Grassmann Averages for Scalable Robust PCA

Søren Hauberg
DTU Compute*
Lyngby, Denmark
sohau@dtu.dk

Aasa Feragen
DIKU and MPIs Tübingen*
Denmark and Germany
aasa@diku.dk

Michael J. Black
MPI for Intelligent Systems
Tübingen, Germany
black@tue.mpg.de

Abstract

As the collection of large datasets becomes increasingly automated, the occurrence of outliers will increase – "big data" implies "big outliers". While principal component analysis (PCA) is often used to reduce the size of data, and scalable solutions exist, it is well-known that outliers can arbitrarily corrupt the results. Unfortunately, state-of-the-art approaches for robust PCA do not scale beyond small-to-medium sized datasets. To address this, we introduce the Grassmann Average (GA), which expresses dimensionality reduction as an average of the subspaces spanned by the data. Because averages can be efficiently computed, we immediately gain scalability. GA is inherently more robust than PCA, but we show that they coincide for Gaussian data. We exploit that averages can be made robust to formulate the Robust Grassmann Average (RGA) as a form of robust PCA. Robustness can be with respect to vectors (subspaces) or elements of vectors; we focus on the latter and use a trimmed average. The resulting Trimmed Grassmann Average (TGA) is particularly appropriate for computer vision because it is robust to pixel outliers. The algorithm has low computational complexity and minimal memory requirements, making it scalable to "big noisy data." We demonstrate TGA for background modeling, video restoration, and shadow removal. We show scalability by performing robust PCA on the entire Star Wars IV movie.

1. Introduction

Across many fields of science and in many application domains, principal component analysis (PCA) is one of the most widely used methods for dimensionality reduction, modeling, and analysis of data. It is a core technique used throughout computer vision. While methods exist for scaling PCA to large datasets, one fundamental problem has not been addressed. Large datasets are often automatically generated and, therefore, are too large to be verified by humans. As a result, "big data" often implies "big outliers." While there are several solutions that make PCA robust to outliers [6, 7, 9, 11, 16, 30], these generally do not scale to large datasets. Typically, robust methods for PCA are not scalable and scalable methods for PCA are not robust. Our contribution is a novel formulation and a scalable algorithm for robust PCA that outperforms previous methods.

*This work was performed when the S.H. was at the Perceiving Systems Dept. at the MPI for Intelligent Systems. A.F. was jointly affiliated with DIKU, University of Copenhagen and the Machine Learning and Computational Biology Group, MPI for Intelligent Systems and MPI for Developmental Biology, Tübingen.

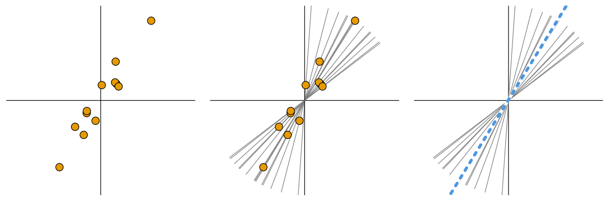
Figure 1. *Left:* A zero-mean dataset represented as a set of points. *Center:* The same data represented as a set of one-dimensional subspaces. *Right:* The blue dotted subspace is the average.

Our first contribution is to formulate subspace estimation as the computation of averages of subspaces. Given a zero-mean dataset $\mathbf{x}_{1:N} \subset \mathbb{R}^D$, we observe that each observation $\mathbf{x}_n$ spans a one-dimensional subspace. Mathematically, each of these subspaces can be described as a point on the Grassmann manifold (the space of subspaces of $\mathbb{R}^D$; see Sec. 3). We then develop an averaging operator on the Grassmann manifold, which formalizes the notion of *the average subspace spanned by the data*; see Fig. 1 for an illustration. We show how this *Grassmann Average (GA)* relates to standard PCA and prove that, for Gaussian data, the subspace found by GA corresponds to that of standard PCA. We show both formally and experimentally that GA is already more robust than PCA; but we can do better.

Understanding how PCA can be reformulated as the computation of averages (of subspaces) leads to a key observation: Robust averages can be computed efficiently. We leverage this fact to define the *Robust Grassmann Average (RGA)*, which generalizes PCA to robust PCA. There are a range of robust averaging methods, and any of these are appropriate for RGA but some scale better than others.

In many fields and applications, outliers exist within a data vector while the majority of the vector is perfectly fine. This is the common situation in computer vision, where, *e.g*. a few pixels in an image may be corrupted. We show that the robust averaging in RGA can be done at the "element" level, and we adopt a trimmed average for this which is both robust and can be computed efficiently. The resulting *Trimmed Grassmann Average (TGA)* can cope with both isolated outliers within a vector (image) or entire data outliers.

The resulting algorithm is straightforward, theoretically sound, and computationally efficient. We provide a detailed evaluation of the method's robustness and efficiency in comparison with related methods. We focus on problems involving images and image sequences although the method is general. We compare with Candes *et al*. [7] on their data and show similar or superior results. Furthermore, we show superior results to both Candes *et al*. and De la Torre and Black [9] on a problem of archival film restoration; here we are able to provide a quantitative comparison. Finally, we show how the method scales by applying it to video modeling problems where all previous robust PCA methods fail; for example, we compute TGA for all 179,415 frames of *Star Wars IV*. Our implementation is online here [15].

2. Related Work

PCA finds the subspace in which the projected observations are as similar as possible to the original observations [19]. This is measured by a least-squares energy, which implies that outliers have a large influence on the results. In fact, it is easy to see that even a single outlier can arbitrarily corrupt the estimated components. We say that PCA has a *break-down point* of 0% [16]; *i.e*. if more than 0% of the data are outliers we cannot trust the results.

Robust PCA: This weakness of PCA has inspired much work on robust extensions [6, 7, 9, 11, 16, 30]. Most estimators treat each data point as either an inlier or an outlier [6, 11, 16, 30]. This can be done by estimating weights for each observation such that outliers have small weights, *e.g*. for robust covariance estimation [6, 30]. Other popular approaches include *iterative reweighting schemes* [11] and optimizing the sum-of-distances to the estimated subspace rather than the sum-of-squared-distances [10]. Kwak [20] optimize the same basic L_1 energy as we do, but do not make the link to subspace averaging, and, thus, do not end up with our general pixel-wise robust approach.

In computer vision, the most popular estimators are those from De la Torre and Black [9] and Candes *et al*. [7] as they allow for pixel-wise outliers (as opposed to having the entire image treated as an outlier). De la Torre and Black phrase robust subspace estimation in the framework of *M–estimators* [16] and optimize an energy governed by the well-studied Geman-McClure error function [12]. This energy is optimized using gradient descent with an annealing scheme over the parameters of the Geman-McClure error function. This requires an initial *singular value decomposition (SVD)* followed by a series of iterative updates. While each iteration of this scheme only has linear complexity, the reliance on SVD gives an overall cubic complexity [14]. This makes the algorithm impractical for large datasets.

The approach from Candes *et al*. [7] decomposes the data matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ into $\mathbf{L} + \mathbf{S}$, where $\mathbf{L}$ is low rank and $\mathbf{S}$ is sparse, *i.e*. it contains the outliers. They elegantly show that under very broad assumptions this can be solved as a convex program, and provide a series of algorithms, where the so-called *Inexact ALM* [31] method is most commonly used. This algorithm repeatedly computes a SVD of the data giving it a cubic complexity [14]. Again, this complexity renders the approach impractical for anything beyond small-to-medium scale data sizes. This can to some extent be alleviated through parallel implementations of SVD [29], but this requires the availability of large compute farms and does not address the underlying complexity issues.

Approximation Techniques: The non-linear complexity of the different robust formulations of PCA has led to the development of a multitude of approximation techniques. Both Mackey *et al*. [25] and Lui *et al*. [24] propose subsampling algorithms for solving the decomposition problem of Candes *et al*. [7]. This allows for improved scalability of the algorithms as they need only solve smaller SVD problems. While theoretical analysis of the subsampling methods shows that they provide good approximations, it is inherently disappointing that to analyse large quantities of data one should ignore most of it. In a similar spirit, Mu *et al*. [27] propose to replace the data matrix with a low-rank counterpart found through a random projection during optimization. This again allows for computational gains as smaller SVD problems need to be computed as parts of the data are not used.

Scalable PCA: The most well-known approaches perform PCA either through an eigenvalue decomposition of the data covariance matrix, or through an SVD of the data matrix [19]. The former only works well in low-dimensional scenarios, where the covariance can easily be stored in memory, while the latter is impractical when either the number of observations or the dimensionality is too large.

Due to the importance of PCA for data analysis, scalable implementations are readily available [1, 28]. A common choice is the EM PCA algorithm [28], which computes each principal component by repeatedly iterating

$$\tilde{\mathbf{v}} \leftarrow \sum_{n=1}^{N} (\mathbf{x}_n^T \mathbf{v}_{i-1}) \mathbf{x}_n \qquad \text{and} \qquad \mathbf{v}_i \leftarrow \frac{\tilde{\mathbf{v}}}{\|\tilde{\mathbf{v}}\|} \tag{1}$$

until convergence. Here $\mathbf{v}_i$ denotes the estimate of the principal component at iteration i of the algorithm. This can either be derived as an EM algorithm [4] under an uninformative

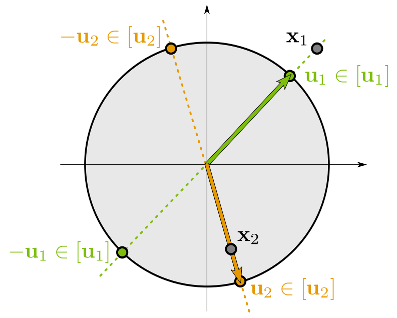
$-\mathbf{u}_2 \in [\mathbf{u}_2]$
$\mathbf{x}_1$
$\mathbf{u}_1 \in [\mathbf{u}_1]$
$\mathbf{x}_2$
$-\mathbf{u}_1 \in [\mathbf{u}_1]$
$\mathbf{u}_2 \in [\mathbf{u}_2]$

Figure 2. An illustration of the Grassmann manifold $\mathrm{Gr}(1, 2)$ of 1-dimensional subspaces of $\mathbb{R}^2$. Any subspace is represented by a point on the unit sphere with the further identification that opposing points on the sphere correspond to the same point on $\mathrm{Gr}(1, D)$.

prior, as a gradient descent algorithm or as a power iteration [14]. One iteration of the algorithm has linear complexity and the low memory use makes it highly scalable. Ideally, we want a robust algorithm with similar properties.

3. The Grassmann Average

Our key contribution is to formulate dimensionality reduction as the estimation of the average subspace of those spanned by the data. Informally, if we seek an averaging operation that returns a subspace, then the input to this operation must be a collection of subspaces. To formalize this idea we need the notion of "a space of subspaces," which is provided by the classical geometric construction known as the *Grassmann manifold* [21, p. 24]:

Definition 1 *The Grassmann manifold* $\mathrm{Gr}(k, D)$ *is the space of all* k*-dimensional linear subspaces of* $\mathbb{R}^D$.

Assuming we are given a zero-mean data set $\mathbf{x}_{1:N}$, then each observation spans a one-dimensional subspace of $\mathbb{R}^D$ and, hence, is a point in $\mathrm{Gr}(1, D)$. The space of one-dimensional subspaces in $\mathbb{R}^D$ takes a particularly simple form since any such subspace can be represented by a unit vector $\mathbf{u}$ or its antipode $-\mathbf{u}$. Thus, $\mathrm{Gr}(1, D)$ can be written as the quotient $S^{D-1}/\{\pm 1\}$ of the unit sphere with respect to the antipodal action of the group $\{\pm 1\}$ [21]. We denote points in $\mathrm{Gr}(1, D)$ as $[\mathbf{u}] = \{\mathbf{u}, -\mathbf{u}\}$, and note that $[\mathbf{u}] = [-\mathbf{u}]$. In other words, given an observation $\mathbf{x}_n$ we represent the spanned subspace as $[\mathbf{u}_n] = \pm \mathbf{x}_n / \|\mathbf{x}_n\|$, where the sign is unknown. This is illustrated in Fig. 2.

3.1. The Average Subspace

With zero-mean data, we now define the average subspace corresponding to the first principal component. Weighted averages are commonly defined as the minimizer of the weighted sum-of-squared distances; *i.e.*

$$[\mathbf{q}] = \operatorname*{arg\,min}_{[\mathbf{v}] \in \mathrm{Gr}(1,D)} \sum_{n=1}^{N} w_n \mathrm{dist}^2_{\mathrm{Gr}(1,D)}([\mathbf{u}_n], [\mathbf{v}]), \qquad (2)$$

where $w_{1:N}$ are weights and $\mathrm{dist}_{\mathrm{Gr}(1,D)}$ is a distance on $\mathrm{Gr}(1, D)$. We define this distance through the quotient space construction of $\mathrm{Gr}(1, D)$ as [5, p. 65]

$$\begin{aligned} &\mathrm{dist}^2_{\mathrm{Gr}(1,D)}([\mathbf{u}_1], [\mathbf{u}_2]) \\ &\quad = \min\{\mathrm{dist}^2_{S^{D-1}}(\mathbf{v}_1, \mathbf{v}_2) | \mathbf{v}_1 \in [\mathbf{u}_1], \mathbf{v}_2 \in [\mathbf{u}_2]\}, \end{aligned} \qquad (3)$$

where $\mathrm{dist}_{S^{D-1}}$ is a yet-to-be-defined distance on the unit sphere S^{D-1}. This distance will be chosen to give an efficient algorithm for computing averages on $\mathrm{Gr}(1, D)$, but first we prove a useful relationship between averages on $\mathrm{Gr}(1, D)$ and those on the unit sphere S^{D-1}:

Lemma 1 *For any weighted average* $[\mathbf{q}] \in \mathrm{Gr}(1, D)$ *satisfying Eq. 2, and any choice of* $\mathbf{q} \in [\mathbf{q}] \subset S^{D-1}$ *there exist* $\mathbf{u}_{1:N} \subset [\mathbf{u}_{1:N}] \subset S^{D-1}$ *such that* $\mathbf{q}$ *is a weighted average on* S^{D-1}*:*

$$\mathbf{q} = \operatorname*{arg\,min}_{\mathbf{v} \in S^{D-1}} \sum_{n=1}^{N} w_n \mathrm{dist}^2_{S^{D-1}}(\mathbf{u}_n, \mathbf{v}). \qquad (4)$$

Proof Select any $\mathbf{q} \in [\mathbf{q}]$. Note, from (3), that $\mathrm{dist}_{S^{D-1}}(\mathbf{q}, \mathbf{u}_n) \geq \mathrm{dist}_{\mathrm{Gr}(1,D)}([\mathbf{q}], [\mathbf{u}_n])$ for all n and $\mathbf{u}_n \in [\mathbf{u}_n]$, thus, the function minimized in (4) will never have a smaller minimum value than the one found in (2). Note moreover that we can find $\mathbf{u}_n \in [\mathbf{u}_n]$ such that $\mathrm{dist}_{S^{D-1}}(\mathbf{u}_n, \mathbf{q}) = \mathrm{dist}_{\mathrm{Gr}(1,D)}([\mathbf{u}_n], [\mathbf{q}])$. But then $\mathbf{q}$ is a minimizer of (4) and hence a weighted average on S^{D-1}. □

Lemma 1 is useful because it reduces the problem of computing means on $\mathrm{Gr}(1, D)$ to a problem of computing means on S^{D-1}, optimized over the possible representatives $\pm\mathbf{u}_{1:N} \subset [\mathbf{u}_{1:N}]$.

3.2. Computing the Average Subspace

To compute averages on $\mathrm{Gr}(1, D)$ we suggest a straightforward algorithm, which repeats the following two steps until convergence (code is in [15]):

1. Pick representations $\mathbf{u}_{1:N}$ of $[\mathbf{u}_{1:N}]$ that are closest to the current average estimate $\mathbf{q}$ under $\mathrm{dist}_{S^{D-1}}$.
2. Update $\mathbf{q}$ to be the spherical average of $\mathbf{u}_{1:N}$.

For this algorithm to be practical, we need the ability to compute fast spherical averages. With this in mind, we pick

$$\mathrm{dist}^2_{S^{D-1}}(\mathbf{u}_1, \mathbf{u}_2) = \frac{1}{2}\|\mathbf{u}_1 - \mathbf{u}_2\|^2 = 1 - \mathbf{u}_1^T \mathbf{u}_2. \qquad (5)$$

Under this distance, the weighted average of data $\mathbf{u}_{1:N} \subset S^{D-1}$ is given in closed-form by [26, §2.1]

$$\mathbf{q} = \frac{\boldsymbol{\mu}(w_{1:N}, \mathbf{u}_{1:N})}{\|\boldsymbol{\mu}(w_{1:N}, \mathbf{u}_{1:N})\|}, \qquad (6)$$

where

$$\boldsymbol{\mu}(w_{1:N}, \mathbf{u}_{1:N}) = \left(\sum_{n=1}^{N} w_n\right)^{-1} \sum_{n=1}^{N} w_n \mathbf{u}_n \qquad (7)$$

denotes the weighted *Euclidean* average. As we need to normalize the data $\mathbf{x}_{1:N}$ to unit length before viewing each observation as a subspace it is natural to pick weights $w_n = \|\mathbf{x}_n\|$. With this choice, we arrive at the following iterative scheme for computing an average subspace:

Algorithm: Grassmann Average (GA)

$$w_n \leftarrow \operatorname{sign}(\mathbf{u}_n^T \mathbf{q}_{i-1})\|\mathbf{x}_n\|, \quad \mathbf{q}_i \leftarrow \frac{\boldsymbol{\mu}(w_{1:N}, \mathbf{u}_{1:N})}{\|\boldsymbol{\mu}(w_{1:N}, \mathbf{u}_{1:N})\|}, \tag{8}$$

where $\mathbf{u}_n = {}^{\mathbf{x}_n}/_{\|\mathbf{x}_n\|}$ and i denotes the iteration number.

We initialize this scheme with a random unit vector.

In Sec. 3.2.1–3.2.3 we discuss how to extend the approach to estimate multiple components, which energy it optimizes, and show that Eq. 8 coincides with ordinary PCA for Gaussian data where $\mathbf{q}_i$ is the first principal component.

3.2.1 Multiple Components

The above algorithm only computes the leading component of the data. We compute additional components in a greedy fashion by adding the further constraint that components should be orthogonal. Following EM PCA [28], we achieve this by removing the estimated component from the data and computing the next component from the residual. Formally, if $\mathbf{X} \in \mathbb{R}^{N \times D}$ denotes the data matrix, then $\hat{\mathbf{X}} \leftarrow \mathbf{X} - (\mathbf{X}\mathbf{q})\mathbf{q}^T$ is the updated data matrix where the component $\mathbf{q}$ is removed. The next component can then be computed on $\hat{\mathbf{X}}$ as above. This approach is optimal for Gaussian data (see Sec. 3.2.3), but we do not have results for other distributions.

This greedy approach is problematic for EM PCA as inaccuracies in one estimated component influences the following components. This is *not* the case for the GA algorithm as it returns an *exact* result.

3.2.2 An Energy Minimization View

The GA algorithm represented by Eq. 8 is derived as an algorithm for computing the average subspace spanned by the data, but it can prove insightful to look at which underlying energy is being optimized. From Eq. 3 and 5 we see that

$$\operatorname{dist}^2_{\mathrm{Gr}(1,D)}([\mathbf{u}_1], [\mathbf{u}_2]) = 1 - |\mathbf{u}_1^T \mathbf{u}_2|, \tag{9}$$

and the average (2) reduces to

$$[\mathbf{q}] = \operatorname*{arg\,max}_{\mathbf{v} \in S^{D-1}} \sum_{n=1}^{N} |\mathbf{x}_n^T \mathbf{v}|. \tag{10}$$

This is an energy in the *projection pursuit* family [17] and was also studied by Kwak [20] who proved that GA algorithm (8) converges to a local optimum in finite time.

Compare Eq. 10 with PCA, which optimizes [19]

$$\mathbf{q}_{\mathrm{PCA}} = \operatorname*{arg\,max}_{\mathbf{v} \in S^{D-1}} \sum_{n=1}^{N} (\mathbf{x}_n^T \mathbf{v})^2. \tag{11}$$

We see that the GA can be expected to be more resilient to outliers than ordinary PCA as the square has been replaced with an absolute value. However, as GA relies on an ordinary average (7) we see that it still sensitive to outliers. In Sec. 3.3 we will improve upon this with robust averages.

3.2.3 Relationship to PCA (Gaussian Data)

To show how GA is related to ordinary PCA, we now show that they coincide when the observed data follows a normal distribution. Let $\mathbf{x}_{1:N} \sim \mathcal{N}(\mathbf{0}, \Sigma)$ be sampled from a normal distribution on $\mathbb{R}^D$. The expected value of Eq. 10 is

$$\operatorname*{arg\,max}_{\mathbf{v} \in S^{D-1}} \mathbb{E}\left(\sum_{n=1}^{N} |\mathbf{v}^T \mathbf{x}_n|\right) = \operatorname*{arg\,max}_{\mathbf{v} \in S^{D-1}} \sum_{n=1}^{N} \mathbb{E}(|\mathbf{v}^T \mathbf{x}_n|). \tag{12}$$

Theorem 2 *The subspace of $\mathbb{R}^D$ spanned by the expected value (12) of the GA of $\mathbf{x}_{1:N}$ coincides with the expected first principal component.*

Proof Since $\mathbf{x}_n$ is sampled from a normal distribution, the projections $\mathbf{v}^T\mathbf{x}_n$ follow a univariate normal distribution $\mathcal{N}(0, \sigma^2_{\mathbf{v}})$ [4, §2.3]. Thus, $|\mathbf{v}^T\mathbf{x}_n|$ follow a half-normal distribution with expected value proportional to $\sigma_{\mathbf{v}}$ [22]. The standard deviation $\sigma_{\mathbf{v}}$ is maximized when $\mathbf{v}$ is the principal eigenvector of Σ, thus the expected value of the GA coincides with the expected first principal component as defined by ordinary PCA. $\square$

The extension to multiple components follows by induction. We empirically verify the result in [15].

3.3. Robust Grassmann Averages

The core computational part of the GA algorithm (8) is the spherical average (6), which, in turn, merely computes a normalized Euclidean average (7). From this, we see that even a single outlier can arbitrarily corrupt the result.

A straight-forward solution to this issue is to replace the average with a *robust average*, giving the following scheme:

Algorithm: Robust Grassmann Average (RGA)

$$w_n \leftarrow \operatorname{sign}(\mathbf{u}_n^T \mathbf{q}_{i-1})\|\mathbf{x}_n\|, \quad \mathbf{q}_i \leftarrow \frac{\boldsymbol{\mu}_{\mathrm{rob}}(w_{1:N}, \mathbf{u}_{1:N})}{\|\boldsymbol{\mu}_{\mathrm{rob}}(w_{1:N}, \mathbf{u}_{1:N})\|}, \tag{13}$$

where $\mathbf{u}_n = {}^{\mathbf{x}_n}/_{\|\mathbf{x}_n\|}$ and $\boldsymbol{\mu}_{\mathrm{rob}}$ denotes any robust average.

We call this approach the *Robust Grassmann Average (RGA)*. Here, $\boldsymbol{\mu}_{\mathrm{rob}}$ can be chosen to have the robustness properties relevant to the application.

In computer vision applications, we are often dealing with image data where individual pixels are corrupted. Consequently, when computing average images robustly it is common to do so on a *per pixel basis*; *i.e.* as a series of one-dimensional robust averages. A standard approach to create a robust one-dimensional average is the *trimmed average* [16]. This removes the largest and smallest observations prior to computing the mean. If we remove $P\%$ of the data from both sides, we get a break-down point of $P\%$. For $P = 50\%$ we get the maximally robust median [16].

To build a subspace estimator that is robust with respect to pixel outliers, we pick $\boldsymbol{\mu}_{\text{rob}}$ as the pixel-wise trimmed mean. We call the resulting estimator the *Trimmed Grassmann Average (TGA)* and let TGA(P, K) denote the estimator that finds K components with an average that trims P percent of the smallest and largest elements in the data. Note that P and K are the only parameters of the estimator.

Each iteration of TGA(P, K) has computational complexity $\mathcal{O}(KND)$ as the trimmed average can be computed in linear time using a selection algorithm [8, §9]. We have, thus, designed a robust subspace estimator that can deal with pixel-wise outliers and has the same computational complexity as both GA (8) and EM PCA (1).

4. Results

In this section we provide results on different tasks, where we compare with the approach from Candes *et al*. [7] and De la Torre and Black [9]. For comparative purposes, the robust Grassmann averages are also implemented in Matlab, and all experiments are performed on an Intel Xeon W3680 with 24 GB of memory. When using TGA we subtract the pixel-wise median to get a robust estimation of a “zero mean” dataset, and we trim at 50% when computing subspaces. For [7, 9] we use the default parameters of the published code[1]. This is discussed further in [15].

In Sec. 4.1 we provide an application of robust PCA for restoration of archival film data, while in Sec. 4.2 and 4.3 we repeat the experiments from [7] on both the previously used data as well as new data. We quantitatively compare the robustness of different methods in Sec. 4.4 and show how the different methods scale to large data in Sec. 4.5. Additional experiments, video, and details are provided in [15]

4.1. Video Restoration

Archival films often contain scratches, gate hairs, splices, dust, tears, brightness flicker and other forms of degradation. Restoring films often involves manual labor to identify and repair such damage. As effects vary in both space and time, we treat them collectively as pixel-level outliers. Assuming a static camera, we perform robust PCA on the frames and reconstruct the movie using the robust subspace. We perform the reconstruction using orthogonal projection, but robust alternative exist [3]; see [15] for a more detailed discussion.

[1]http://perception.csl.illinois.edu/matrix-rank/sample_code.html#RPCA and http://users.salleurl.edu/~ftorre/papers/rpca2.html

As an example, we consider a scene from the 1922 classic movie *Nosferatu*. We restore the frames using TGA and [7, 9]. For TGA and [9] we compute 80 components, which corresponds to the number of frames in the scene. Representative frames are shown in Fig. 3 and the restored movies are available in [15]. TGA removes or downweighs the outliers while mostly preserving the inliers. Both algorithms from Candes *et al*. [7] and De la Torre and Black [9] oversmooth the frames and drastically reduce the visual quality of the video. This is highly evident in the supplementary video.

Note that this is not a fully general algorithm for film restoration but with modifications for camera stabilization and dealing with fast-moving regions, TGA could form a core component in a film restoration system, *cf.* [18].

To quantify the performance of the algorithms, we generate artificially corrupted videos by taking the detected outliers from Nosferatu, binarizing them to create an outlier mask, translating these masks and combining them in random order to produce new outlier masks and then applying them to clean frames from recent Hollywood movies. To measure the accuracy of the different reconstructions, we measure the mean absolute deviation between the original and the reconstructed images at the pixels where noise was added. This is shown in Table 1 where TGA is consistently better than the other algorithms as they tend to oversmooth the results. Here we use a pixel-wise median; *i.e.* 50% trimming. A plot of the impact of this trimming parameter is available in [15]. For completeness, Table 1 also shows the accuracy attained by ordinary PCA and GA (8).

4.2. Background Modeling

We now repeat an experiment from Candes *et al*. [7] on background modeling under varying illumination conditions. A static camera records a scene with walking people, while the lighting conditions change throughout the sequence. Robust PCA should be able to capture the illumination difference such that the background can be subtracted.

The first sequence, recorded at an airport [23], was used by Candes *et al*. [7] to compare with the algorithm from De la Torre and Black [9]. Due to lack of space we only compare with [7], which was reported to give the best results.

Figure 4 (left) shows select frames from the airport sequence as well the reconstruction of the frame with Inexact ALM and TGA with 50% trimming and 5 components. See also the supplementary video. In general, TGA gives noticeably fewer ghosting artifacts than [7].

We repeat this experiment on another video sequence from the CAVIAR[2] dataset. The results, which are shown in Fig. 4 (right), confirm the results from the airport sequence as TGA(50%,5) produces noticeably less ghosting than [7].

[2]http://homepages.inf.ed.ac.uk/rbf/CAVIAR/

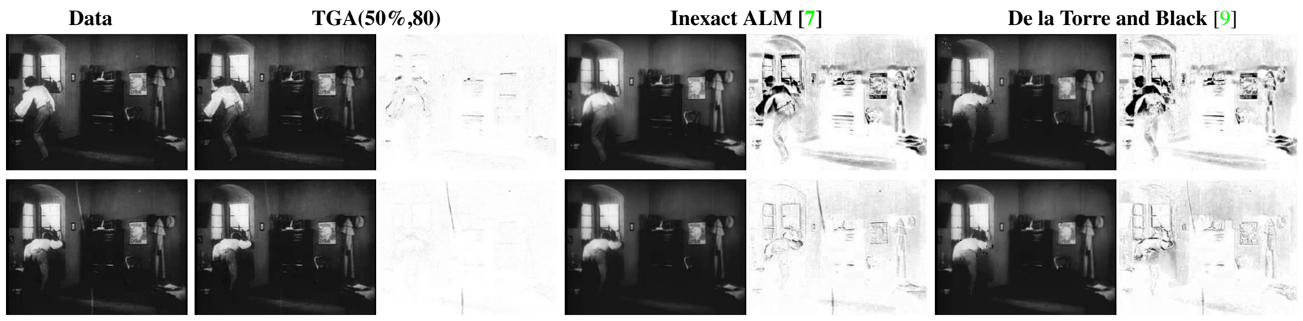

Figure 3. Representative frames from the 1922 film *Nosferatu* as well as their restoration using TGA and the algorithms from [7, 9]. TGA removes many outliers and generally improves the visual quality of the film, while Inexact ALM oversmoothes the results and De la Torre and Black oversmooth and introduce artifacts. This is also seen in the absolute difference between the data and the reconstruction, which is also shown (inverted and multiplied by 2 for contrast purposes). *We also refer the reader to the supplementary video* [15].

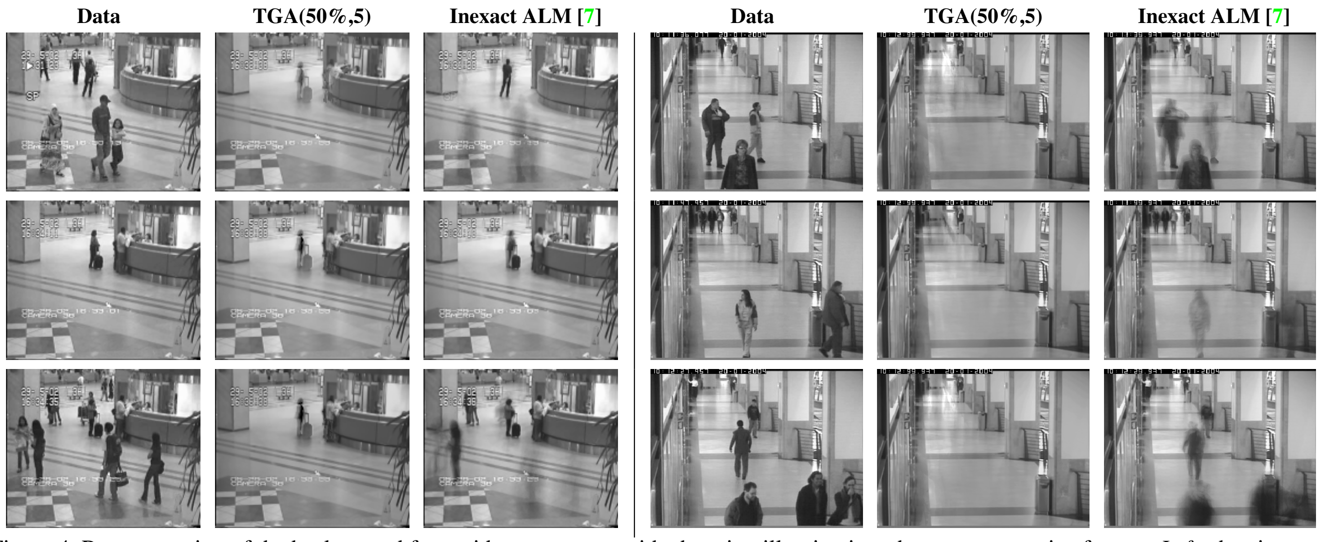

Figure 4. Reconstruction of the background from video sequences with changing illumination; three representative frames. *Left:* the airport sequence (3584 frames), which was also used in [7]. *Right:* a sequence from the CAVIAR dataset (1500 frames).

	Groundhog Day (85 frames)	**Pulp Fiction (85 frames)**
TGA(50%,80)	**0.0157**	**0.0404**
Inexact ALM [7]	0.0168	0.0443
De la Torre and Black [9]	0.0349	0.0599
GA (80 comp)	0.3551	0.3773
PCA (80 comp)	0.3593	0.3789

Table 1. The mean absolute reconstruction error of the different algorithms on recent movies with added noise estimated from *Nosferatu*. TGA is consistently better as it does not oversmooth.

4.3. Shadow Removal for Face Data

We also repeat the shadow removal experiment from Candes *et al*. [7]. The *Extended Yale Face Database B* [13] contains images of faces under different illumination conditions, and, hence, with different cast shadows. Assuming the faces are convex Lambertian objects, they should lie near a nine-dimensional linear space [2]. Relative to this model, the cast shadows can be thought of as outliers and we should be able to remove them with robust PCA. We, thus, estimate the shadow-free images using TGA(50%,9).

We study the same two people as in Candes *et al*. [7]. Figure 5 shows the original data, the reconstruction using different algorithms as well as the absolute difference between images and reconstructions. While there is no "ground truth", both TGA and Inexact ALM [7] do almost equally well at removing the cast shadows, though TGA seems to keep more of the shading variation and specularity while Inexact ALM produces a result that appears more matte.

4.4. Vector-level Robustness

So far, we have considered examples where outliers appear pixel-wise. For completeness, we also consider a case where the entire vector observation is corrupted. We take two 425-frame clips from contemporary Hollywood movies (the same scenes as in the quantitative experiment in Sec. 4.1) and treat one clip as the inliers. We then add an increasing

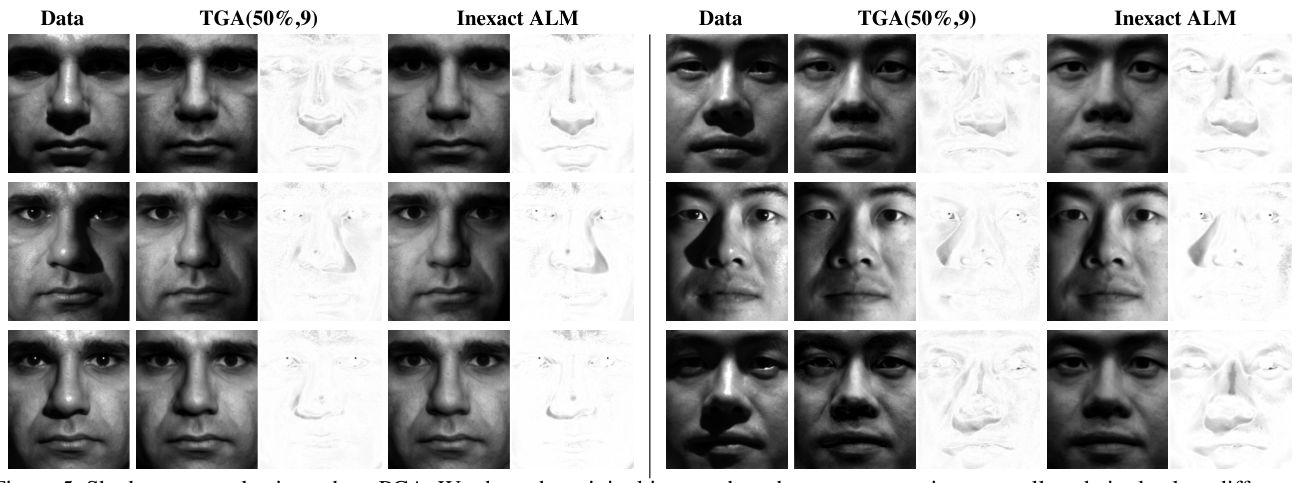

Figure 5. Shadow removal using robust PCA. We show the original image, the robust reconstructions as well as their absolute difference (inverted). TGA preserves more specularity, while Inexact ALM produce more matte results.

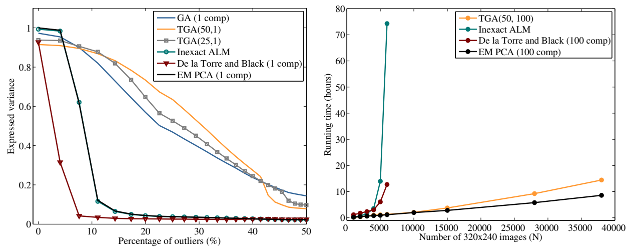

Figure 6. *Left:* the *expressed variance* for different methods as a function of the percentage of vector-level outliers. *Right:* Running time of the algorithms as a function of the size of the problem. The dimensionality is fixed at $D = 320 \times 240$ while the number of observations N increase. TGA is comparable to EM PCA, while [7] and [9] are unable to handle more than 6000 observations.

number of frames from the second clip and measure how this influences the estimated components. We use the standard error measure, *expressed variance* [11], which is the ratio between the amount of variance captured by the estimated principal component and that captured by the ground truth component. Here we only consider one component as this emphasizes the difference between the methods; for [7] we first remove outliers and then reduce to a one-dimensional subspace using PCA. Figure 6 (left) show the results. Both [7, 9] have difficulties with this problem, but this is not surprising as they are designed for pixel-wise outliers. Both TGA and GA handle the data more gracefully.

In summary GA is quite robust when there are vector-level outliers and is a good basic alternative to PCA. When every vector has some data-level outliers, GA performs like PCA (Table 1), and TGA offers significant robustness.

4.5. Scalability

An important feature of the Grassmann average is that the algorithm scales gracefully to large datasets. To show this, we report the running time of the different algorithms for increasing numbers of observations. We use video data recorded with a static camera looking at a busy parking lot over a two-day period. Each frame has a resolution of 320×240 and we consider up to 38000 frames.[3] Figure 6 (right) shows the running time. Inexact ALM [7] and the algorithm from De la Torre and Black [9] quickly become impractical, due to their memory use; both algorithms run out of memory with more than 6000 observations. TGA, on the other hand, scales roughly linearly with the data. In this experiment, we compute 100 components, which is sufficient to capture the variation in the background. For comparison, we also show the running time of EM PCA [28]. TGA is only slightly slower than EM PCA, which is generally acknowledged as being among the most practical algorithms for ordinary PCA on large datasets. It is quite remarkable that a robust PCA algorithm can achieve a running time that is comparable to ordinary PCA.[4] While each iteration of EM PCA is computationally cheaper than for TGA, we find that the required number of iterations is often lower for TGA than for EM PCA, which explains the comparable running times. For 38000 images, TGA(50%,100) used 2–134 iterations per component, with an average of 50 iterations. EM PCA, on the other hand, used 3–729 iterations with an average of 203.

To further emphasize the scalability of TGA, we compute the 20 leading components of the entire *Star Wars IV* movie. This consist of 179,415 frames with a resolution of 352×153. Computing these 20 components (see [15]) took 8.5 hours on an Intel Xeon E5-2650 with 128 GB memory.

[3]When represented in double precision floating point numbers, as required by SVD based methods [7, 9], this data requires 22 GB of memory.

[4]Both EM PCA and TGA are implemented in Matlab and have seen the same level of code optimization.

5. Discussion

Principal component analysis is a fundamental tool for data analysis and dimensionality reduction. Previous work has addressed robustness at both the data vector and vector-element level but fails to scale to large datasets. This is troublesome for big data applications where the likelihood of outliers increases as data acquisition is automated.

In this paper, we introduce the *Grassmann average (GA)*, which is a simple and highly scalable approach to subspace estimation that coincides with PCA for Gaussian data. We have further shown how this approach can be made robust by using a robust average, yielding the *robust Grassmann average (RGA)*. For a given application, we only need to define a robust average to produce a suitable robust subspace estimator. We develop the *trimmed Grassmann average (TGA)*, which is a robust subspace estimator working at the vector-element level. This has the same complexity as the scalable EM PCA [28], and empirical results show that TGA is not much slower while being substantially more robust.

The availability of a scalable robust PCA algorithm opens up to many new applications. We have shown that TGA performs well on different tasks in computer vision, where alternative algorithms either produce poor results or fail to run at all. We have shown how we can even compute robust components of entire movies on a desktop computer in a reasonable time. This could enable new methods for representing and searching videos. Further, the ubiquity of PCA makes RGA relevant beyond computer vision; *e.g.* for large datasets found in biology, physics and weather forecasting. Finally, our approach is based on standard building blocks like fast trimmed averages. This makes it very amenable to speedup via parallelization and to on-line computation.

References

[1] R. Arora, A. Cotter, K. Livescu, and N. Srebro. Stochastic optimization for PCA and PLS. *Communication, Control, and Computing (Allerton)*, pp. 861–868, 2012. 2

[2] R. Basri and D. Jacobs. Lambertian reflectance and linear subspaces. *TPAMI*, 25(2):218–233, 2003. 6

[3] M. J. Black and A. D. Jepson. *EigenTracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation*. *IJCV*, 26:63–84, 1998. 5

[4] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. 2, 4

[5] M. R. Bridson and A. Haefliger. *Metric Spaces of Non-Positive Curvature*. Springer, 1999. 3

[6] N. Campbell. Robust procedures in multivariate analysis I: Robust covariance estimation. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 29(3):231–237, 1980. 1, 2

[7] E. J. Candes, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *ACM*, 58(1):1–37, 2011. 1, 2, 5, 6, 7

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 3rd edition, 2009. 5

[9] F. De la Torre and M. J. Black. A framework for robust subspace learning. *IJCV*, 54:117–142, 2003. 1, 2, 5, 6, 7

[10] C. Ding, D. Zhou, X. He, and H. Zha. R_1-PCA: rotational invariant L1-norm principal component analysis for robust subspace factorization. *ICML*, pp. 281–288, 2006. 2

[11] J. Feng, H. Xu, and S. Yan. Robust PCA in high-dimension: A deterministic approach. *ICML*, pp. 249–256, 2012. 1, 2, 7

[12] S. Geman and D. McClure. Statistical methods for tomographic image reconstruction. *Bulletin of the International Statistical Institute*, 52(4):5–21, 1987. 2

[13] A. Georghiades, P. Belhumeur, and D. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *TPAMI*, 23(6):643–660, 2001. 6

[14] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 3rd edition, Oct. 1996. 2, 3

[15] S. Hauberg, A. Feragen, M.J. Black. Supplementary Material. http://ps.is.tue.mpg.de/project/Robust_PCA 2, 3, 4, 5, 7

[16] P. J. Huber. *Robust Statistics*. Wiley: New York, 1981. 1, 2, 5

[17] P. J. Huber. Projection pursuit. *Annals of Statistics*, 13(2):435–475, 1985. 4

[18] H. Ji, S. Huang, Z. Shen and Y. H. Xu. Robust video restoration by joint sparse and low rank matrix approximation. SIIMS, 4(4):1122–1142, 2011. 5

[19] I. Jolliffe. *Principal Component Analysis*. Springer, 2002. 2, 4

[20] N. Kwak. Principal Component Analysis Based on L1-Norm Maximization. *TPAMI*, 30(9):1672–1680, 2008. 2, 4

[21] J. Lee. *Introduction to Smooth Manifolds*. Springer, 2002. 3

[22] F. C. Leone, L. S. Nelson, and R. B. Nottingham. The folded normal distribution. *Technometrics*, 3(4):543–550, 1961. 4

[23] L. Li, W. Huang, I. Gu, and Q. Tian. Statistical modeling of complex backgrounds for foreground object detection. *IEEE Trans. Image Process.*, 13(11):1459–1472, 2004. 5

[24] R. Liu, Z. Lin, S. Wei and Z. Su. Solving Principal Component Pursuit in Linear Time via l_1 Filtering. http://arxiv.org/abs/1108.5359, 2011 2

[25] L.W. Mackey, A.S. Talwalkar and M.I. Jordan. Divide-and-Conquer Matrix Factorization. *NIPS*, pp. 1134–1142, 2011. 2

[26] K. V. Mardia and P. E. Jupp. *Directional Statistics*. 1999. 3

[27] Y. Mu, J. Dong, X. Yuan and S. Yan. Accelerated low-rank visual recovery by random projection. *CVPR*, pp. 2609-2616, 2011. 2

[28] S. T. Roweis. EM algorithms for PCA and SPCA. *NIPS*. pp. 626–323, 1998. 2, 4, 7, 8

[29] R. Tron and R. Vidal. Distributed computer vision algorithms through distributed averaging. *CVPR*, pp. 57–63, 2011. 2

[30] L. Xu and A. L. Yuille. Robust principal component analysis by self-organizing rules based on statistical physics approach. *IEEE Trans. Neural Networks*, 6(1):131–143, 1995. 1, 2

[31] L. W. Z. Lin, M. Chen and Y. Ma. The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices. Technical report, UILU-ENG-09-2215, 2009. 2

Acknowledgements The authors are grateful to the anonymous reviewers and area chairs for the remarkably useful feedback. S.H. is funded in part by the Villum Foundation and the Danish Council for Independent Research (DFF), Natural Sciences; A.F. is funded in part by the DFF, Technology and Production Sciences.