is the new best minimal distance. The computation above is then repeated for the next nearest cluster $\mathrm{c}_{\mathbf{M}(i,j+1)}$ (if there is any, i.e. if $j < k$).

The SortMeans algorithm was used for accelerating k-means clustering. In the next section we use it as a basic building block for developing our new SortClusters LPCA algorithm for accelerating the more general LPCA problem.

4. SortClusters LPCA: Acceleration of General LPCA

In this section we develop our SortClusters LPCA algorithm as a generalization of the SortMeans algorithm [Phi02] described above. Since clusters in LPCA are defined by affine subspaces rather than just centroids, we need a definition of distance between two subspaces. The subspace distance is then used to develop a generalized triangle inequality for a data point and two affine subspaces. Such a triangle inequality is in turn used in our SortClusters LPCA algorithm to avoid unnecessary point-subspace distance calculations.

4.1. Distance Between Affine Subspaces and The Generalized Triangle Inequality

Let $\mathbf{a}_a, \mathbf{a}_b$ be arbitrary non-empty affine subspaces in V^d. The distance between $\mathbf{a}_a$ and $\mathbf{a}_b$ is defined as [DK92]:

$$d(\mathbf{a}_a,\mathbf{a}_b) = \inf\{\|\mathbf{p}-\mathbf{q}\|;\ \mathbf{p} \in \mathbf{a}_a, \mathbf{q} \in \mathbf{a}_b\}. \qquad (3)$$

Intuitively, the distance between two affine subspaces is equal to the length of the shortest line that is orthogonal to both subspaces.

Lemma: The following *generalized triangle inequality* holds for an arbitrary point $\mathbf{x} \in V^d$ and arbitrary non-empty affine subspaces $\mathbf{a}_a, \mathbf{a}_b \subset V^d$:

$$d(\mathbf{a}_a,\mathbf{a}_b) \leq d(\mathbf{x},\mathbf{a}_a) + d(\mathbf{x},\mathbf{a}_b). \qquad (4)$$

Proof. For an arbitrary point $\mathbf{x} \in V^d$ using (3) we have

$$\begin{aligned} d(\mathbf{a}_a,\mathbf{a}_b) &= \inf\{\|\mathbf{p}-\mathbf{q}\|;\ \mathbf{p} \in \mathbf{a}_a, \mathbf{q} \in \mathbf{a}_b\} \\ &\leq \inf\{\|\mathbf{p}-\mathbf{x}\| + \|\mathbf{x}-\mathbf{q}\|;\ \mathbf{p} \in \mathbf{a}_a, \mathbf{q} \in \mathbf{a}_b\} \\ &= \inf\{\|\mathbf{p}-\mathbf{x}\|;\ \mathbf{p} \in \mathbf{a}_a\} + \inf\{\|\mathbf{x}-\mathbf{q}\|;\ \mathbf{q} \in \mathbf{a}_b\} \\ &= d(\mathbf{x},\mathbf{a}_a) + d(\mathbf{x},\mathbf{a}_b). \end{aligned}$$

4.2. The SortClusters LPCA Algorithm

We use the generalized triangle inequality (4) to reduce point-subspace computations performed during the classification stage of LPCA in a similar way as in the SortMeans algorithm. The structure of our SortClusters LPCA (SC-LPCA) algorithm is similar to the SortMeans algorithm. However, instead of computing distances with respect to the clusters' centroids, we compute distances with respect to affine subspaces. Following the SortMeans algorithm we construct the distance and permutation matrices, $\mathbf{D}$ and $\mathbf{M}$, at the beginning of the classification stage. Elements of $\mathbf{D}$ are set to distances between pairs of affine subspaces. After

that, we loop over all data points and for each we find the nearest affine subspace. The pseudo-code for searching the nearest affine subspace for a given point is given in Algorithm 1. The algorithm takes a data point $\mathbf{x} \in \mathcal{X}$, which was assigned to the i-th affine subspace in the previous iteration, and returns the index of the current nearest affine subspace.

Algorithm 1 Point-subspace classification

input : $\mathbf{x}$ ... data point to be classified
i ... index of the affine subspace to which $\mathbf{x}$ was assigned to in the previous iteration
output : index of the new nearest affine subspace

$d_{\min} \leftarrow d(\mathbf{x}, \mathbf{a}_i)$
$i_{\min} \leftarrow i$
for $j \leftarrow 2$ ***to*** k **do**
  if $(\mathbf{D}(i, \mathbf{M}(i,j)) \geq d(\mathbf{x}, \mathbf{a}_i) + d_{\min})$ **then**
    break
  end
  dist $\leftarrow d(\mathbf{x}, \mathbf{a}_{\mathbf{M}(i,j)})$
  if (dist $< d_{\min}$) **then**
    $d_{\min} \leftarrow$ dist
    $i_{\min} \leftarrow \mathbf{M}(i,j)$
  end
end
return $i_{\min}$

4.3. Efficient Evaluation of Inter-Subspace Distance

Determination of the distance between two subspaces defined by (3) can be converted to the computation of the distance between a point and a linear subspace using the following Lemma [DK92]: Let $\mathbf{a}_a, \mathbf{a}_b \subset V^d$ be non-empty affine subspaces. Then for arbitrary points $\mathbf{p} \in \mathbf{a}_a$ and $\mathbf{q} \in \mathbf{a}_b$ the distance between $\mathbf{a}_a$ and $\mathbf{a}_b$ is equal to

$$d(\mathbf{a}_a, \mathbf{a}_b) = d(\mathbf{p} - \mathbf{q},\ \mathrm{dir}(\mathbf{a}_a) \cup \mathrm{dir}(\mathbf{a}_b)). \tag{5}$$

The above Lemma gives us a recipe to compute the distance between non-empty affine subspaces $\mathbf{a}_a, \mathbf{a}_b \in V^d$. We use Equation (1) to solve Equation (5). We put $\mathbf{p} - \mathbf{q} = \mu(\mathbf{a}_a) - \mu(\mathbf{a}_b)$ as $\mathbf{x}$ and use the affine subspace with origin of $\mathbf{0}$ and basis of $\mathrm{dir}(\mathbf{a}_a) \cup \mathrm{dir}(\mathbf{a}_b)$ as $\mathbf{a}$. To evaluate Equation (1) we need to compute $\mathbf{x}_s$. It requires finding an orthonormal basis $\mathrm{dir}(\mathbf{a}_a) \cup \mathrm{dir}(\mathbf{a}_b)$ followed by the projection of $\mu(\mathbf{a}_a) - \mu(\mathbf{a}_b)$ onto this basis using (2). Orthonormalization is, however, a computationally demanding task. Instead, we can compute $\mathbf{x}_s$ directly without explicit construction of the orthonormal basis as shown below. Please, see [DK92] for more details.

Let m and n be the dimension of $\mathrm{dir}(\mathbf{a}_a)$ and $\mathrm{dir}(\mathbf{a}_b)$, respectively. To compute $\mathbf{x}_s$ we need to find a solution $\mathbf{z} = (z_1, \ldots, z_m, z_{m+1}, \ldots, z_{m+n})^T \in \mathbb{R}^{m+n}$ of the following linear system and set $\mathbf{x}_s = \mathbf{K} \cdot \mathbf{z}$:

$$\mathbf{G}(\mathbf{K}) \cdot \mathbf{z} = \mathbf{K}^T \cdot (\mu(\mathbf{a}_a) - \mu(\mathbf{a}_b)),$$

where $\mathbf{K} = [\mathrm{dir}_1(\mathbf{a}_a), \ldots, \mathrm{dir}_m(\mathbf{a}_a), \mathrm{dir}_1(\mathbf{a}_b), \ldots, \mathrm{dir}_n(\mathbf{a}_b)]$, and $\mathbf{G}(\mathbf{K})$ is the Gram matrix of all inner products of $\mathbf{K}$. We solve the linear system through the Cholesky decomposition of $\mathbf{G}(\mathbf{K})$, since $\mathbf{G}(\mathbf{K})$ is positive semi-definite. Finally, we compute the distance between $\mathbf{a}_a$ and $\mathbf{a}_b$ as $d(\mathbf{a}_a, \mathbf{a}_b) = \|\mu(\mathbf{a}_b) - \mu(\mathbf{a}_a) - \mathbf{x}_s\|$.

5. Cluster Initialization

The very first step of the LPCA (and k-means) algorithm is to choose k initial cluster centroids among the data points. We found that the initialization has a significant impact both on the approximation accuracy and—quite surprisingly—on the performance of our SortClusters LPCA algorithm. We have investigated random initialization with uniform probability [SHHS03], random initialization based on distance sums inpired by [HPB07], and the state-of-the-art k-means++ algorithm [AV07].

Distance sums-based initialization. For each data point $\mathbf{x}_i$, we compute the sum of squared distances to other data points, $\alpha_i = \sum_{\mathbf{x} \in \mathcal{X}} \|\mathbf{x} - \mathbf{x}_i\|^2$, and use it as a probability distribution for picking the k initial cluster centroids.

***k*-means++.** The first centroid is chosen from among the data points at random with uniform probability. Distance to all other data points is then used as a probability distribution for choosing the second centroid. When a new centroid is chosen, all remaining data points are classified to their nearest centroid. The updated distance to the nearest centroid for each data point is then used as a probability distribution for choosing another centroid, and so on until we have k initial centroids.

The k-means++ initialization produces by far the most accurate data approximation, however the computation cost can be high (on a par with one iteration of k-means). Below, we describe our new SortMeans++ algorithm which uses the ideas on which the SortMeans algorithm is built to accelerate the k-means++ initialization.

5.1. SortMeans++: Accelerated *k*-means++

The structure of our SortMeans++ algorithm is similar to k-means++ but we eliminate some of the point-cluster distance calculations when a new cluster is created. For each data point, the algorithm maintains the distance to its nearest cluster in point $\mathbf{N}$ and the index of its nearest cluster in point $\mathbf{I}$. Similar to k-means++ we start by choosing the first centroid $\mathbf{x}_{\text{new}} \in \mathcal{X}$ at random with uniform probability. Point $\mathbf{N}$ is initialized with distances to this centroid, i.e. $\mathbf{N}(i) \leftarrow \|\mathbf{x}_{\text{new}} - \mathbf{x}_i\|$, and point $\mathbf{I}$ is initialized with all ones (index of the only existing cluster). The following steps are then repeated until the desired number of clusters is created.

1. Pick a new centroid $\mathbf{x}_{\text{new}} \in \mathcal{X}$ at random with probability proportional to $\mathbf{N}$, and create a new cluster c_{new} with $\mu(\mathrm{c}_{\text{new}}) = \mathbf{x}_{\text{new}}$.

(a) Horse (b) Dragon (c) Happy Buddha (d) Walt Disney (e) 16 x Error Image

Figure 1: Scenes used in our experiments. We tested our SortClusters LPCA (SC-LPCA) for compression of transfer matrices for these scenes. The models in 1a, 1b, and 1c are made of glossy materials represented using Phong's BRDF with exponent from 10 to 30. The model in 1d is represented using Ward's BRDF with the roughness of 0.1. Compared to LPCA, we achieve a 5× to 20× speed-up using SC-LPCA, while providing identical output. For the sake of completeness, Figure 1e shows a 16× amplified difference between renderings of Walt Disney Hall using the original and compressed transfer matrix.

2. Compute the distance from the new cluster to all existing clusters, $d_{\text{new}}(i) \leftarrow \|\mu(\mathrm{c}_{\text{new}}) - \mu(c_i))\|$.
3. Loop over all data points $\mathbf{x}_i \in \mathcal{X}$ and check if the new point c_{new} is closer than their currently assigned cluster. If $d_{\text{new}}(\mathbf{I}(i)) \geq 2\mathbf{N}(i)$, then the assignment of $\mathbf{x}_i$ cannot change and we can safely skip the calculation of the point-cluster distance $\|\mu(\mathrm{c}_{\text{new}}) - \mathbf{x}_i\|$. Otherwise, we calculate the distance and if it is smaller than $\mathbf{N}(i)$, we update $\mathbf{N}(i)$ and $\mathbf{I}(i)$. After that, we proceed to the next data point.

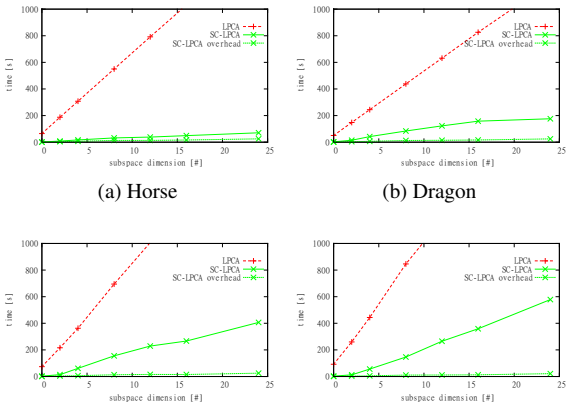
6. Results

We compare our algorithm (SortCluster LPCA) with the original LPCA [KL97] for compression of PRT and BTF datasets. Both algorithms were implemented in C++ using Intel MKL library for matrix computations. All measurements were performed on a PC with Intel Xeon W3540, 2.93 GHz and 14GB RAM. We use all 4 CPUs exploiting the parallelism of MKL routines yielding the CPU load of about 90% (as reported by Windows 7 Task Manager).

6.1. Compression of PRT data

We tested our algorithms on PRT data matrix precomputed for three different scenes shown in Figure 1. According to [NRH03] we use a raw lighting basis of cubemap pixel lights when computing the transfer matrix. The cubemap resolution is $6 \times 32 \times 32$. We account for self-shadowing effects only and we eliminate visibility alias by applying a 4×4 super-sampling to estimate the light transfer for each cubemap pixel. We used $k = 256$ clusters for the matrix compression, and up to $l = 24$ basis vectors for each cluster. According to Sloan et al. [SHHS03] we perform several iterations of LPCA before we increase the dimension of PCA clusters. More specifically, we use the following iterative scheme (0:15), (2:10), (4:7), (8:5), (12:4), (16:2), (24:1), where the first number is the maximum dimension of PCA subspaces and the second one is the number of iterations done for the same dimension. The plots in this section report the time for point classification only and do not include the time for computing the PCA approximation of the data points within clusters.

Scalability with the PCA dimension. Breakdown of the average computation times for one iteration of the classification using our SortClusters LPCA (SC-LPCA) and original LPCA as a function of the dimension of affine subspaces is shown in Figures 2. The average number of distances computed for both algorithms is shown in Figure 3. We can see that our SC-LPCA scales well with the dimension of affine subspaces. We achieve more than 20× speed up for the Horse scene while the speed-up for the more com-



(a) Horse (b) Dragon

(c) Happy Buddha (d) Walt Disney Hall

Figure 2: Breakdown of the average computation time per one iteration of our SC-LPCA and original LPCA. Note that the curves for our SC-LPCA contain both the overhead and the time required for the classification itself. The overhead of the SC-LPCA consist of the computation of distance and permutation matrices **D** and **M**, respectively, and is very small even for a high number of PCA basis vectors.

plex models (Dragon, Buddha, Disney) is about 6. The reason for higher speed-up for the Horse scene is presumably the higher degree of light transport coherence in this relatively simple scene. Note that the output of our SC-LPCA is exactly the same as provided by the original LPCA algorithm.

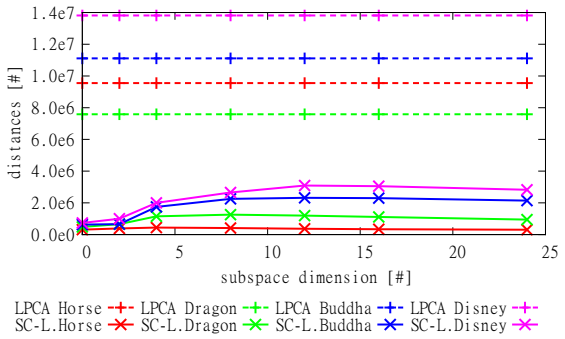


Figure 3: Average number of distances evaluated by the compared algorithms per one iteration of the classification.

Scalability with the number of clusters. Breakdown of the classification times in dependence on the total number of clusters is shown in Figure 4. We use the Dragon scene for this comparison. The computation time of our SC-LPCA stays roughly constant with the number of clusters, meaning that our SC-LPCA is well suited for applications where clustering to a high number of clusters is required. The computation time of the classical LPCA, on the other hand, increases linearly with the number of clusters. Interestingly, the average time for an SC-LPCA iteration for subspace dimension of 24 *decreases* with the number of clusters, in spite of the fact that the SC-LPCA overhead (computation of the distance matrix **D**) increases quadratically.

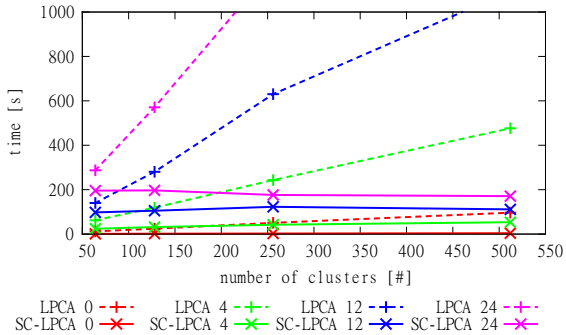
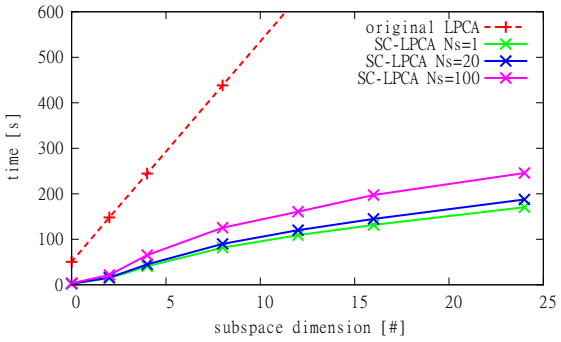
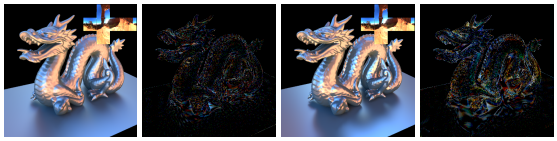


Figure 4: Scalability with the number of clusters for the Dragon scene. The plots show the average time spent on one iteration of the classification for subspaces' dimensions of 0, 4, 12, and 24.

Scalability with the Phong's lobe exponent. Unlike the original LPCA, the performance of our SC-LPCA does depend on the data set itself. Figure 5 illustrates the scalability of SC-LPCA with different Phong's lobe exponent. We can see only a small decrease in performance for a high Phong's lobe exponent, meaning that our algorithms is applicable for a wide range of materials.





(a) $N_s = 20$ (b) 16 x Error (c) $N_s = 100$ (d) 16 x Error

Figure 5: Scalability with the Phong's lobe exponent for the Dragon scene. The top image shows the average time spent on one iteration of the classification. The bottom images show visual examples of the scene with dragon made of glossy material represented using Phong's BRDF with the lobe exponent of $N_s = 20$ and $N_s = 100$ in Figures 5a and 5c, respectively. Images in Figures 5b and 5d shows the corresponding 16× difference images between renderings of the scenes using original and LPCA-compressed light transfer matrices.

Latency and accuracy of initialization algorithms. Our previous results only focused on speeding-up LPCA without improving its accuracy. Here we investigate different strategies for initial centroid seeding and their impact on accuracy and performance. The results for the Dragon scene are summarized in Figure 6. We perform 50 iterations of the original SortMeans algorithm [Phi02] after clusters' initialization and plot averages from 5 different runs. The naïve algorithm that select centroids purely at random [SHHS03] is prone to getting stuck in a local minimum. The *k*-means++ algorithm [AV07] provides a much lower approximation error at the cost of high initialization latency. While maintaining the approximation quality, our SortMeans++ algorithm decreases the start-up latency of the original *k*-means++ 6× to a value that is even less than the latency for purely random initialization (because SortMeans++ initialization produces a valid point-cluster classification).

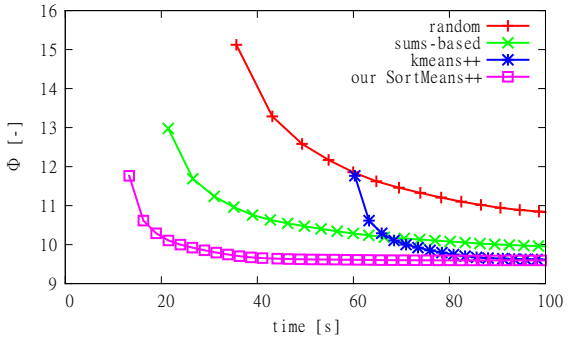


Figure 6: Latency and accuracy of initialization algorithms. Note that our algorithm SortMeans++ maintains the quality of approximation provided by the original k-means++ while substantially decreasing the start-up latency of the point-cluster assignment.

In addition to producing a better clustering, a good initial seeding also improves the performance of our SC-LPCA in subsequent iterations. The total computation time spent by SC-LPCA over all iterations using our SortMeans++ for the initial seeding was 1700s with resulting error $\phi = 0.174$. On the other hand, when we initialize the centroids randomly, the total computation time is 1780s and the error increases to $\phi = 0.224$.

Overall statistics. Table 1 lists the statistics about the precomputation of the PRT matrices and the overall computation time required for their compression. Note that the most computationally demanding part is the classification of high dimensional data points to affine subspaces, which is the focus of our work. Evaluating PCA approximations within clusters is only a minor part of the total compression time for our data sets.

scene	vertices [#]	PRT [s]	Classification [s] LPCA	SC-LPCA	speedup	PCA [s]	ϕ [-]
Horse	67.6k	22.5	13 700	674	20.3	146	0.029
Dragon	57.5k	25.5	10 900	1700	6.38	155	0.174
Buddha	85.2k	31.6	16 700	3 020	5.55	170	0.316
Disney	106.3k	46.5	20 900	4 080	5.12	170	0.394

Table 1: Summary results and timings for the example scenes. The columns list the total number of vertices in the scene, transfer matrix computation time (PRT) and the total classification time using original LPCA and our SC-LPCA algorithm. The rightmost two columns shows the time spent on the PCA approximation evaluation and the value of the objective function ϕ. Transfer matrices were computed on a Geforce GTX 580 GPU, while the Classification and PCA computation we performed on 4 CPU cores.

6.2. BTF compression

To investigate the efficiency of our algorithm on more complex data sets, we ran our algorithm on three measured BTFs (shown in Figure 7) from the BTF data provided by University of Bonn, http://btf.cs.uni-bonn.de. For each material the total 81×81 images were taken from different directions of the camera and light source. The images have a resolution of 256×256 pixels.

We run the LPCA in the apparent BRDF arrangement [FH09] (i.e. one data point corresponds to an image of the apparent BRDF). For this arrangement the LPCA provides an approximation of higher quality than for standard arrangement, while maintaining the same compression ratio [MMK03]. The total number of clusters and the maximum dimension of the affine subspaces was set to 32 and 8, respectively, in accordance with Müller [MMK03].

Breakdown of the classification times using the SC-LPCA and original LPCA for all tested materials is shown in Figure 8. For the proposte BTF the speed-up of the SC-LPCA is about 1.6. For the wallpaper and wool materials, which exhibit different reflectance characteristics in all sampled dimensions, the speed up of the SC-LPCA is about 1.2.

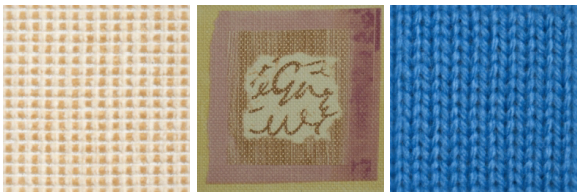


Figure 7: BTF materials used in our for testing our algorithm, proposte, wallpaper, and wool BTFs.

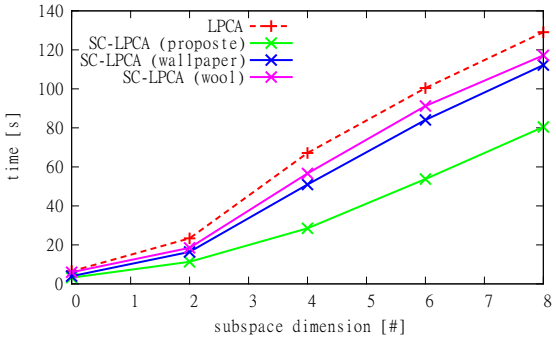


Figure 8: Breakdown of the computation for different BTF. Computation time for the original LPCA algorithm is independent of the BTF material. On the other hand our algorithm performs better for simple materials with a regular spatial/angular structure.

7. Conclusion

We present an accelerated and more accurate local PCA (LPCA) algorithm for compact approximation of large matrices. The improved performance is due to the significant reduction of point-cluster distance calculations in the classification stage of the algorithm. The accuracy is achieved through improved seeding of the initial cluster centroids. Our measurements on computer graphics data sets show a speed-up of 5 to 20 for radiance transfer matrices, though the speed-up is lower for complex and high-dimensional BTF data sets. Devising new algorithms for accurate and efficient compression of these more complex data set is an exciting avenue for future work. In the shorter term, our fast LPCA could be combined with a GPU implementation of Huang and Ramamoorthi's algorithm [HR10] to obtain a near-interactive pre-computation and compression of PRT data sets. We want to investigate the efficiency of the proposed approach on other computer graphics data sets, such as local light transport [HPB06], and to achieve further speed-up by performing the clustering in wavelet domain.

Acknowledgements

We are thankful to P. Zlatoš for his feedback on properties of affine subspaces. This work was supported by the Marie Curie Fellowship PIOF-GA-2008-221716 within the 7th European Community Framework Programme and by the Ministry of Education, Youth and Sports of the Czech Republic under the research program LC-06008.

References

[AV07] ARTHUR D., VASSILVITSKII S.: *k*-means++: the advantages of careful seeding. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (2007), Society for Industrial and Applied Mathematics, pp. 1027–1035.

[DFK*04] DRINEAS P., FRIEZE A. M., KANNAN R., VEMPALA S., VINAY V.: Clustering large graphs via the singular value decomposition. *Machine Learning 56* (2004), 9–33.

[DK92] DUPRE A. M., KASS S.: Distance and parallelism between flats in R^n. *Linear Algebra and its Applications 171* (July 1992), 99–107.

[Elk03] ELKAN C.: Using the Triangle Inequality to Accelerate K-Means.

[FH09] FILIP J., HAINDL M.: Bidirectional texture function modeling: A state of the art survey. *IEEE Trans. Pattern Anal. Mach. Intell. 31* (November 2009), 1921–1940.

[Ham10] HAMERLY G.: Making k-means even faster. In *SIAM International Conference on Data Mining* (2010), pp. 130–140.

[Hod88] HODGSON M. E.: Reducing the computational requirements of the minimum-distance classifier. *Remote Sensing of Environment 25*, 1 (1988), 117 – 128.

[HPB06] HAŠAN M., PELLACINI F., BALA K.: Direct-to-indirect transfer for cinematic relighting. *ACM Trans. Graph. 25* (July 2006), 1089–1097.

[HPB07] HAŠAN M., PELLACINI F., BALA K.: Matrix row-column sampling for the many-light problem. *ACM Trans. Graph. 26* (July 2007).

[HR10] HUANG F.-C., RAMAMOORTHI R.: Sparsely precomputing the light transport matrix for real-time rendering. *Computer Graphics Forum (EGSR 2010) 29*, 4 (2010), 1335–1345.

[HS85] HOCHBAUM D. S., SHMOYS D. B.: A Best Possible Heuristic for the k-Center Problem. *MATHEMATICS OF OPERATIONS RESEARCH 10*, 2 (May 1985), 180–184.

[KL97] KAMBHATLA N., LEEN T. K.: Dimension reduction by local principal component analysis. *Neural Comput. 9* (October 1997), 1493–1516.

[KMN*02] KANUNGO T., MOUNT D. M., NETANYAHU N. S., PIATKO C. D., SILVERMAN R., WU A. Y.: An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 7 (2002), 881–892.

[LSSS04] LIU X., SLOAN P.-P., SHUM H.-Y., SNYDER J.: All-frequency precomputed radiance transfer for glossy objects. *Computer Graphics Forum (EGSR 2004)* (2004), 337Ű344.

[MMK03] MÜLLER G., MESETH J., KLEIN R.: Compression and real-time rendering of measured BTFs using local PCA. In *Vision, Modeling and Visualisation 2003* (Nov. 2003), Akademische Verlagsgesellschaft Aka GmbH, Berlin, pp. 271–280.

[Moo00] MOORE A. W.: The anchors hierarchy: Using the triangle inequality to survive high dimensional data. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence* (San Francisco, CA, USA, 2000), UAI '00, Morgan Kaufmann Publishers Inc., pp. 397–405.

[MSRB07] MAHAJAN D., SHLIZERMAN I. K., RAMAMOORTHI R., BELHUMEUR P.: A theory of locally low dimensional light transport. *ACM Trans. Graph. 26* (July 2007).

[NRH03] NG R., RAMAMOORTHI R., HANRAHAN P.: All-frequency shadows using non-linear wavelet lighting approximation. *ACM Trans. Graph. 22* (July 2003), 376–381.

[ORSS06] OSTROVSKY R., RABANI Y., SCHULMAN L. J., SWAMY C.: The effectiveness of Lloyd-type methods for the k-means problem. In *In 47th IEEE Symposium on the Foundations of Computer Science (FOCS)* (2006), pp. 165–176.

[Phi02] PHILLIPS S. J.: Acceleration of K-means and related clustering algorithms. In *Algorithm Engineering and Experiments (ALENEX)* (2002), pp. 166–177.

[PM99] PELLEG D., MOORE A.: Accelerating exact k-means algorithms with geometric reasoning. In *Proceedings of the Fifth International Conference on Knowledge Discovery in Databases* (aug 1999), Chaudhuri S., Madigan D., (Eds.), AAAI Press, pp. 277–281.

[Ram09] RAMAMOORTHI R.: *Precomputation-Based Rendering*. Foundations and Trends© in Computer Graphics and Vision. Now Publishers Inc, 2009.

[SHHS03] SLOAN P.-P., HALL J., HART J., SNYDER J.: Clustered principal components for precomputed radiance transfer. *ACM Trans. Graph. 22* (July 2003), 382–391.

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph. 21* (July 2002), 527–536.

[XJF*08] XU K., JIA Y.-T., FU H., HU S.-M., TAI C.-L.: Spherical piecewise constant basis functions for all-frequency precomputed radiance transfer. *IEEE Transactions on Visualization and Computer Graphics 14*, 2 (2008), 454–467.