Grassmann Averages for Scalable Robust PCA — Supplementary Material —

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

A. Overview

This document contains the supplementary material for *Grassmann Averages for Scalable Robust PCA* [4]. Here we provide

- pseudo-code for the different algorithms (Sec. B);
- a proof of convergence (Sec. C);
- a more detailed look at the impact of the trimming parameter in TGA (Sec. D);
- some thoughts on statistical efficiency in the context of Fig. 6 in the paper (Sec. E);
- an empirical verification of Theorem 2 stating that the Grassmann Average conincides with PCA for Gaussian data (Sec. F);
- an empirical study of the influence of outliers when the inliers are Gaussian (Sec. G);
- an investigation on the impact of robustness in projections (Sec. H);
- some notes on the comparison with Inexact ALM [2, 10] (Sec. I);
- a brief discussion of the use of extrinsic averages (Sec. J); and
- the images of the 20 leading components of *Star Wars IV* as estimated by EM PCA [9] and TGA (Sec. K).

The supplements further contain video material.

B. Pseudo-Code

In this section we provide pseudo-code for the different algorithms used in the paper; see Fig. 7. First, we point to Algorithm 1, which depicts EM PCA. This is very similar to the Grassmann Average (GA) algorithm, which is depicted in Algorithm 2. But, the derivation of GA results in an important difference that translates into the use of a `sign` operator in Algorithm 2. It is interesting that this change already gives improved robustness as shown and derived in the main paper.

In practice, we further find that the GA algorithm typically requires substantially fewer iterations to converge than EM PCA — in our experience this is because EM PCA "zig zags" around the optimum, which GA avoids as it only has a discrete set of solutions (it only need to estimate the sign given to each observation). This also ensure that the exact optimum is located, which prevents accumulation of errors when multiple components are estimated.

Algorithm 3 shows the *Robust Grassmann Average (RGA)*. This is slightly more complicated than the two previous algorithms, but still only requires a few lines of code. RGA depends on a function `robust_mean`, which should be defined on a per-application basis. For most experiments in the paper we use a version of RGA that we call the *Trimmed Grassmann Average (TGA)*, where the `robust_mean` simply computes the trimmed mean on a per-element basis.

C. A Convergence Proof

Here we prove that the Grassmann Average algorithm converges to a local optima in finite time. Before stating the result we remind the reader that GA seeks to optimize

$$\begin{aligned}\operatorname*{arg\,min}_{[\mathbf{q}]\in \mathrm{Gr}(1,D)} \sum_{n=1}^{N} w_n \mathrm{dist}^2_{\mathrm{Gr}(1,D)}([\mathbf{u}_n],[\mathbf{q}]) \qquad (14)\\ = \operatorname*{arg\,max}_{\mathbf{q}\in S^{D-1}} \sum_{n=1}^{N} w_n \left|\mathbf{u}_n^T\mathbf{q}\right| \qquad (15)\\ = \operatorname*{arg\,max}_{\mathbf{q}\in S^{D-1}} \sum_{n=1}^{N} \left|\mathbf{x}_n^T\mathbf{q}\right|, \qquad (16)\end{aligned}$$

where $w_n = \|\mathbf{x}_n\|$.

Theorem 3 *Sample the data $\mathbf{x}_{1:N}$ from a probability measure which is absolutely continuous with respect to the*

*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.

Algorithm 1: EM PCA

$\mathbf{q}_0 \leftarrow$ random unit vector
$i \leftarrow 1$
while $\|\mathbf{q}_i - \mathbf{q}_{i-1}\| \neq 0$ **do**
  $\tilde{\mathbf{q}} \leftarrow \sum_{n=1}^{N} (\mathbf{x}_n^T \mathbf{q}_{i-1}) \mathbf{x}_n$
  $\mathbf{q}_i \leftarrow \frac{\tilde{\mathbf{q}}}{\|\tilde{\mathbf{q}}\|}$
  $i \leftarrow i + 1$
end while

Algorithm 2: Grassmann Average (GA)

$\mathbf{q}_0 \leftarrow$ random unit vector
$i \leftarrow 1$
while $\|\mathbf{q}_i - \mathbf{q}_{i-1}\| \neq 0$ **do**
  $\tilde{\mathbf{q}} \leftarrow \sum_{n=1}^{N} \operatorname{sign}(\mathbf{x}_n^T \mathbf{q}_{i-1}) \mathbf{x}_n$
  $\mathbf{q}_i \leftarrow \frac{\tilde{\mathbf{q}}}{\|\tilde{\mathbf{q}}\|}$
  $i \leftarrow i + 1$
end while

Algorithm 3: Robust Grassmann Average (RGA)

$\mathbf{q}_0 \leftarrow$ random unit vector
$i \leftarrow 1$
while $\|\mathbf{q}_i - \mathbf{q}_{i-1}\| \neq 0$ **do**
  $\tilde{\mathbf{x}}_n \leftarrow \operatorname{sign}(\mathbf{x}_n^T \mathbf{q}_{i-1}) \mathbf{x}_n \quad \forall n$
  $\tilde{\mathbf{q}} \leftarrow \texttt{robust\_mean}(\tilde{\mathbf{x}}_{1:N})$
  $\mathbf{q}_i \leftarrow \frac{\tilde{\mathbf{q}}}{\|\tilde{\mathbf{q}}\|}$
  $i \leftarrow i + 1$
end while

Figure 7. Pseudo-code for different algorithms.

Lebesgue measure in $\mathbb{R}^D$. With probability 1, the GA algorithm converges to a local optimum of Eq. 16 in finite time.

The proof, while independently developed, follows that of Kwak [6] and is included only for completeness.

To prove the theorem, we split the statement into two parts from which the result follows.

Lemma 4 *Sample the data $\mathbf{x}_{1:N}$ from a probability measure which is absolutely continuous with respect to the Lebesgue measure in $\mathbb{R}^D$. With probability 1, GA converges in finite time.*

Proof GA iteratively computes the following weighted mean until convergence:

$$\mathbf{q}_i = \arg\max_{\|\mathbf{q}\|=1} \sum_{n=1}^{N} w_n \mathbf{u}_{n,i}^T \mathbf{q}, \tag{17}$$

where

$$\mathbf{u}_{n,i} = \operatorname{sign}(\mathbf{x}_n^T \mathbf{q}_{i-1}) \mathbf{u}_n. \tag{18}$$

We show that, with probability 1, there exists $M \in \mathbb{N}_0$ such that $\mathbf{q}_i = \mathbf{q}_{i'}$ for all $i, i' \geq M$, and

$$\max_{\|\mathbf{q}\|=1} \sum_{n=1}^{N} w_n \mathbf{u}_{n,i}^T \mathbf{q} < \max_{\|\mathbf{q}\|=1} \sum_{n=1}^{N} w_n \mathbf{u}_{n,i'}^T \mathbf{q}, \tag{19}$$

whenever $i < i' < M$. That is, the value of (17) increases strictly for steps 1 to $(M-1)$. In the i^{th} iteration, the set of points $\mathbf{u}_{1:N,i}$ orthogonal to $\mathbf{q}_{i-1}$ has Lebesgue measure 0, so with probability 1, $\mathbf{u}_{n,i}^T \mathbf{q}_{i-1} \neq 0 \ \forall n$. If $\mathbf{u}_{n,i}^T \mathbf{q}_{i-1} > 0$ for all n, then the algorithm has converged and $i \geq M$. Otherwise, there exists some n for which $\mathbf{u}_{n,i}^T \mathbf{q}_{i-1} < 0$ and it remains to prove that

$$\max_{\|\mathbf{q}\|=1} \sum_{n=1}^{N} w_n \mathbf{u}_{n,i+1}^T \mathbf{q} > \max_{\|\mathbf{q}\|=1} \sum_{n=1}^{N} w_n \mathbf{u}_{n,i}^T \mathbf{q}. \tag{20}$$

In order to see this, note that we have:

$$\sum_{n=1}^{N} w_n \mathbf{u}_{n,i+1}^T \mathbf{q}_i > \sum_{n=1}^{N} w_n \mathbf{u}_{n,i}^T \mathbf{q}_i \tag{21}$$

since $\mathbf{u}_{n,i} \neq \mathbf{u}_{n,i+1}$ if and only if $\mathbf{u}_{n,i}^T \mathbf{q}_i < 0$. Because $\mathbf{u}_{n,i+1}^T \mathbf{q}_i > 0$ we must have $\mathbf{u}_{n,i}^T \mathbf{q}_i < \mathbf{u}_{n,i+1}^T \mathbf{q}_i$. But then, certainly,

$$\max_{\|\mathbf{q}\|=1} \sum_{n=1}^{N} w_n \mathbf{u}_{n,i+1}^T \mathbf{q} > \sum_{n=1}^{N} w_n \mathbf{u}_{n,i+1}^T \mathbf{q}_i \tag{22}$$

$$> \sum_{n=1}^{N} w_n \mathbf{u}_{n,i}^T \mathbf{q}_i \tag{23}$$

$$= \max_{\|\mathbf{q}\|=1} \sum_{n=1}^{N} w_n \mathbf{u}_{n,i}^T \mathbf{q}, \tag{24}$$

which proves (20). The fact that M exists and GA converges in a finite number of steps follows from the fact that there are only finitely many ways to change the sign of $\mathbf{u}_{1:N}$, each giving a fixed value of (17), so there cannot be an infinite sequence of increasing values. □

Remark 1 The "with probability 1" is a necessary condition. Pathological examples exist where GA fails, e.g. for $\mathbf{x}_{1:N} = \{\mathbf{x}, -\mathbf{x}\}$, initializing GA with $\mathbf{q}_0$ orthogonal to $\mathbf{x}$.

Remark 2 While the algorithm does converge in finite time, the number of iterations may in theory be very large. The notion of "finite time" is therefore mostly of theoretical interest, though in practice the algorithm generally converges in fewer iterations than *e.g.* EM PCA.

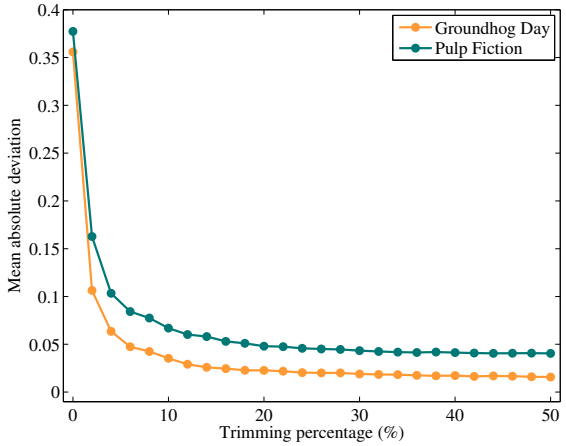


Figure 8. The mean absolute deviation measured at noisy pixels for two sequences.

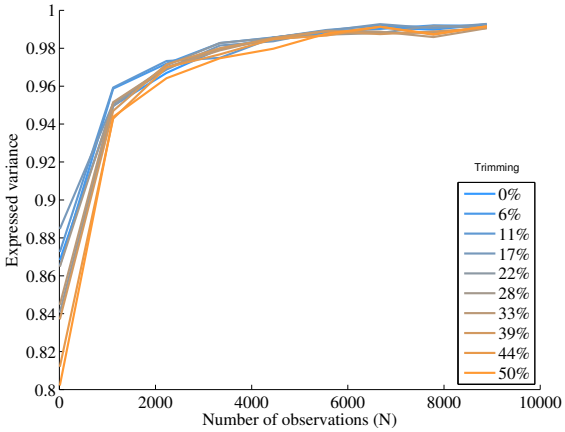


Figure 9. The expressed variance as a function of the number of observations N. Here the dimensionality is fixed at $D = 100$. The different curves correspond to different levels of trimming.

Lemma 5 *Under the assumptions of Theorem 3, GA finds a local optimum of Eq. 16.*

Proof Let $\mathbf{q}$ denote the point of convergence of GA and let $\mathbf{u}_{1:N}$ denote the representations picked by the algorithm. If $\tilde{\mathbf{q}}$ is any small perturbation of $\mathbf{q}$ then, with probability 1, the algorithm will pick the same representations $\mathbf{u}_{1:N}$ when started from $\tilde{\mathbf{q}}$ and the algorithm returns $\mathbf{q}$. □

D. Impact of the Trimming Parameter

Our algorithm only has two parameters – the number of robust principal components and the amount of trimming to use. With 0% trimming the algorithm is only as robust as the original GA algorithm (still more robust than PCA); at 50% we have maximal robustness for this approach.

In Sec. 4.1 of the main paper we perform an experiment on images where we add approximately 5% noise from *Nosferatu* to scenes from the two contemporary Hollywood movies *Pulp Fiction* and *Groundhog Day*. As we change the value of the trimming parameter of TGA we will get different results. Fig. 8 show the impact. When we trim more than 10–15% we see substantial improvements compared to no trimming.

E. Statistical Efficiency vs. Robustness

In Sec. 4.4 of the paper we conduct an experiment on robustness when the outliers appear at the vector-level; Fig. 6 in the paper provides the results. Here it is worth noting that TGA(25,1) performs better than TGA(50,1) when the number of outliers is small. It is a well-known [5] that medians require more observations to give accurate answers than *e.g.* 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 trade-off for robust estimators and it is not surprising that the TGA is affected by this as well. One should, thus, use as low a trimming parameter as the data allows to get as efficient an estimator as possible; see *e.g.* [5] for further discussions on the matter.

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

In the 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.

F. Performance on Gaussian Data

Theorem 2 in the main paper says that for Gaussian data, the *expected* component as estimated by GA conincides with the principal component as 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 eigen decomposition. Figure 10 show 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 lowered statistical efficiency, *c.f.* Sec. E.

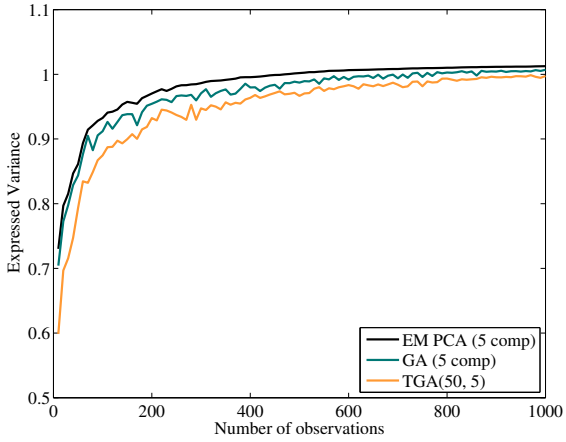


Figure 10. Expressed variance as a function of number of samples from a known Gaussian distribution. The provides results are for $D = 30$ dimensional data, and have been averaged over 5 expriments with randomly generated covariance matrices.

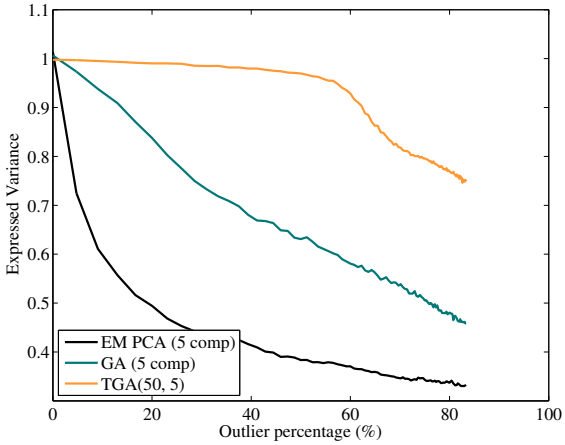


Figure 11. Expressed variance for Gaussian inliers with an increasing number of outliers. The provides results are for $D = 30$ dimensional data, and have been averaged over 50 expriments with randomly generated covariance matrices.

G. Performance with Synthetic Outliers

We further measure the performance of the different methods when the data is Gaussian with added outliers. We generate a random covariance matrix Σ and sample from the corresponding Gaussian distribution. We add an increasing number of outliers to this data, which is sampled from a Gaussian distribution with a mean placed along the eigenvector of Σ with the smallest eigenvalue. This gives a consistent bias away from the principal components of the inliers. Figure 11 show the experessed 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 with lower speed. TGA, on the other hand, is very robust and attains a strong performance until the number of outliers exceed that of the inliers.

H. Robust vs. Orthogonal Projections

In the experiments, we estimate robust subspaces with a basis $\mathbf{Q} \in \mathbb{R}^{D \times K}$, 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,

$$\mathbf{P} = \mathbf{X}\mathbf{Q}\mathbf{Q}^T, \tag{25}$$

where $\mathbf{X} \in \mathbb{R}^{N \times D}$ is the data matrix. This projection finds the point $\mathbf{P}$ in the subspace that is closest to the original point $\mathbf{X}$. This is a least-squares optimization problem where Eq. 25 is the solution [1]. 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 [1] using the Geman-McClure error functions. This projection operator is then optimized using gradient descent.

We have experimented with the robust operator, and have found that it generally improves results. First, we consider a synthetic experiment: we select an increasing number of random pixels from the *Groundhog Day* sequence and make them white, thereby adding consistent 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 projection from the noise-free data and those from noisy data. Figure 12 show that the robust projection generally performs 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. 13a–c, where the different projections appear to give nearly identical results. Minor differences between the methods do appear and Fig. 13d show 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 were 1.

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 1) the fairly small amount of outliers in the images; and 2) the limited range of the outliers, *i.e.* they have values between 0 and 255.

I. Notes on the Regualirization of Inexact ALM

The Inexact ALM [2, 10] algorithm relies on a regularization parameter λ, which indrectly controls the rank of the

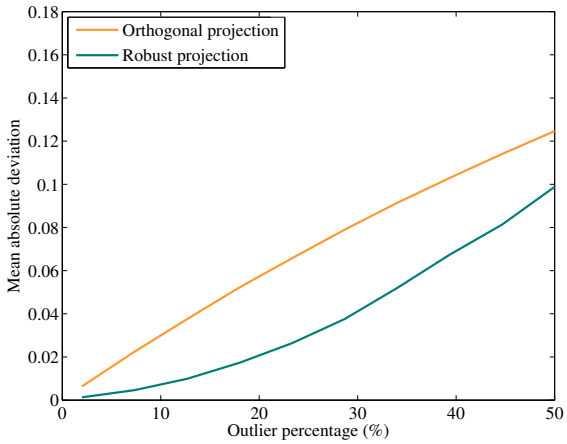
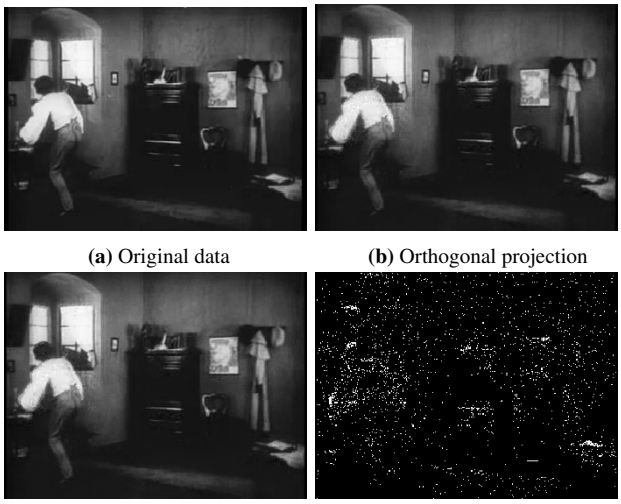


Figure 12. Mean absolute deviation of a reconstruction attained by different projection operators. Robust projection generally improves results.



(a) Original data **(b)** Orthogonal projection

(c) Robust projection **(d)** Thresholded difference

Figure 13. Comparison between orthogonal and robust projection. (a) The data. (b) The reconstruction of (a) using orthogonal projection onto a 80 dimensional subspace estimated using the trimmed Grassmann average. (c) The reconstruction using a robust projection onto the same subspace. (d) The pixels where the two reconstructions differ; when pixels are represented using 8 bits (between 0 and 255), the maximal observed absolute pixel-difference is 1.

resulting data reconstruction. In all experiments we use the default setting of this parameter as provides by the reference implementation. The GA and TGA algorithms, on the other hand, are given a suitable number of components, which directly controls the rank of the reconstructed data matrices. A completely fair comparison between the GA algorithms and Inexact ALM, should choose λ such that the resulting data reconstruction has same rank as those provided by the GA algorithms. We opted against this as we found tuning of λ to be rather difficult as the relationship between λ and

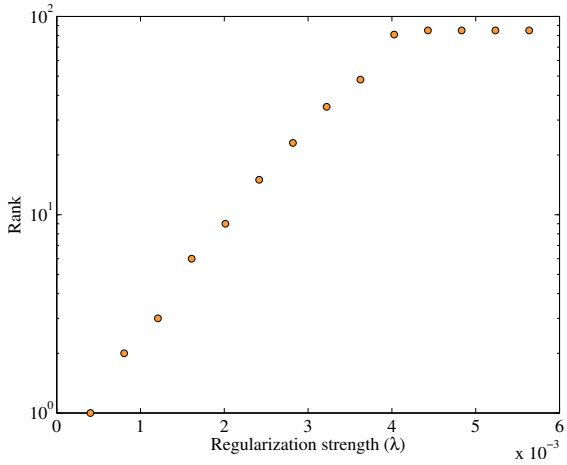


Figure 14. The rank of reconstructed data matrices according to Inexact ALM as a function of the regularization parameter λ. *Note the logarithmic scale.*

the resulting rank was sensitive. Figure 14 show the rank of reconstructed data matrices as a function of λ (note the logarithmic scale); here we used the data from Sec. 4.1 in the paper. As can be seen, the rank grows exponentially fast with λ until full rank is attained. Combined with the large running time of Inexact ALM we found the tuning of λ to be impractical.

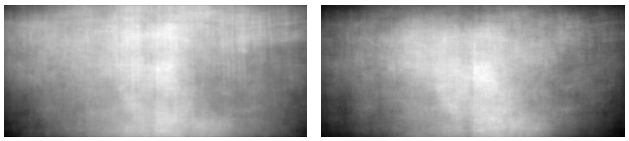
J. Extrinsic vs. Intrinsic Averages

In the computation of the Grassmann average we rely heavily on closed-form computations of spherical averages. This average is also known as an *extrinsic average* as it relies on the Euclidean metric of the ambient space of the unit sphere. One can argue, that this might not be the best metric to use on the sphere and that the *intrinsic metric*, *i.e.* the angle between vectors, is more natural. We did not investigate this further as the computation of averages under the intrinsic metric is not available in closed-form and requires non-linear optimization [7, 8].

K. Star Wars

To show the scalability of TGA we have computed the leading 20 components of the entire *Star Wars IV* movie on a desktop computer. To make this practical we work only with grayscale images and reduced the image resolution to 352×153.

Fig. 15 shows the average and median images of the movie. Likewise Fig. 16 shows the 20 leading ordinary principal components as computed by EM PCA; and Fig. 17 show the leading 20 components as computed by TGA with 50% trimming. It is interesting to note the the robust components appear more focused on specific regions of the images — in particular close to the center where most of the action



Mean **Pixel-wise median**

Figure 15. Pixel-wise mean and median of the entire *Star Wars IV* movie.

appears. One can also clearly see the effects of outliers in some of the non-robust components; *e.g*. in the bottom row of Fig. 16 far left and right images. It is interesting to note some similarities between the estimated components and those attained from analysis of natural images [3].

References

[1] 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. 4

[2] E. J. Candes, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the Association for Computing Machinery*, 58(3), 2011. 1, 4

[3] P. J. Hancock, R. J. Baddeley, and L. S. Smith. The principal components of natural images. *Network: computation in neural systems*, 3(1):61–70, 1992. 6

[4] S. Hauberg, A. Feragen, and M. J. Black. Grassmann averages for scalable robust pca. In *Computer Vision and Pattern Recognition (CVPR)*, 2014. 1

[5] P. J. Huber. *Robust Statistics*. Wiley: New York, 1981. 3

[6] N. Kwak. Principal component analysis based on l1-norm maximization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(9):1672–1680, Sept 2008. 2

[7] K. V. Mardia and P. E. Jupp. *Directional Statistics*. Wiley, 1999. 5

[8] X. Pennec. Probabilities and statistics on Riemannian manifolds: Basic tools for geometric measurements. In *Proceedings of Nonlinear Signal and Image Processing*, pages 194–198, 1999. 5

[9] S. T. Roweis. EM algorithms for PCA and SPCA. In *NIPS*. MIT Press, 1998. 1

[10] 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. 1, 4

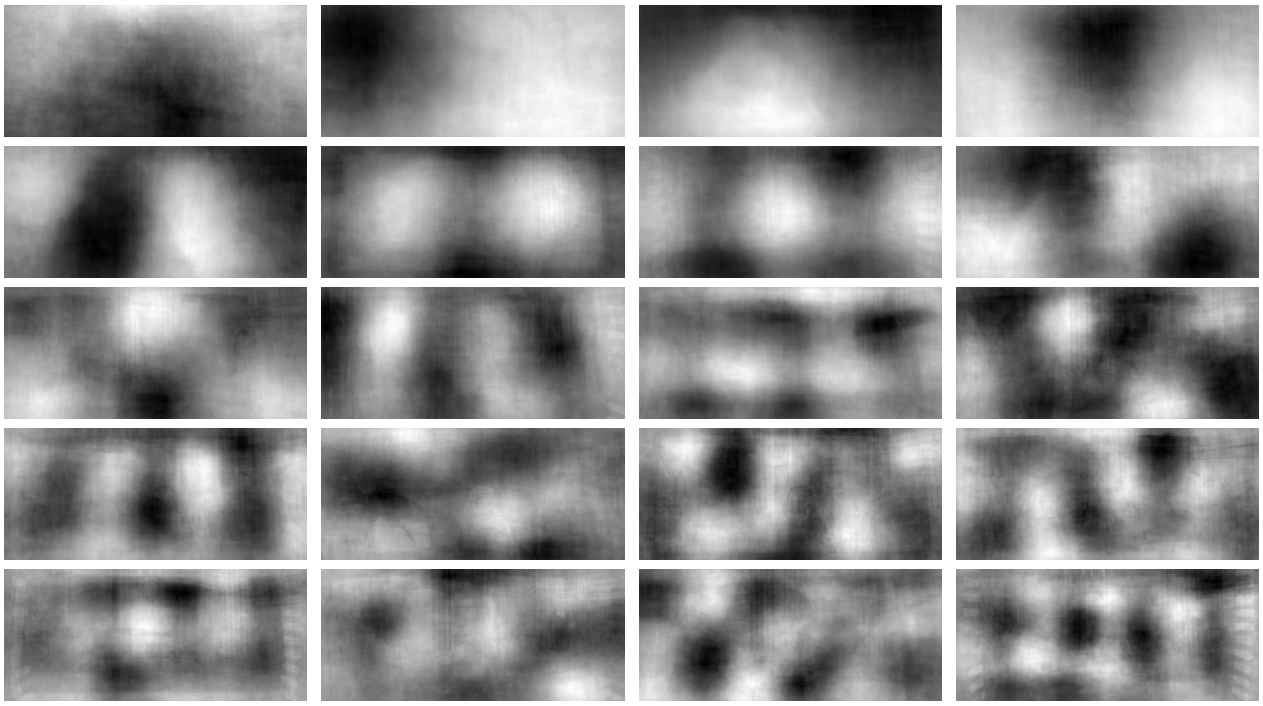


Figure 16. The leading 20 principal components of *Star Wars IV* as computed by EM PCA. The top row contains components 1–4 from left to right; the second row components 5–8 and so forth.

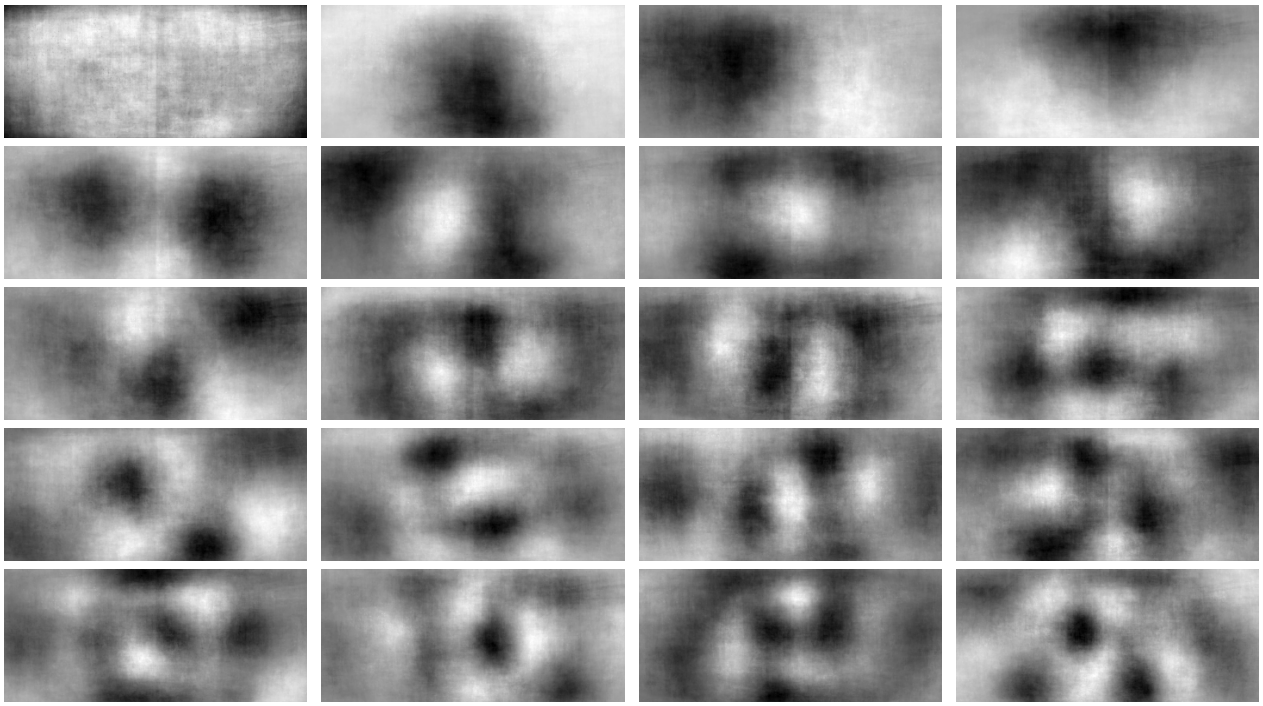


Figure 17. The leading 20 principal components of *Star Wars IV* as computed by TGA(50%, 20). The top row contains components 1–4 from left to right; the second row components 5–8 and so forth.