Scalable Robust Principal Component Analysis using Grassmann Averages

Søren Hauberg, Aasa Feragen, Raffi Enficiaud, and Michael J. Black

Abstract—In large datasets, manual data verification is impossible, and we must expect the number of outliers to increase with data size. While principal component analysis (PCA) can reduce data size, and scalable solutions exist, it is well-known that outliers can arbitrarily corrupt the results. Unfortunately, state-of-the-art approaches for robust PCA are not scalable. We note that in a zero-mean dataset, each observation spans a one-dimensional subspace, giving a point on the Grassmann manifold. We show that the average subspace corresponds to the leading principal component for Gaussian data. We provide a simple algorithm for computing this Grassmann Average (GA), and show that the subspace estimate is less sensitive to outliers than PCA for general distributions. Because averages can be efficiently computed, we immediately gain scalability. We exploit robust averaging to formulate the Robust Grassmann Average (RGA) as a form of robust PCA. The resulting Trimmed Grassmann Average (TGA) is appropriate for computer vision because it is robust to pixel outliers. The algorithm has linear computational complexity and minimal memory requirements. 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; a task beyond any current method. Source code is available online.

Index Terms—Dimensionality reduction, Subspace estimation, Robust Principal Component Analysis

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 yet to be addressed. Large datasets are often collected either fully or semi-automatically and are too large to be verified by humans. As a result, we should expect a significant number of outliers. In essence, “big data” implies “big outliers.” This requires a form of PCA that is robust to outliers. While there are several solutions that make PCA robust to outliers [1]–[6], 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, has low computational complexity, and scales to large datasets.

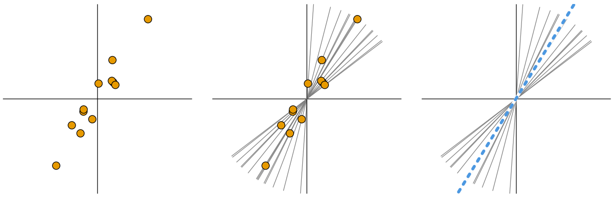

Fig. 1. *Left:* A zero-mean dataset represented as a set of points. *Center:* The same data represented as a set of one-dimensional subspaces. *Right:* The blue dotted subspace is the average subspace.

Our first contribution is to formulate subspace estimation as the computation of averages of subspaces. Given a zero-mean dataset $\boldsymbol{x}_{1:N} \subset \mathbb{R}^D$, we observe that each observation $\boldsymbol{x}_n$ 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 that 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 the GA corresponds to that of standard PCA. We show both formally and experimentally that GA is already more robust than PCA as it optimizes an L_1 style energy. But robustness can be further improved.

Understanding how PCA can be reformulated as the computation of averages (of subspaces) leads to a key observation: Robust averages are theoretically well understood and 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

- *S. Hauberg is with the Section for Cognitive Systems at the Technical University of Denmark.*
E-mail: `sohau@dtu.dk`
- *A. Feragen is with the Department of Computer Science at the University of Copenhagen, Denmark.*
E-mail: `aasa@diku.dk`
- *R. Enficiaud is with the Software Workshop at the Max Planck Institute for Intelligent Systems in Tübingen, Germany.*
E-mail: `raffi.enficiaud@tue.mpg.de`
- *M.J. Black is with the Perceiving Systems Department at the Max Planck Institute for Intelligent Systems in Tübingen, Germany.*
E-mail: `black@tue.mpg.de`

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 that 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.* [5] 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 [6] 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 available online[1], which makes it easy to apply it to new problems.

This paper extends a previous conference paper [7] with a probabilistic interpretation of subspace averages, a more elaborate review of the literature, more intuitive illustrations, and a more thorough experimental evaluation.

2 Related Work

Consider the problem of estimating a one-dimensional subspace. PCA finds the subspace in which the projected observations are as similar as possible to the original observations [8]

$$\boldsymbol{q}_{\text{PCA}} = \mathop{\arg\min}_{\boldsymbol{v},\ \|\boldsymbol{v}\|=1} \sum_{n=1}^{N} \left\| \boldsymbol{x}_n - (\boldsymbol{x}_n^\intercal \boldsymbol{v})\, \boldsymbol{v} \right\|^2 , \tag{1}$$

where the resulting unit vector, $\boldsymbol{v}$, spans the estimated subspace. This least-squares energy implies that outliers have a large influence on the results. Indeed, a single outlier can arbitrarily corrupt the estimated components. We say that PCA has a *break-down point* of 0% [2]; *i.e.* if more than 0% of the data are outliers we cannot trust the results.

2.1 Robust PCA

Many suggestions have been made for how to make PCA robust to outliers. The most classic approaches attempt to estimate the covariance matrix in a robust way by assigning weights to each data point, depending on how likely it is to be an outlier [1]–[4], [9], [10]. In this scenario, the principal components are defined as the eigenvectors of

$$\boldsymbol{\Sigma}_{\text{robust}} = \sum_{n=1}^{N} \alpha_n \boldsymbol{x}_n \boldsymbol{x}_n^\intercal \ , \ \text{ with } \ \alpha_n \geq 0. \tag{2}$$

Xu and Yuille [4] restrict the weights $\alpha_{1:N}$ to be binary and estimate these by optimizing Eq. 1 with an added regularizer $\nu \sum_n (1-\alpha_n)$ that avoids the trivial solution $\alpha_n = 0$. This is optimized using a mean field approximation, which reduces the problem to solving a non-linear program. This works well and provides a classification of the data points into inliers / outliers, but is computationally demanding.

Feng *et al.* [3] iteratively estimate the weights $\alpha_{1:N}$ by performing ordinary PCA and assigning lower weights to points that contribute substantially to the total projected variance. The approach can theoretically handle up to 50% outliers, which makes it very powerful. Unfortunately, the repeated computation of ordinary principal components prevents the algorithm from scaling to large datasets.

Rather than estimating weights for each data point, several authors have suggested replacing the squared error in (1) with a measure that is less sensitive to outliers [5], [6], [11]. Ding *et al.* [11] suggest replacing $\|\cdot\|^2$ with $\|\cdot\|$ in (1), leading to the so-called R_1-PCA. The resulting energy can be optimized by iteratively estimating a robust covariance matrix, which makes the approach computationally demanding. Alternatively, Eq. 1 can be rewritten as the maximization of explained variance

$$\boldsymbol{q}_{\text{PCA}} = \mathop{\arg\max}_{\boldsymbol{v},\ \|\boldsymbol{v}\|=1} \sum_{n=1}^{N} (\boldsymbol{x}_n^\intercal \boldsymbol{v})^2 . \tag{3}$$

Kwak [12] suggests replacing the square in this expression with an absolute value to provide a robust energy, though it is not robust to pixel-level outliers. Kwak provides a local optimizer, while McCoy & Tropp [13] show that the general problem is NP-hard and provide a semi-definite approximation algorithm, which is guaranteed to provide a near-optimal solution. Naor *et al.* [14] extend this work to estimate multiple components. We show that Kwak's approach gives an average subspace, and use this observation to derive a general pixel-wise robust subspace average.

2.1.1 Robust M–Estimators

In computer vision, outliers often occur at the *pixel level* rather than at the *image level* [5], [6]. De la Torre and Black phrase robust subspace estimation in the framework of *M–estimators* [2] and develop a robust extension of Eq. 1 using the Geman-McClure error function [15]. This energy is optimized using gradient descent with an annealing scheme over the parameters of the Geman-McClure error function. Their approach uses 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 [16]. This makes the algorithm impractical for large datasets.

2.1.2 Convex Programs

The approach from Candes *et al.* [5] decomposes the data matrix $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ into

$$\boldsymbol{X} = \boldsymbol{L} + \boldsymbol{S}, \tag{4}$$

[1] http://http://ps.is.tue.mpg.de/research_projects/robust-pca

where $\boldsymbol{L}$ is low rank and $\boldsymbol{S}$ is sparse; *i.e.* it contains the pixel-level outliers. They 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 [17] method is most commonly used. This algorithm repeatedly computes an SVD of the data giving it cubic complexity [16]. Again, this complexity renders the approach impractical for anything beyond small-to-medium scale data sizes. This can be alleviated somewhat through parallel implementations of SVD [18], but this requires the availability of large compute farms and does not address the underlying complexity issues. Netrapalli *et al.* [19] show that the decomposition can be performed with a per-iteration rank-reduced SVD, which only has quadratic. complexity. While an improvement, this is still impractical for large datasets.

2.1.3 *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.* [20] and Lui *et al.* [21] propose subsampling algorithms for solving the decomposition problem of Candes *et al.* [5]. This improves scalability as they need only solve smaller SVD problems. While theoretical analysis shows that subsampling provides 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.* [22] 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 when parts of the data are not used.

2.2 Scalable PCA

Most PCA implementations either perform eigenvalue decomposition on the data covariance matrix or SVD on the data matrix [8]. The former only works 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 large.

Due to the importance of PCA for data analysis, scalable implementations are readily available [23], [24]. A common choice is the EM PCA algorithm [24], which computes each principal component by repeatedly iterating

$$\tilde{\boldsymbol{v}} \leftarrow \sum_{n=1}^{N} (\boldsymbol{x}_n^\intercal \boldsymbol{v}_{i-1}) \boldsymbol{x}_n \qquad \text{and} \qquad \boldsymbol{v}_i \leftarrow \frac{\tilde{\boldsymbol{v}}}{\|\tilde{\boldsymbol{v}}\|} \tag{5}$$

until convergence. Here $\boldsymbol{v}_i$ denotes the estimate of the principal component at iteration i of the algorithm. This can either be derived as an EM algorithm [24] under an uninformative prior, as a gradient descent algorithm or as a power iteration [16]. One iteration of the algorithm has linear complexity and the low memory use makes it highly scalable. Unfortunately, the approach converges slowly and "zig-zags" around the optimum, implying low numerical precision. This imprecision accumulates when multiple components are estimated, which practically limits the approach to only estimate a few of the leading components.

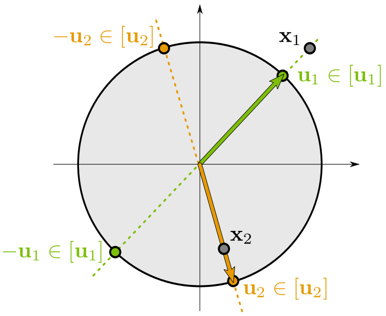
$-\mathbf{u}_2 \in [\mathbf{u}_2]$
$\mathbf{x}_1$
$\mathbf{u}_1 \in [\mathbf{u}_1]$
$\mathbf{x}_2$
$-\mathbf{u}_1 \in [\mathbf{u}_1]$
$\mathbf{u}_2 \in [\mathbf{u}_2]$

Fig. 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 $\mathrm{Gr}(1,D)$.

2.3 Subspaces in Vision

Subspace estimation appears throughout computer vision, so it is not surprising that the Grassmann manifold has seen many applications; for a survey see *e.g.* [25]. In this paper we consider the problem of estimating a single subspace from data. The more general problem of estimating *multiple* subspaces can also be considered. For example *Generalized PCA* [26] provides an elegant algebraic solution to this subspace clustering problem. The numerical solution requires fitting a higher order polynomial to the data, which tends to be unstable in poor signal-to-noise scenarios.

In the absence of noise, the subspace clustering problem can be studied by assuming that any data point can be expressed as a linear combination of *a few* other data points. This leads to a *sparse* convex optimization problem, which can be made robust to noisy data [27]–[29]. The algorithms, however, do not scale to large datasets as they rely on SVD.

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* [30, 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 dataset $\boldsymbol{x}_{1:N}$ with the mean subtracted, 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 $\boldsymbol{u}$ or its antipode $-\boldsymbol{u}$. 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 $\{\pm 1\}$ [30].

We denote points in $\mathrm{Gr}(1, D)$ as

$$[\boldsymbol{u}] = \{\boldsymbol{u}, -\boldsymbol{u}\}, \tag{6}$$

and note that $[\boldsymbol{u}] = [-\boldsymbol{u}]$; *i.e.*, given an observation, $\boldsymbol{x}_n$, we represent the spanned subspace as $[\boldsymbol{u}_n] = \pm \boldsymbol{x}_n / \|\boldsymbol{x}_n\|$, where the sign is unknown (see 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.*

$$[\boldsymbol{q}] = \operatorname*{arg\,min}_{[\boldsymbol{v}] \in \mathrm{Gr}(1,D)} \sum_{n=1}^{N} w_n \mathrm{dist}^2_{\mathrm{Gr}(1,D)}([\boldsymbol{u}_n], [\boldsymbol{v}]), \tag{7}$$

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 [31, p. 65]

$$\begin{aligned}&\mathrm{dist}^2_{\mathrm{Gr}(1,D)}([\boldsymbol{u}_1], [\boldsymbol{u}_2]) \\ &\quad = \min\{\mathrm{dist}^2_{S^{D-1}}(\boldsymbol{v}_1, \boldsymbol{v}_2) \mid \boldsymbol{v}_1 \in [\boldsymbol{u}_1], \boldsymbol{v}_2 \in [\boldsymbol{u}_2]\},\end{aligned} \tag{8}$$

where $\mathrm{dist}_{S^{D-1}}$ is a yet-to-be-defined distance on the unit sphere S^{D-1}. With this, the subspace distance $\mathrm{dist}_{\mathrm{Gr}(1,D)}$ is merely the shortest spherical distance between any possible pair of unit vectors representing the subspaces. The spherical 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 $[\boldsymbol{q}] \in \mathrm{Gr}(1, D)$ satisfying Eq. 7, and any choice of $\boldsymbol{q} \in [\boldsymbol{q}] \subset S^{D-1}$, there exist $\boldsymbol{u}_{1:N} \subset [\boldsymbol{u}_{1:N}] \subset S^{D-1}$ such that $\boldsymbol{q}$ is a weighted average on S^{D-1}:*

$$\boldsymbol{q} = \operatorname*{arg\,min}_{\boldsymbol{v} \in S^{D-1}} \sum_{n=1}^{N} w_n \mathrm{dist}^2_{S^{D-1}}(\boldsymbol{u}_n, \boldsymbol{v}). \tag{9}$$

Proof: Select any $\boldsymbol{q} \in [\boldsymbol{q}]$. Note, from Eq. 8, that $\mathrm{dist}_{S^{D-1}}(\boldsymbol{q}, \boldsymbol{u}_n) \geq \mathrm{dist}_{\mathrm{Gr}(1,D)}([\boldsymbol{q}], [\boldsymbol{u}_n])$ for all n and $\boldsymbol{u}_n \in [\boldsymbol{u}_n]$, thus, the function minimized in Eq. 9 will never have a smaller minimum value than the one found in Eq. 7. Note moreover that we can find $\boldsymbol{u}_n \in [\boldsymbol{u}_n]$ such that $\mathrm{dist}_{S^{D-1}}(\boldsymbol{u}_n, \boldsymbol{q}) = \mathrm{dist}_{\mathrm{Gr}(1,D)}([\boldsymbol{u}_n], [\boldsymbol{q}])$ for all n. But then $\boldsymbol{q}$ is a minimizer of Eq. 9 and hence a weighted average on the sphere S^{D-1}. □

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

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

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

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

$$\mathrm{dist}^2_{S^{D-1}}(\boldsymbol{u}_1, \boldsymbol{u}_2) = \frac{1}{2}\|\boldsymbol{u}_1 - \boldsymbol{u}_2\|^2 = 1 - \boldsymbol{u}_1^{\intercal}\boldsymbol{u}_2. \tag{10}$$

Under this distance, the weighted average of data $\boldsymbol{u}_{1:N} \subset S^{D-1}$ is given in closed-form by [32, §2.1]

$$\boldsymbol{q} = \frac{\boldsymbol{\mu}(w_{1:N}, \boldsymbol{u}_{1:N})}{\|\boldsymbol{\mu}(w_{1:N}, \boldsymbol{u}_{1:N})\|}, \tag{11}$$

where

$$\boldsymbol{\mu}(w_{1:N}, \boldsymbol{u}_{1:N}) = \left(\sum_{n=1}^{N} w_n\right)^{-1} \sum_{n=1}^{N} w_n \boldsymbol{u}_n \tag{12}$$

denotes the weighted *Euclidean* average. This is also the max-likelihood estimator of the mean of the spherical von Mises-Fisher distribution [32]. As we need to normalize the data, $\boldsymbol{x}_{1:N}$, to unit length before viewing each observation as a subspace it is natural to pick subspace weights

$$w_n = \|\boldsymbol{x}_n\|. \tag{13}$$

Noting that the optimal sign for $\boldsymbol{u}_n$ is $\alpha_n = \mathrm{sign}(\boldsymbol{u}_n^{\intercal}\boldsymbol{q}_{i-1})$, we arrive at the following iterative scheme for computing an average subspace:

Algorithm: Grassmann Average (GA)

```
Let $\boldsymbol{u}_n = \frac{\boldsymbol{x}_n}{\|\boldsymbol{x}_n\|}$, and $i = 1$.
repeat
    $\alpha_n \leftarrow \mathrm{sign}(\boldsymbol{u}_n^{\intercal}\boldsymbol{q}_{i-1}) \quad \forall n,$
    $\boldsymbol{q}_i \leftarrow \frac{\boldsymbol{\mu}(w_{1:N}, (\boldsymbol{\alpha u})_{1:N})}{\|\boldsymbol{\mu}(w_{1:N}, (\boldsymbol{\alpha u})_{1:N})\|},$    (14)
    $i \leftarrow i + 1,$
until $\|\boldsymbol{q}_i - \boldsymbol{q}_{i-1}\| < \epsilon$.
```

The algorithm is initialized with a uniform random sample $\boldsymbol{q}_0$. Each iteration of this algorithm has complexity $\mathcal{O}(ND)$, which is the same as EM PCA (5). This algorithm was also studied by Kwak [12] who proved finite-time convergence.

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. We compute this using *deflation* [16] by removing the estimated component from the data and computing the next component from the residual. Formally, if $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ denotes the data matrix, then

$$\hat{\boldsymbol{X}} \leftarrow \boldsymbol{X} - (\boldsymbol{X}\boldsymbol{q})\boldsymbol{q}^{\intercal} \tag{15}$$

is the updated data matrix where the component $\boldsymbol{q}$ is removed. The next component can then be computed on $\hat{\boldsymbol{X}}$ as above. This approach is optimal for Gaussian data (see Sec. 3.2.3). While we do not have results for other distributions we show that it works well in practice. As

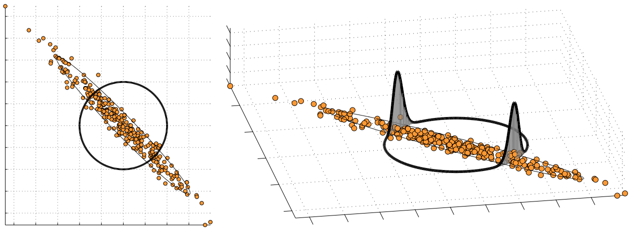

Fig. 3. *Left:* Zero-mean Gaussian data (orange) and the unit circle (black). *Right:* The distribution of the Gaussian data projected onto the unit circle. As the data has zero mean, the projected distribution has two equal-sized modes at opposing points. The Grassmann average is such a mode.

projection on a subspace does not remove outliers, the same level of robustness is needed for every component. This greedy approach is problematic for EM PCA as inaccuracy in one estimated component influences the following components. This is *not* the case for the GA algorithm as the found local minimum is *exact* to numerical precision due to the closed-form spherical average.

3.2.2 An Energy Minimization View

The GA algorithm represented by (14) is derived as an algorithm for computing the average subspace spanned by the data, but it proves insightful to look at which underlying energy is being optimized. From (8) and (10) we see that

$$\mathrm{dist}^2_{\mathrm{Gr}(1,D)}([\boldsymbol{u}_1],[\boldsymbol{u}_2]) = 1 - |\boldsymbol{u}_1^\intercal \boldsymbol{u}_2|, \tag{16}$$

and the average (7) reduces to

$$[\boldsymbol{q}] = \operatorname*{arg\,max}_{\boldsymbol{v}\in S^{D-1}} \sum_{n=1}^{N} |\boldsymbol{x}_n^\intercal \boldsymbol{v}|. \tag{17}$$

This is an energy in the *projection pursuit* family [33]. Compared with PCA, which maximizes explained variance (3), 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 the GA relies on an ordinary average (12) it still has a break-down point of 0%. 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. Before stating a formal result, we provide some intuitions. Let $\boldsymbol{x}_{1:N} \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{\Sigma})$ be sampled from a multivariate zero-mean normal distribution on $\mathbb{R}^D$, with a covariance with distinct eigenvalues. The subspace spanned by data point $\boldsymbol{x}_n$ is represented by the natural basis vector $\boldsymbol{u}_n = \frac{\boldsymbol{x}_n}{\|\boldsymbol{x}_n\|}$. This follows a *projected normal distribution* [34], $\boldsymbol{u}_n \sim \mathcal{PN}(\boldsymbol{0},\boldsymbol{\Sigma})$. Our interest is in the distribution of the one-dimensional subspaces $[\boldsymbol{u}_n]$ spanned by the data. As $\mathcal{PN}(\boldsymbol{0},\boldsymbol{\Sigma})$ is mirror-symmetric we have

$$\begin{aligned} p([\boldsymbol{u}_n]) &= \mathcal{PN}(\{\boldsymbol{u}_n, -\boldsymbol{u}_n\} \mid \boldsymbol{0},\boldsymbol{\Sigma}) && (18)\\ &= \mathcal{PN}(\boldsymbol{u}_n \mid \boldsymbol{0},\boldsymbol{\Sigma}) + \mathcal{PN}(-\boldsymbol{u}_n \mid \boldsymbol{0},\boldsymbol{\Sigma}) && (19)\\ &= 2\mathcal{PN}(\boldsymbol{u}_n \mid \boldsymbol{0},\boldsymbol{\Sigma}). && (20)\end{aligned}$$

As $\mathcal{PN}(\boldsymbol{u}_n \mid \boldsymbol{0},\boldsymbol{\Sigma}) = \mathcal{PN}(-\boldsymbol{u}_n \mid \boldsymbol{0},\boldsymbol{\Sigma})$ we note that the distribution of subspaces $p([\boldsymbol{u}_n])$ is unimodal with a mode corresponding to the first principal component of the data; see also Fig. 3.

Since $p([\boldsymbol{u}_n])$ is symmetric around the mode, it is natural to think that the mode and the mean coincide. This, however, depends on the metric used to compare subspaces. The argument holds for any rotationally invariant metric; in the following we provide a proof for the metric in Eq. 16.

Theorem 2. *Let $\boldsymbol{x}_{1:N} \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{\Sigma})$ be sampled from a multivariate normal distribution on $\mathbb{R}^D$. In the limit of infinite data, the average subspace* (17) *coincides with the first principal component.*

Proof: First note that $\frac{1}{N}\sum_{n=1}^N |\boldsymbol{x}_n^\intercal \boldsymbol{v}|$ is the finite sample approximation of the expectation value $\mathbb{E}(|\boldsymbol{x}^\intercal \boldsymbol{v}|) = \int |\boldsymbol{x}^\intercal \boldsymbol{v}| p(\boldsymbol{x}) d\boldsymbol{x}$. Therefore, in the limit of infinite data, $\arg\max_{\boldsymbol{v}\in S^{D-1}} \frac{1}{N}\sum_{n=1}^N |\boldsymbol{x}_n^\intercal \boldsymbol{v}|$ coincides with $\arg\max \mathbb{E}(|\boldsymbol{x}^\intercal \boldsymbol{v}|)$. Since $\boldsymbol{x}$ is sampled from a normal distribution, the projection $\boldsymbol{x}^\intercal \boldsymbol{v}$ follows a univariate normal distribution, $\mathcal{N}(0,\sigma_{\boldsymbol{v}}^2)$, with $\sigma_{\boldsymbol{v}}^2 = \boldsymbol{v}^\intercal \boldsymbol{\Sigma} \boldsymbol{v}$ [35, §2.3]. Thus, $|\boldsymbol{x}^\intercal \boldsymbol{v}|$ follows a half-normal distribution with expected value proportional to $\sigma_{\boldsymbol{v}}$ [36]. The standard deviation, $\sigma_{\boldsymbol{v}}$, is maximized when $\boldsymbol{v}$ is the principal eigenvector of $\boldsymbol{\Sigma}$, thus $\arg\max \mathbb{E}(|\boldsymbol{x}^\intercal \boldsymbol{v}|)$ coincides with the first principal component as defined by ordinary PCA. □

3.2.4 A Probabilistic Interpretation

The idea of computing an *average* subspace implicitly assumes a unimodal subspace distribution $p([\boldsymbol{u}_n])$. As $[\boldsymbol{u}_n] = \{\boldsymbol{u}_n, -\boldsymbol{u}_n\}$ this implicit assumption can also be seen as a mirror-symmetric bimodal distribution on $\mathcal{S}^{D-1}$; *i.e.* $p(\boldsymbol{u}_n) = p(-\boldsymbol{u}_n) = {}^1\!/\!_2\, p([\boldsymbol{u}_n])$.

An equivalent formulation of the Grassmann average can, thus, be attained as a two-class clustering problem on the unit sphere with antipodal cluster centers. For clusters following a von Mises-Fisher distribution [32], the Grassmann average algorithm (14) can be derived as a spherical two-class k-means with antipodal means. This view has the advantage that it provides a generative model for subspaces. Let the spherical clustering have cluster means $\pm\mu$, *i.e.*

$$p(\boldsymbol{u}_n) = \frac{1}{2}\mathsf{vMF}(\boldsymbol{u}_n \mid \mu,\kappa) + \frac{1}{2}\mathsf{vMF}(\boldsymbol{u}_n \mid -\mu,\kappa) \tag{21}$$

$$\mathsf{vMF}(\boldsymbol{u}_n \mid \mu,\kappa) = \frac{\kappa^{D/2-1}}{(2\pi)^{D/2} I_{D/2-1}(\kappa)} \exp\left(\kappa \mu^\intercal \boldsymbol{u}_n\right), \tag{22}$$

where vMF is the von Mises-Fisher density; I_v the modified Bessel function of the first kind and order v; and $\kappa \geq 0$ a

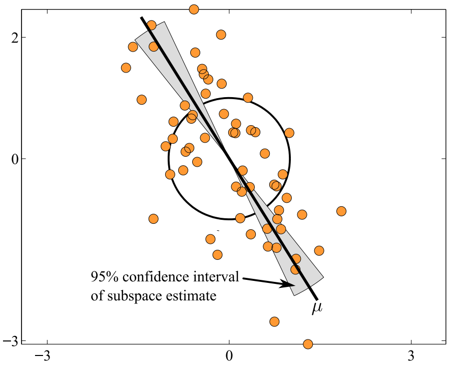

Fig. 4. Gaussian data (orange) along with the average subspace (black line). The light gray subspace interval corresponds to the 95% confidence interval of the mean estimate under the probabilistic interpretation (27).

concentration parameter. The subspace distribution is then

$$p([\boldsymbol{u}_n]) = p(\boldsymbol{u}_n) + p(-\boldsymbol{u}_n) \tag{23}$$
$$= \frac{1}{2}\mathsf{vMF}(\boldsymbol{u}_n \mid \mu, \kappa) + \frac{1}{2}\mathsf{vMF}(-\boldsymbol{u}_n \mid \mu, \kappa) \tag{24}$$
$$+ \frac{1}{2}\mathsf{vMF}(\boldsymbol{u}_n \mid -\mu, \kappa) + \frac{1}{2}\mathsf{vMF}(-\boldsymbol{u}_n \mid -\mu, \kappa) \tag{25}$$
$$= \mathsf{vMF}(\boldsymbol{u}_n \mid \mu, \kappa) + \mathsf{vMF}(-\boldsymbol{u}_n \mid \mu, \kappa) \tag{26}$$
$$= \mathsf{vMF}([\boldsymbol{u}_n] \mid \mu, \kappa). \tag{27}$$

This model immediately allows us to sample new subspaces [32], estimate confidence intervals for the subspace [37, §5.3.1–5.3.2], and derive central limit theorems for the mean estimate [38]. As an example, Fig. 4 shows 95% confidence intervals for the estimation of the leading component from Gaussian data. This type of subspace reasoning is not available with conventional PCA models, but follows easily from the subspace estimation formulation.

3.3 Robust Grassmann Averages

The core computational part of the GA algorithm (14) is the spherical average (11) which, in turn, merely computes a normalized Euclidean average (12). 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)

```
Let $\boldsymbol{\mu}_{\text{rob}}$ denote any robust averaging operator.
Let $\boldsymbol{u}_n = \frac{\boldsymbol{x}_n}{\|\boldsymbol{x}_n\|}$ and $i = 1$.
repeat
        $\alpha_n \leftarrow \text{sign}(\boldsymbol{u}_n^\intercal \boldsymbol{q}_{i-1}) \quad \forall n,$
        $\boldsymbol{q}_i \leftarrow \frac{\boldsymbol{\mu}_{\text{rob}}(w_{1:N}, (\boldsymbol{\alpha u})_{1:N})}{\|\boldsymbol{\mu}_{\text{rob}}(w_{1:N}, (\boldsymbol{\alpha u})_{1:N})\|},$    (28)
        $i \leftarrow i + 1$
until $\|\boldsymbol{q}_i - \boldsymbol{q}_{i-1}\| < \epsilon$ or $i >$ max_iter.
```

The algorithm is initialized with a uniform random sample $\boldsymbol{q}_0$. We call this the *Robust Grassmann Average* (RGA). Here, $\boldsymbol{\mu}_{\text{rob}}$ can be chosen to have the robustness properties relevant to the application. While we do not have a formal general convergence proof, empirical experience indicates that the algorithm converges in finite time.

In computer vision applications, we often deal 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* [2]. 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 [2].

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 and trims P percent of the smallest and largest elements in the data. Note that P and K are the only parameters of the estimator, where the choice $P = 50\%$ gives the maximally robust median [2].

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 [39, §9]. We thus have a robust subspace estimator that can deal with pixel-wise outliers and has the same computational complexity as both GA (14) and EM PCA (5). In the appendix we show that a version of TGA$(50\%, 1)$ converges in finite time to a local minimum of the trimmed version of Eq. 7. We do not have similar results for other versions of RGA, but empirically we find that an optima is always found.

3.4 Practical Considerations

It has great practical value that a robust subspace estimator can be created using only a robust average, as it is often easy to compute a robust average which takes the structure of the problem into account. With this, the data can be made “zero mean”, and a robust Grassmann average (RGA) can be computed by repeated averaging and deflation.

Picking a suitable dimensionality for the estimated subspace is a general problem that influences all current methods either directly or through a regularization parameter. For PCA it is common to pick enough dimensions to explain *e.g.* 90% of the variance in the data. This approach is, however, troublesome in the presence of outliers as they will dominate the variance estimate. Our probabilistic Grassmann average (Sec. 3.2.4) provides an estimate of the confidence interval of a given subspace. One option is to keep increasing the dimensionality of the estimated subspace until this confidence interval is larger than a specified threshold. In practice this is, however, also sensitive to outliers. In our experiments, we either rely on prior knowledge or use a trimmed variance estimator to determine dimensionality.

The RGA algorithm finds a local optimum, so the average subspaces are not guaranteed to be computed in order of decreasing variance. This is a general issue for methods that estimate one basis vector at a time; *e.g.* both EM PCA and the method from De la Torre and Black [6] are not guaranteed to provide ordered components. The standard solution is a simple *post hoc* reordering according to the amount of variance captured by each subspace.

The recursive deflation approach (15) for estimating multiple components relies on the specification of a projection operator. As linear projection is known to be sensitive to outliers [40] it may be beneficial to use a robust projection operator. We investigate this option empirically in Sec. 4.9.

4 Results

In this section we compare our method with the approaches of Candes *et al.* [5] and De la Torre and Black [6]. For comparative purposes, all algorithms are implemented in `Matlab`, and, unless otherwise stated, the experiments are performed on an 6 core 3.3GHz 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 [5] we use the default parameters of the published code[2]. For [6] we also use the default parameters of the published code[3], but with the same number of components as TGA.

Section 4.1 explores the application of robust PCA to the restoration of archival film data, while in Sec. 4.2 and 4.3 we repeat the experiments from [5]. We quantitatively compare the robustness of different methods in Sec. 4.4 and investigate the empirical impact of trimming in Sec. 4.5. Theorem 2 is empirically verified in Sec. 4.6, and the impact of outliers is studied in Sec. 4.7. The random initialization is studied in Sec. 4.8, different projections are compared in Sec. 4.9, and finally Sec. 4.10 measures scalability.

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 these degradations 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 of a movie clip and reconstruct the clip using the robust subspace. We perform the reconstruction using orthogonal projection; this choice is investigated in Sec. 4.9.

As an example, we consider a scene from the 1922 classic movie *Nosferatu*. We restore the frames using TGA, [5], and [6]. We expect the true rank of the inliers to correspond to the number of frames in the sequence. This consists of 80 frames, so for TGA and [6] we compute 80 components. This may intuitively seem like "too many" components, but as robust methods disregard parts of the data (*e.g.* through trimming) they often allow for more components than observations. Figure 5 shows two representative frames and the restored movies are available in the supplementary material. TGA removes or downweights the outliers while mostly preserving the inliers. Both algorithms from Candes *et al.* [5] and De la Torre and Black [6] oversmooth the frames and drastically reduce the visual quality of the video. This is visually evident in the supplementary video.

	Groundhog Day (85 frames)	**Pulp Fiction (85 frames)**
TGA(50%, 80)	**0.0157**	**0.0404**
Inexact ALM [5]	0.0168	0.0443
De la Torre and Black [6]	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*.

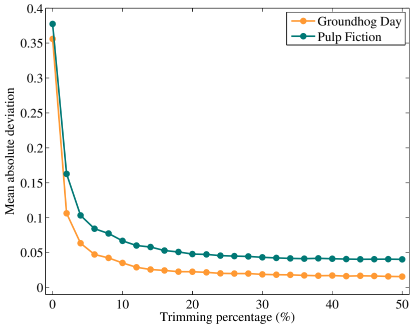

Fig. 6. The mean absolute deviation from ground-truth measured at noisy pixels for two sequences.

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 might form a component in a film restoration system, *cf.* [41].

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 Hollywood movies. This corrupts approximately 10% of the pixels. 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, though Inexact ALM does almost equally well. For completeness, Table 1 also shows the accuracy attained by ordinary PCA and GA (14).

These results are obtained with a pixel-wise median; *i.e.* 50% trimming. Figure 6 shows the impact of this parameter. When we trim more than 10–15% we see substantial improvements compared to no trimming.

4.2 Background Modeling

We now repeat an experiment from Candes *et al.* [5] on background modeling under varying illumination conditions.

[2] http://perception.csl.illinois.edu/matrix-rank/sample_code.html#RPCA

[3] http://users.salleurl.edu/~ftorre/papers/rpca2.html

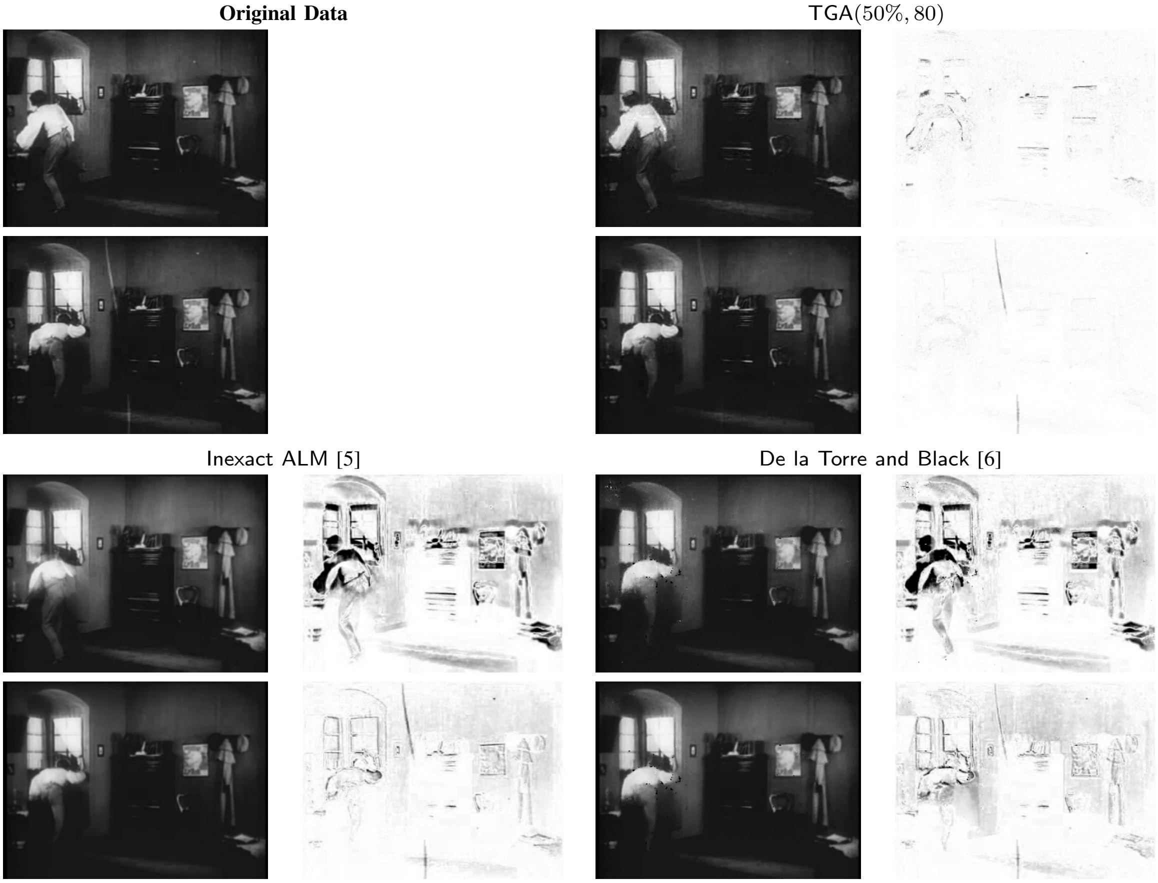

Fig. 5. Two representative frames from the 1922 film *Nosferatu* as well as their restoration using TGA and the algorithms from [5], [6]. TGA removes many outliers and generally improves the visual quality of the film, while Inexact ALM oversmoothes the results and De la Torre and Black oversmooth and 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.

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 [42], was used by Candes *et al.* [5] to compare with the algorithm from De la Torre and Black [6]. We only compare with [5], which was reported to give the best results.

Figure 7 (left) shows select frames from the airport sequence as well as 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 [5].

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

4.3 Shadow Removal for Face Data

We repeat the shadow removal experiment from Candes *et al.* [5], which considers images of faces under different illumination conditions, and, hence, with different cast shadows [43]. Assuming the faces are convex Lambertian objects, they should lie near a nine-dimensional linear space [44]. In this model, the cast shadows are outliers that can be removed with robust PCA. We estimate images with smooth shading, but without cast shadows using TGA$(50\%, 9)$.

We study the same people as Candes *et al.* [5]. Figure 8 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 [5] remove cast shadows equally well, though TGA keeps more of the shading variation and specularity while Inexact ALM produces a more matte result. Note that the theory says that a nine-dimensional subspace is sufficient to capture most of the variance in the reflectance of a smooth Lambertian surface under distant isotropic illumination [44]. It does not say that it has to contain *only* Lambertian reflectance, particularly if the reflectance itself is low dimensional.

4.4 Vector-level Robustness

So far, we have considered examples with pixel-wise outliers. For completeness, we also consider a case where the entire

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

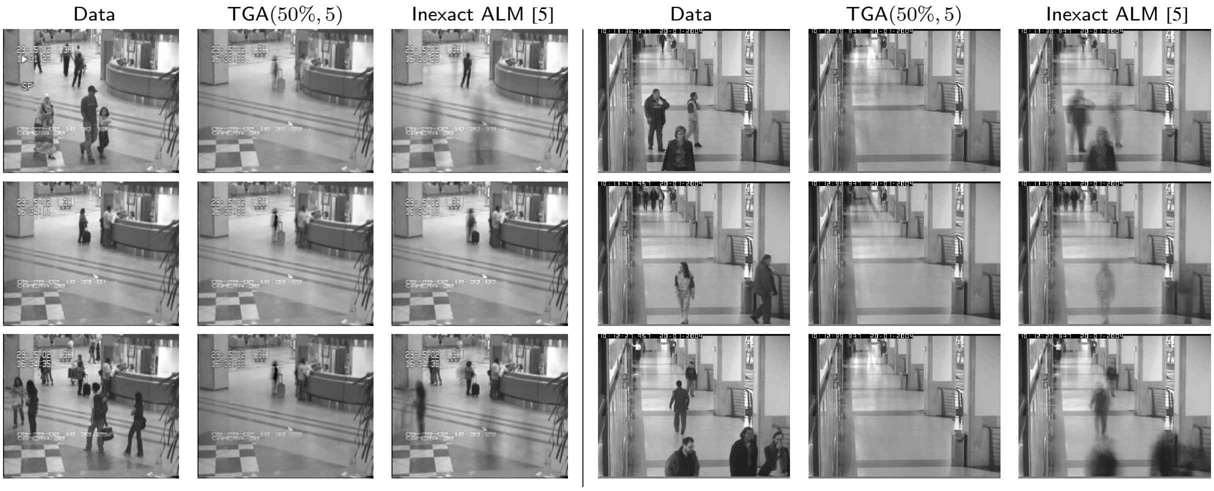

Fig. 7. Reconstruction of the background from video sequences with changing illumination; three representative frames. *Left:* the airport sequence (3584 frames), which was also used in [5]. *Right:* a sequence from the CAVIAR dataset (1500 frames).

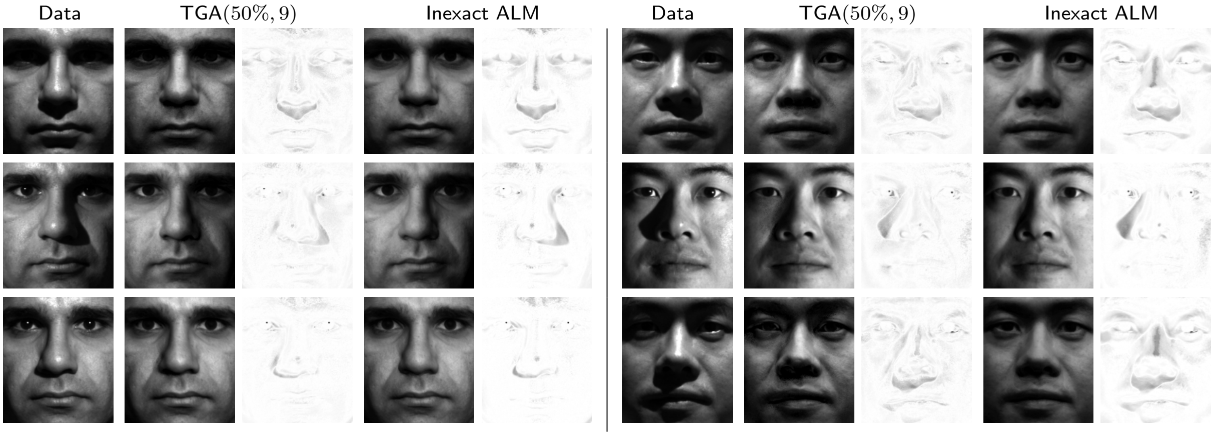

Fig. 8. Shadow removal using robust PCA. We show the original image, the robust reconstructions as well as their absolute difference (inverted). TGA preserves more specularity, while Inexact ALM produce more matte results.

vector observation is corrupted. We take two 425-frame clips from contemporary Hollywood movies (the same as in the quantitative experiment in Sec. 4.1) and treat one clip as the inliers. We then add an increasing number of frames from the second clip and measure how this influences the estimated components. We use the standard error measure, *expressed variance* [3], which is the ratio between the amount of variance captured by the estimated principal component and that captured by the ground truth component $\boldsymbol{v}_{\text{gt}}$,

$$\mathcal{E}(\boldsymbol{v}) = \frac{\sum_{n=1}^{N} (\boldsymbol{x}_n^\intercal \boldsymbol{v})^2}{\sum_{n=1}^{N} (\boldsymbol{x}_n^\intercal \boldsymbol{v}_{\text{gt}})^2}. \tag{29}$$

Here we only consider one component to emphasize the difference between the methods; for [5] we remove outliers and then reduce to a one-dimensional subspace using PCA. Figure 9a shows the results. Both [5] and [6] 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 particular versus PCA.

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 Statistical Efficiency vs. Robustness

In Fig. 9a it is worth noting that TGA(25%, 1) performs better than TGA(50%, 1) for a small number of outliers. Medians are known [2] to require more observations to give accurate answers than a non-robust average, but on the other hand the median is maximally robust. The trimming parameter, thus, provides a trade-off between *statistical efficiency* and *robustness*. This is well-known for robust estimators and it is not surprising that the TGA is affected by this as well. Thus, one should use as low a trimming parameter as the data allows to get as efficient an estimator as possible; see *e.g.* [2] for further discussions on the matter.

Some insight into the statistical efficiency of the TGA estimator can be gained by studying its behavior on data with no outliers. Figure 9b shows the expressed variance as a function of the number of observations N and the level of

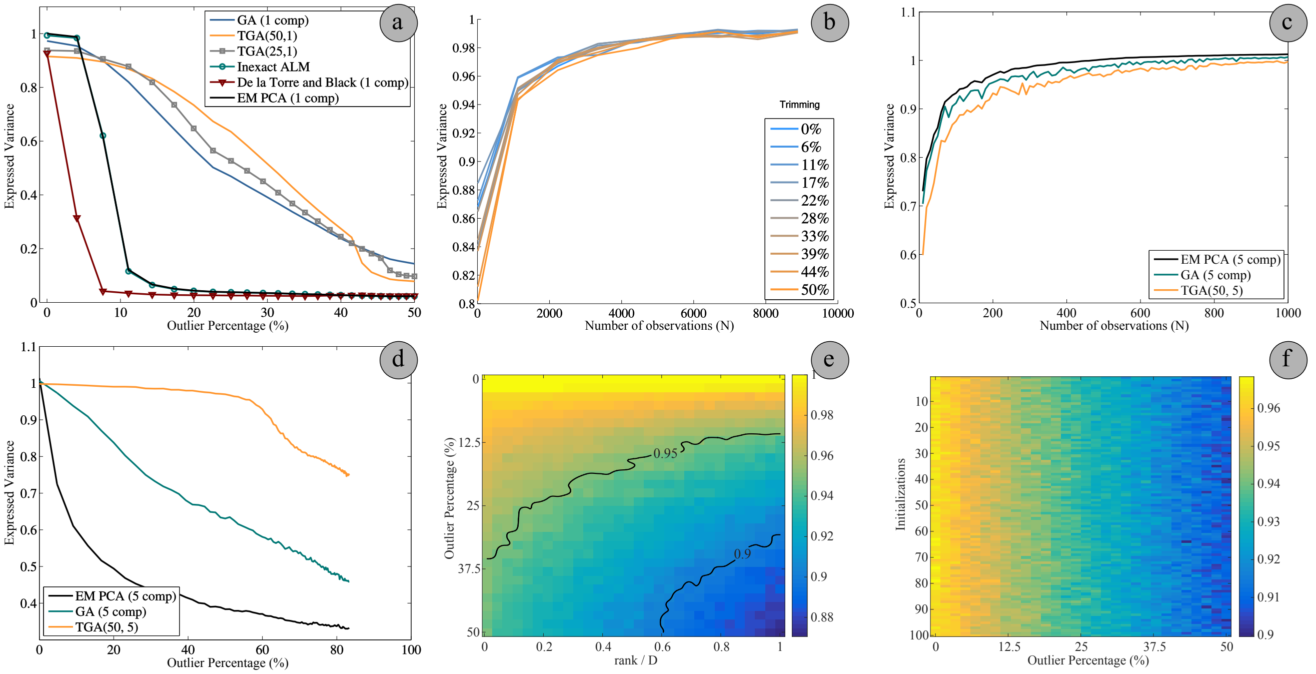

Fig. 9. *(a)* The *expressed variance* for different methods as a function of the percentage of vector-level outliers. Note that EM PCA and Inexact ALM provide near-identical solutions. *(b)* The expressed variance as a function of the number of observations N. Here the dimensionality is fixed at $D = 100$. The different curves correspond to different levels of trimming. *(c)* Expressed variance as a function of number of samples from a known Gaussian distribution. The results are for $D = 30$ dimensional data, and are averaged over 5 experiments with randomly generated covariance matrices. *(d)* Expressed variance for Gaussian inliers with an increasing number of outliers. The results are for $D = 30$ dimensional data, and are averaged over 50 experiments with randomly generated covariance matrices. *(e)* Expressed variance (color coded from 0.86 (blue) to 1 (yellow)) as a function of the relative rank of the inliers verses the percentage of outliers. *(f)* Impact of initialization is studied in TGA by measuring expressed variance (see color code) for different random initializations.

trimming. Here the dimensionality is fixed at $D = 100$. In low-sample scenarios the amount of trimming has a large impact on the quality of the estimator. As the number of observations increase, the impact of trimming decreases.

Throughout this paper, we consistently trim 50%; *i.e.* a pixel-wise median. In some experiments slightly better results can be attained by trimming less, but we opted for a median to avoid parameter tuning.

4.6 Performance on Gaussian Data

Theorem 2 states that for Gaussian data, the component estimated by GA coincides with the principal component defined by PCA. The result is, however, only in expectation and may not hold for finite data samples. We investigate this empirically by sampling data from a Gaussian distribution with known covariance, such that the ground truth principal component can be estimated through an eigendecomposition. Figure 9c shows the expressed variance as a function of the number of samples for both EM PCA, GA and TGA with 50% trimming. EM PCA has slightly better performance compared to GA, which, in turn, is slightly better than TGA. This is to be expected as EM PCA directly optimizes to get the principal component. TGA does slightly worse than GA due to lower statistical efficiency as described in Sec. 4.5. Overall GA and TGA provide results that are very similar to PCA for Gaussian data.

4.7 Performance with Vector-level Outliers

We further measure the methods' performance when the data is Gaussian with vector-level outliers. We sample inliers from a Gaussian distribution with a random covariance $\mathbf{\Sigma}$. Outliers are sampled from a Gaussian distribution with a mean placed along the smallest eigenvector of $\mathbf{\Sigma}$, giving a consistent bias away from the principal components of the inliers. Figure 9d shows the expressed variance for an increasing number of outliers for EM PCA, GA and TGA with 50% trimming. As expected, the performance of EM PCA quickly drops, while the performance of GA drops more slowly. TGA, on the other hand, is very robust and attains a strong performance until the number of outliers exceeds that of the inliers.

We repeat the experiment for TGA with inliers of varying rank. We consider $N = 1000$ inliers in $D = 100$ dimensions with a rank between 1 and D, and add up to N outliers for a total of $2N$ observations. We estimate a robust subspace using TGA$(50\%, 5)$. Figure 9e shows the expressed variance as a function of both rank and number of outliers. In contrast to Fig. 9d, the expressed variance is now color coded with yellow being 1 and dark blue being approximately 0.86. TGA is robust over a broad range of ranks and outlier percentages as can be seen by the contours showing robust solutions capturing 0.9 and 0.95 of expressed variance.

4.8 Quality of Local Optima

TGA provides a locally optimal solution, which depends on a random initialization. To understand the quality of these optima, we sample a single set of inliers ($N = 1000, D = 100$), and add an increasing number of outliers as in Sec. 4.7. Figure 9f shows the expressed variance as a function of the outlier percentage for 100 different initializations. While small fluctuations appear, it is evident that almost equally good solutions are found for all initializations.

4.9 Robust vs. Orthogonal Projections

In the experiments, we estimate robust subspaces with a basis $\boldsymbol{Q} \in \mathbb{R}^{D \times K}$, where K is the number of components and D is the dimensionality of the data space. Data is then projected into the subspace using orthogonal projection,

$$\boldsymbol{P} = \boldsymbol{X}\boldsymbol{Q}\boldsymbol{Q}^{\intercal}, \tag{30}$$

where $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ is the data matrix. This projection finds the subspace point $\boldsymbol{P}$ closest to the original point $\boldsymbol{X}$; *i.e.* a least-squares optimization problem where (30) is the solution [40]. As least-squares problems are sensitive to outliers, it is reasonable to have a robust projection operator.

One such robust projection operator is derived by Black and Jepson [40] using the Geman-McClure error function [15]. This projection is then optimized using gradient descent.

We have experimented with the robust operator, and find that it provides minor improvements. We first consider a synthetic experiment: we select an increasing number of random pixels from the *Groundhog Day* sequence and make them white, thereby adding biased outliers. We estimate a subspace from the noise-free data, and reconstruct the noisy data by projecting it into the subspace. We consider both orthogonal and robust projection. We measure the mean absolute difference between the projections from noise-free and noisy data. Figure 10(left) shows that the robust projection generally is better than orthogonal projection.

For non-synthetic examples we observe less of a difference. To illustrate this, we reconstruct a frame from the *Nosferatu* experiment using both orthogonal and robust projection. The data and results are given in Fig. 10(center), where the different projections appear to give nearly identical results. Minor differences between the methods do appear and the figure shows the pixels where they disagree. When the images are represented using 8 bits per pixel (*i.e.* between 0 and 255), the largest observed absolute difference between the reconstructions was 1 gray level.

As the use of a robust projection only resulted in small improvements, we opted for the much faster orthogonal projection throughout the paper. We speculate that the small difference between robust and orthogonal projection is due to the outliers having a maximal value of 255.

It is worth noting that outliers generally do not influence the deflation estimator (15) of multiple components. After the leading component has been subtracted from the data, the outliers will generally still be present in the data, if the leading component was estimated robustly. As long as a robust subspace estimator is applied, this procedure can be repeated until the outliers dominate the residual data.

4.10 Scalability

An important feature of the Grassmann average is that it 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 $38,000$ frames, requiring a total of 22 GB of memory. Figure 10(right) shows the running times. Inexact ALM [5] and the algorithm from De la Torre and Black [6] 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 [24]. 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 may be surprising that a robust PCA algorithm can achieve a running time that is comparable to ordinary PCA. 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 $38,000$ 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 iterations.

In the above, all algorithms are implemented in `Matlab` and have seen the same level of code optimization. We also have a `C++` implementation of TGA that is significantly faster than the `Matlab` implementation (Fig. 10 (right)). Both implementations are available online[1].

To further emphasize the scalability of TGA, we compute the 30 leading components of the entire *Star Wars IV* movie. This consists of 179,415 RGB images in full DVD resolution (704×306). This requires 431 GB of memory using single precision. Computing 30 components takes 34 hours on an AMD Opteron 6378 (64 cores) with 1TB of memory, running at 2.4GHz, using the `C++` implementation of TGA. It takes 24 hours to compute the leading 30 principal components using a `C++` implementation of EM PCA with the same multi-threading scheme.

Figure 11 shows the leading odd-numbered components as computed by TGA with 50% trimming. We note some similarities between the estimated components and those attained from the analysis of natural images [45]. An application made possible by the scalability of the Grassmann

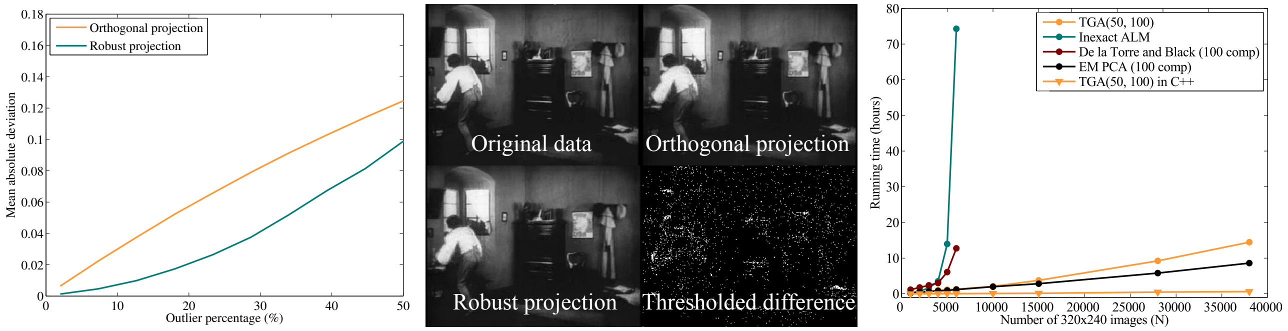

Fig. 10. *Left:* Mean absolute deviation of a reconstruction attained by different projection operators. Robust projection generally improves results. *Center:* Comparison between orthogonal and robust projection. The panel shows the data, its reconstruction using both orthogonal and robust projections, as well as the thresholded difference between the two projections methods; when pixels are represented using 8 bits (between 0 and 255), the maximal observed absolute pixel-difference is 1. *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 increases. TGA is comparable to EM PCA, while [5] and [6] are unable to handle more than 6000 observations.

average is the recent work on optical flow from Wulff & Black [46]. They learn a robust basis for full-frame optical flow using TGA on 180,000 flow fields computed from movies. Robustness is important because the computed flow fields contain outliers.

5 Discussion

The principle idea of this work is to rephrase linear subspace estimation directly in the space of subspaces. In particular, we suggest to compute the average subspace spanned by the data, *i.e.* the first moment on the Grassmann manifold. As standard PCA is concerned with the second moment of the data (covariance) our approach simplifies subspace estimation. This is also evident in the efficient algorithm and the ease with which robust extensions are developed.

The concept of studying the subspaces spanned by the data (rather than the data itself) opens several new directions which can be explored; here we discuss a few.

5.1 Choice of Metric

The general formulation of the average subspace (7) allows for any metric on the Grassmann manifold. We picked the Euclidean metric (10) as it both allows for closed-form spherical averages and makes it straightforward to extend to *pixel-wise* outliers. While these advantages are substantial, other metrics may be worth investigating.

A first question is: *does standard PCA compute an average subspace under a particular metric?* Standard PCA minimizes the reconstruction error (1), which can be rewritten as

$$\boldsymbol{q}_{\text{PCA}} = \operatorname*{arg\,min}_{\boldsymbol{v} \in S^{D-1}} \sum_{n=1}^{N} w_n^2 \left(1 - \cos^2\left(\theta_n(\boldsymbol{v})\right)\right), \tag{31}$$

where $w_n = \|\boldsymbol{x}_n\|$ and $\theta_n(\boldsymbol{v})$ denotes the angle between $\boldsymbol{x}_n$ and $\boldsymbol{v}$. This implies that PCA computes an average subspace under the metric

$$\text{dist}^2_{\text{PCA}}(\boldsymbol{x}_n, \boldsymbol{v}) = 1 - \cos^2\left(\theta_n(\boldsymbol{v})\right). \tag{32}$$

However, this is not a proper metric as it does not satisfy the triangle inequality. It is unclear if a proper metric exists under which standard PCA is the average subspace.

A perhaps more natural distance measure between subspaces is the angle between them; this is known as the *intrinsic metric*. Unfortunately, even on a sphere, the computation of an average under the intrinsic metric requires non-linear optimization [47]. This may limit the practical use of the intrinsic metric.

5.2 Probabilistic Interpretation

On the Grassmann manifold $\text{Gr}(1, D)$, it is not hard to develop distributions of the subspaces spanned by the data. We considered a von Mises-Fisher distribution as its mode coincides with the Grassmann average. Other choices are equally possible, though the choice of metric will influence which distributions are easy to work with.

Having a distribution over subspaces has several advantages not found in classic subspace estimators. It becomes easy to specify a prior over which subspaces are of interest; confidence intervals for the estimated subspace can be derived, allowing hypothesis testing; new random subspaces can be drawn from the distribution; *etc*.

5.3 Online Algorithms

We have not explored an online extension of the GA algorithm, which would only see one data point at a time. As we are only considering averages, we expect such online algorithms exist. As central limit theorems are available on the Grassmann manifold [38] it should be further possible to bound the performance of such online estimators.

5.4 Non-orthogonal Components?

In our formulation, we require orthogonal subspaces when we compute multiple components. This has computational advantages, but it may not always be ideal. If orthogonality is not a desired property, one can, instead of computing

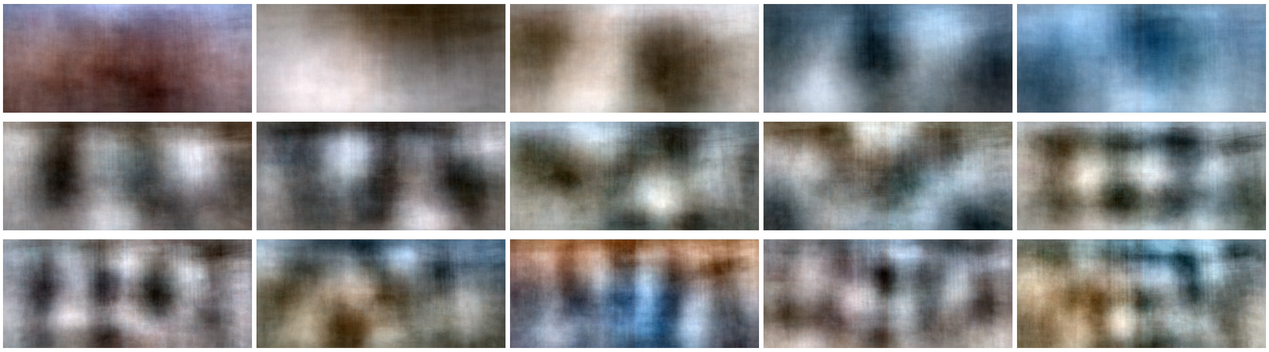

Fig. 11. The leading odd-numbered principal components of *Star Wars IV* as computed by TGA$(50\%, 30)$). The top row contains components 1, 3, 5, 7, and 9 from left to right, and so forth.

averages, seek modes of the distribution of subspaces. Finding such modes can be done, for example, with k-means or mean-shift [48] directly on the Grassmann manifold. These are straightforward exercises.

6 Conclusion

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.

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 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 computational complexity as the well-known scalable EM PCA [24], and empirical results show that TGA is not much slower while being substantially more robust. Our implementation is available online[1].

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 can even compute robust components of entire movies at full resolution in a reasonable time. Further, the ubiquity of PCA makes RGA relevant beyond computer vision; *e.g.* for large datasets found in biology, physics and weather forecasting. Finally, as our approach is based on standard building blocks like trimmed averages, it is very easy to speedup via parallelization.

Acknowledgments

Søren Hauberg is funded in part by the Villum Foundation and the Danish Council for Independent Research (DFF), Natural Sciences; Aasa Feragen is funded in part by the DFF, Technology and Production Sciences. The authors are grateful to anonymous reviewers for helpful feedback.

References

[1] N. Campbell, "Robust procedures in multivariate analysis I: Robust covariance estimation," *Applied Statistics*, vol. 29, no. 3, 1980.

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

[3] J. Feng, H. Xu, and S. Yan, "Robust PCA in high-dimension: A deterministic approach," in *International Conference on Machine Learning*, 2012.

[4] L. Xu and A. L. Yuille, "Robust principal component analysis by self-organizing rules based on statistical physics approach," *IEEE Transactions on Neural Networks*, vol. 6, no. 1, pp. 131–143, 1995.

[5] E. J. Candes, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *Journal of the Association for Computing Machinery*, vol. 58, no. 3, 2011.

[6] F. De la Torre and M. J. Black, "A framework for robust subspace learning," *International Journal of Computer Vision*, vol. 54, pp. 117–142, 2003.

[7] S. Hauberg, A. Feragen, and M. J. Black, "Grassmann averages for scalable robust PCA," in *Computer Vision and Pattern Recognition (CVPR)*, 2014.

[8] I. Jolliffe, *Principal component analysis*. Springer, 2002.

[9] F. H. Ruymgaart, "A robust principal component analysis," *Journal of Multivariate Analysis*, vol. 11, pp. 485–497, 1981.

[10] H. Aanæs, R. Fisker, K. Åström, and J. Carstensen, "Robust factorization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1215–1225, Sep 2002.

[11] C. Ding, D. Zhou, X. He, and H. Zha, "R_1-PCA: rotational invariant l1-norm principal component analysis for robust subspace factorization," in *International Conference on Machine Learning*, 2006, pp. 281–288.

[12] N. Kwak, "Principal component analysis based on l1-norm maximization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 9, pp. 1672–1680, Sept 2008.

[13] M. McCoy and J. A. Tropp, "Two proposals for robust pca using semidefinite programming," *Electronic Journal of Statistics*, vol. 5, pp. 1123–1160, 2011.

[14] A. Naor, O. Regev, and T. Vidick, "Efficient rounding for the noncommutative grothendieck inequality," in *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, ser. STOC '13. ACM, 2013, pp. 71–80.

[15] S. Geman and D. McClure, "Statistical methods for tomographic image reconstruction," *Bulletin of the International Statistical Institute*, vol. 52, no. 4, pp. 5–21, 1987.

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

[17] L. W. Z. Lin, M. Chen and Y. Ma, "The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices," UILU-ENG-09-2215, Tech. Rep., 2009.

[18] R. Tron and R. Vidal, "Distributed computer vision algorithms through distributed averaging," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2011, pp. 57–63.

[19] P. Netrapalli, N. U N, S. Sanghavi, A. Anandkumar, and P. Jain, "Non-convex robust pca," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 1107–1115.

[20] L. W. Mackey, A. S. Talwalkar, and M. I. Jordan, "Divide-and-conquer matrix factorization," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., 2011, pp. 1134–1142.

[21] R. Liu, Z. Lin, Z. Su, and J. Gao, "Linear time principal component pursuit and its extensions using l1 filtering," *Neurocomputing*, vol. 142, pp. 529 – 541, 2014, SI Computational Intelligence Techniques for New Product Development.

[22] Y. Mu, J. Dong, X. Yuan, and S. Yan, "Accelerated low-rank visual recovery by random projection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2011, pp. 2609–2616.

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

[24] S. T. Roweis, "EM algorithms for PCA and SPCA," in *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, 1998.

[25] Y. M. Lui, "Advances in matrix manifolds for computer vision," *Image and Vision Computing*, vol. 30, pp. 380–388, 2012.

[26] R. Vidal, Y. Ma, , and S. Sastry, "Generalized principal component analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 12, pp. 1945–1959, 2005.

[27] P. Favaro, R. Vidal, and A. Ravichandran, "A closed form solution to robust subspace estimation and clustering," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 1801–1807.

[28] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 11, pp. 2765–2781, 2013.

[29] G. Liu, Z. Lin, and Y. Yu, "Robust subspace segmentation by low-rank representation," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 663–670.

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

[31] M. R. Bridson and A. Haefliger, *Metric spaces of non-positive curvature*. Springer, 1999.

[32] K. V. Mardia and P. E. Jupp, *Directional Statistics*. Wiley, 1999.

[33] P. J. Huber, "Projection pursuit," *Annals of Statistics*, vol. 13, no. 2, pp. 435–475, 1985.

[34] F. Wang, "Space and space-time modeling of directional data," Ph.D. dissertation, Duke University, 2013.

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

[36] F. C. Leone, L. S. Nelson, and R. B. Nottingham, "The folded normal distribution," *Technometrics*, vol. 3, no. 4, pp. 543–550, 1961.

[37] N. I. Embleton, T. Fisher, and B. J. J. Lewis, *Statistical analysis of spherical data*. Cambridge University Press, 1993.

[38] R. Bhattacharya and V. Patrangenaru, "Large sample theory of intrinsic and extrinsic sample means on manifolds," *The Annals of Statistics*, vol. 31, no. 1, pp. 1–29, 2003.

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

[40] M. J. Black and A. D. Jepson, "Eigentracking: Robust matching and tracking of articulated objects using a view-based representation," *International Journal of Computer Vision*, vol. 26, pp. 63–84, 1998.

[41] H. Ji, S. Huang, Z. Shen, and Y. H. Xu, "Robust video restoration by joint sparse and low rank matrix approximation," *SIAM Journal on Imaging Sciences (SIIMS)*, vol. 4, no. 4, pp. 1122–1142, 2011.

[42] L. Li, W. Huang, I. Gu, and Q. Tian, "Statistical modeling of complex backgrounds for foreground object detection," *IEEE Transactions on Image Processing*, vol. 13, no. 11, pp. 1459–1472, 2004.

[43] A. Georghiades, P. Belhumeur, and D. Kriegman, "From few to many: Illumination cone models for face recognition under variable lighting and pose," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 643–660, 2001.

[44] R. Basri and D. Jacobs, "Lambertian reflectance and linear subspaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 2, pp. 218–233, 2003.

[45] P. J. Hancock, R. J. Baddeley, and L. S. Smith, "The principal components of natural images," *Network: computation in neural systems*, vol. 3, no. 1, pp. 61–70, 1992.

[46] J. Wulff and M. J. Black, "Efficient sparse-to-dense optical flow estimation using a learned basis and layers," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) 2015*, Jun. 2015.

[47] X. Pennec, "Probabilities and statistics on Riemannian manifolds: Basic tools for geometric measurements," in *Proceedings of Nonlinear Signal and Image Processing*, 1999, pp. 194–198.

[48] H. Çetingul and R. Vidal, "Intrinsic mean shift for clustering on Stiefel and Grassmann manifolds," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2009, pp. 1896–1902.

Søren Hauberg received his Ph.D. in computer science from the Univ. of Copenhagen in 2011. He has been a visiting scholar at UC Berkeley (2010), and a post doc at the Perceiving Systems department at the Max Planck Institute for Intelligent Systems in Tübingen, Germany (2012–2014). He is currently a post doc at the Section for Cognitive Systems, at the technical Univ. of Denmark. His research is at the interplay of geometry and statistics.

Aasa Feragen holds a Ph.D. in mathematics from the Univ. of Helsinki (2010) and has since been post doc and later Associate Professor at DIKU at the Univ. of Copenhagen. From 2012 to 2014 she was a post doc in the Machine Learning and Computational Biology Research Group at the Max Planck institute for Developmental Biology and Max Planck Institute for Intelligent Systems, Tübingen, Germany. Her research is on geometric modeling for imaging and vision.

Raffi Enficiaud received his M.Sc. from the National Institute of Telecommunications, Paris-Evry, France, in 2000, and his Ph.D. from Paris School of Mines, France, in 2007. He worked as a research engineer for DxO Labs, Paris and INRIA Paris-Rocquencourt (2007-2013). He is currently running the Software Workshop at the Max Planck Institute for Intelligent Systems, Tübingen, Germany. His research interests include computer vision and image processing.

Michael J. Black received his B.Sc. from the Univ. of British Columbia (1985), his M.S. from Stanford (1989), and his Ph.D. from Yale (1992). After research at NASA Ames and the Univ. of Toronto, he was at Xerox PARC as a member of research staff and area manager (1993-2000). From 2000 to 2010 he was on the faculty of Brown Univ. (Assoc. Prof. 2000-2004, Prof. 2004-2010). Since 2011 his one of the founding director at the Max Planck Institute for Intelligent Systems in Tübingen, Germany, where he leads the Perceiving Systems department. Significant awards include the 2010 Koenderink Prize (ECCV) and the 2013 Helmholtz Prize (ICCV). He is a foreign member of the Royal Swedish Academy of Sciences and is a co-founder and board member of Body Labs Inc.