Grassmann Averages for Scalable Robust PCA

Søren Hauberg, Aasa Feragen and Michael J. Black

DTU Compute
Technical University of Denmark

Dept. of Computer Science
University of Copenhagen

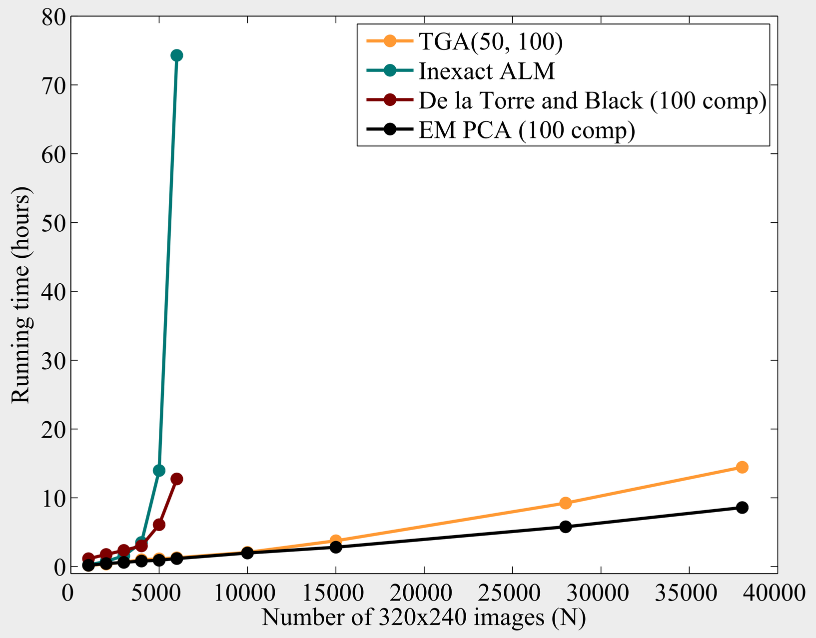
MPI for Intelligent Systems
Tübingen, Germany

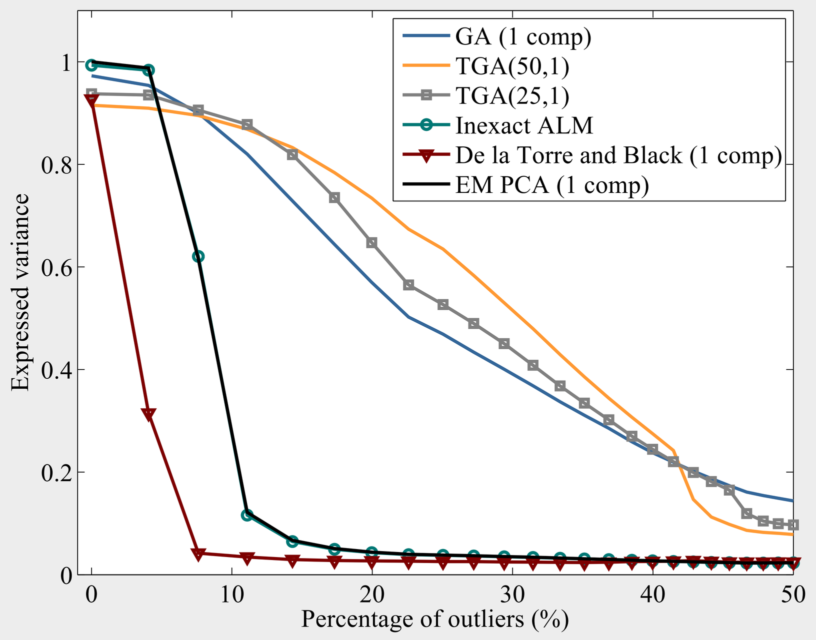
1) Motivation: Scalable robust statistics

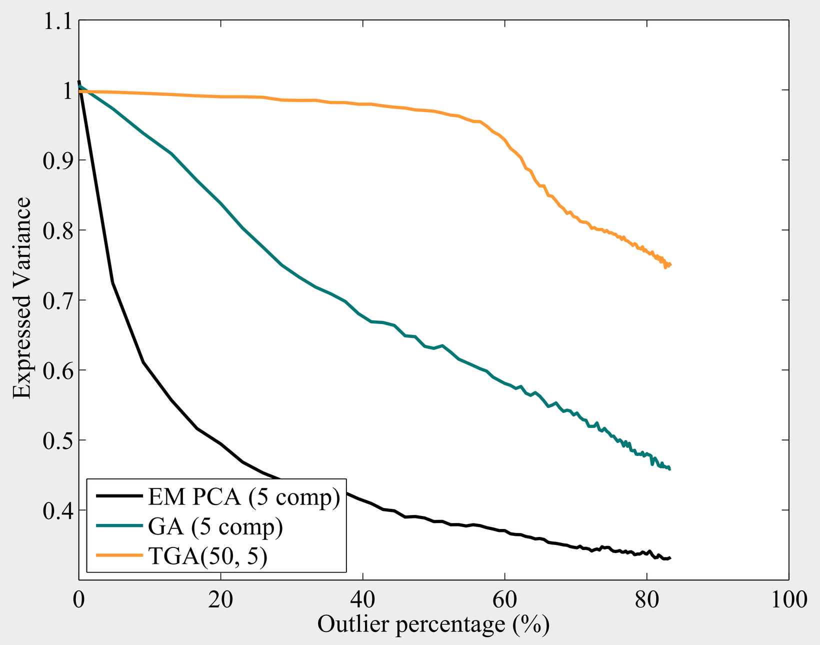
We are collecting increasing amounts of data in a, largely, automated way. This large-scale harvesting increases the odds of collecting unwanted data. In other words we should expect increasing amounts of outliers when automating data collection.

We, thus, need algorithms which can both cope with large amounts of data and deal with many outliers.

We investigate large-scale robust PCA.

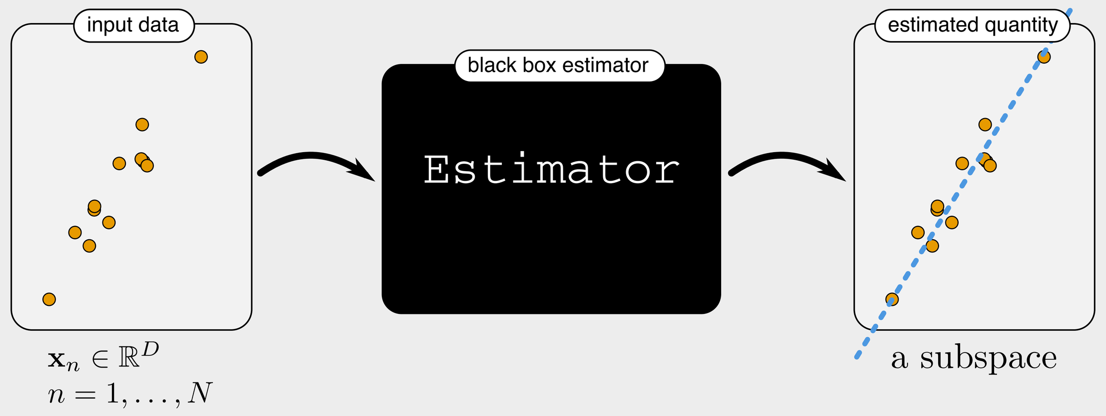


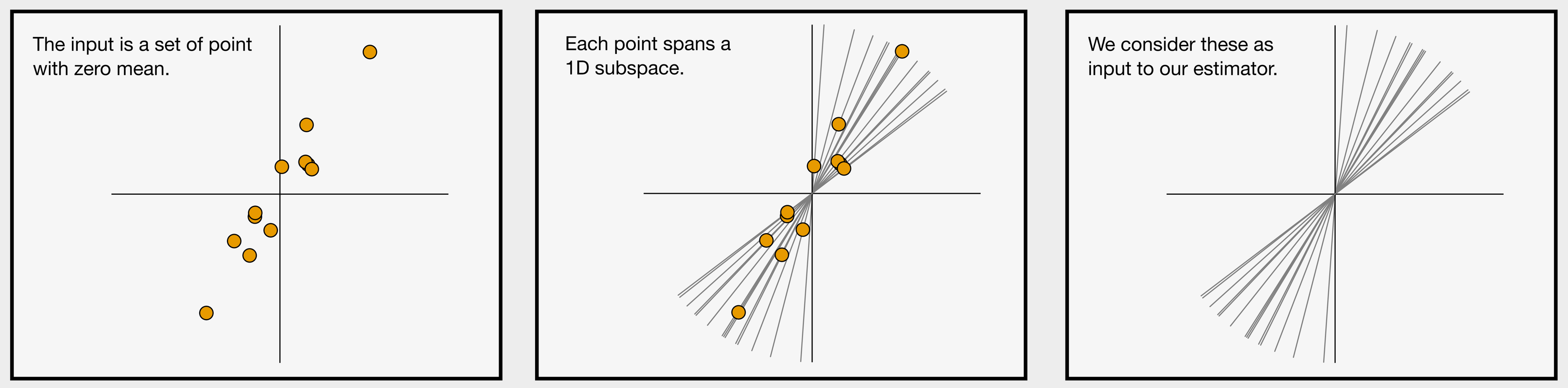


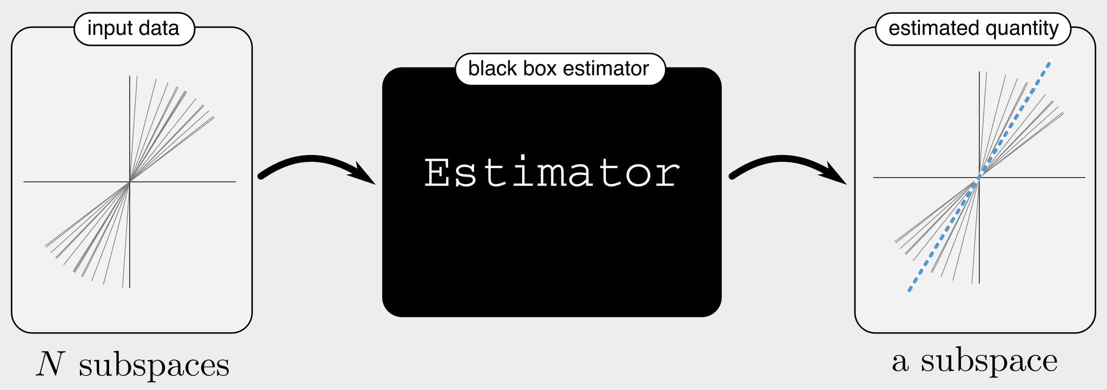


2) Idea #1: Match input and output types

The simplest and most well-understood statistical estimators are those where the "type" of the input and output match. Example: the standard average (input: points; output: a point).



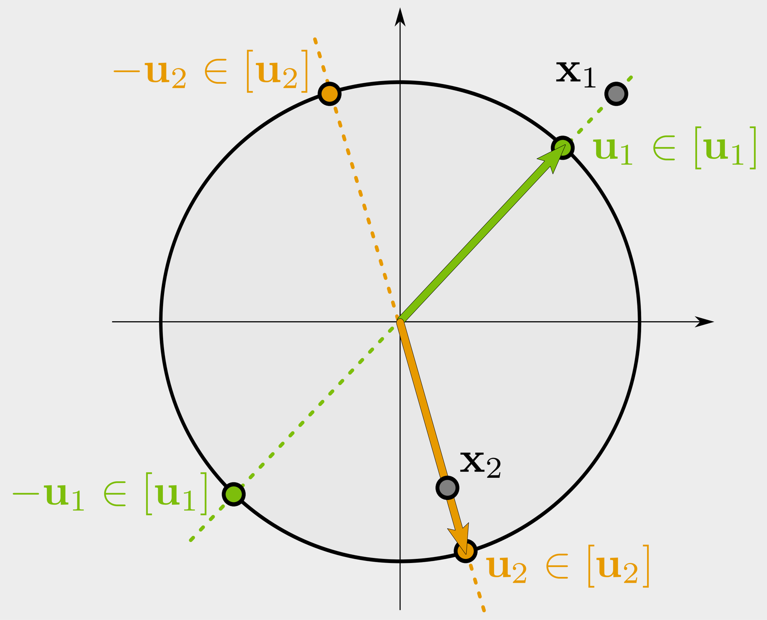




Corresponding types will simplify both math and algorithms. Many estimators are now easily phrased.

We study the average subspace.

3) Representing subspaces



The most obvious representation of 1D subspaces is a unit vector spanning the space. Changing the sign of a unit vector does not change the space being spanned, so our representation has an unknown sign.

This is a simple instance of the general Grassmann manifold, which represents higher dimensional subspaces.

notation

$\mathbf{u}_n = \frac{\mathbf{x}_n}{\|\mathbf{x}_n\|}$

$[\mathbf{u}_n] = \{-\mathbf{u}_n, \mathbf{u}_n\}$

4) Grassmann average algorithm

A first result for deriving an algorithm is

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

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

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

With an simple Euclidean distance measure

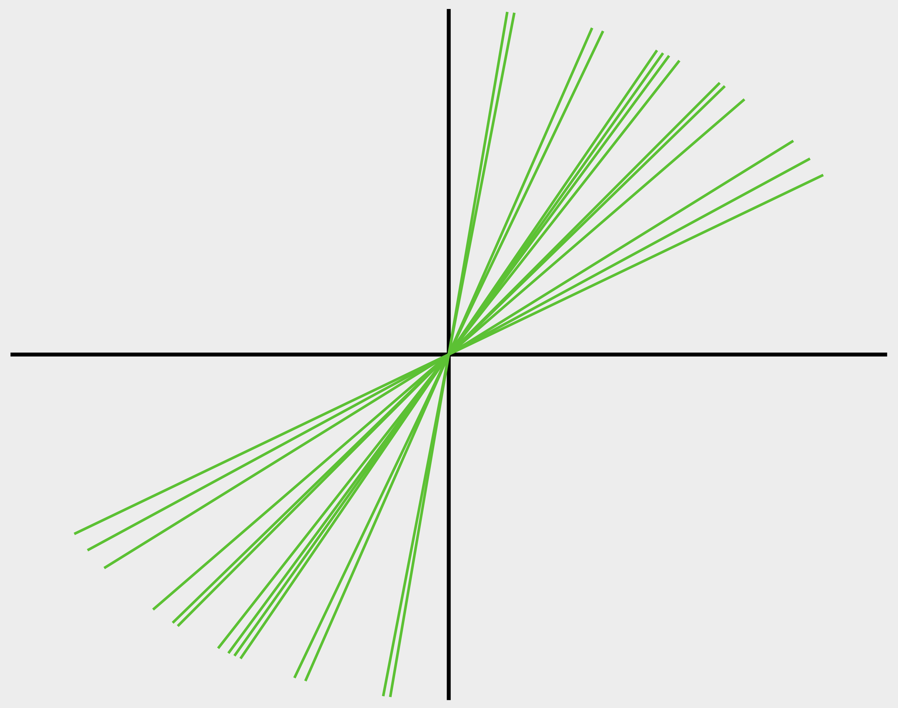
$$\mathrm{dist}^2_{S^{D-1}}(\mathbf{u}_1, \mathbf{u}_2) = \frac{1}{2}\|\mathbf{u}_1 - \mathbf{u}_2\|^2 = 1 - \mathbf{u}_1^T\mathbf{u}_2$$

the following algorithm naturally appears

Algorithm: Grassmann Average (GA)

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

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



The algorithm optimises this robust energy:

$$[\mathbf{q}] = \underset{\mathbf{v}\in S^{D-1}}{\arg\max} \sum_{n=1}^{N} |\mathbf{x}_n^T\mathbf{v}|$$

(also known as L1-PCA; see Kwak, TPAMI, 2008)

This is more robust than the standard PCA energy:

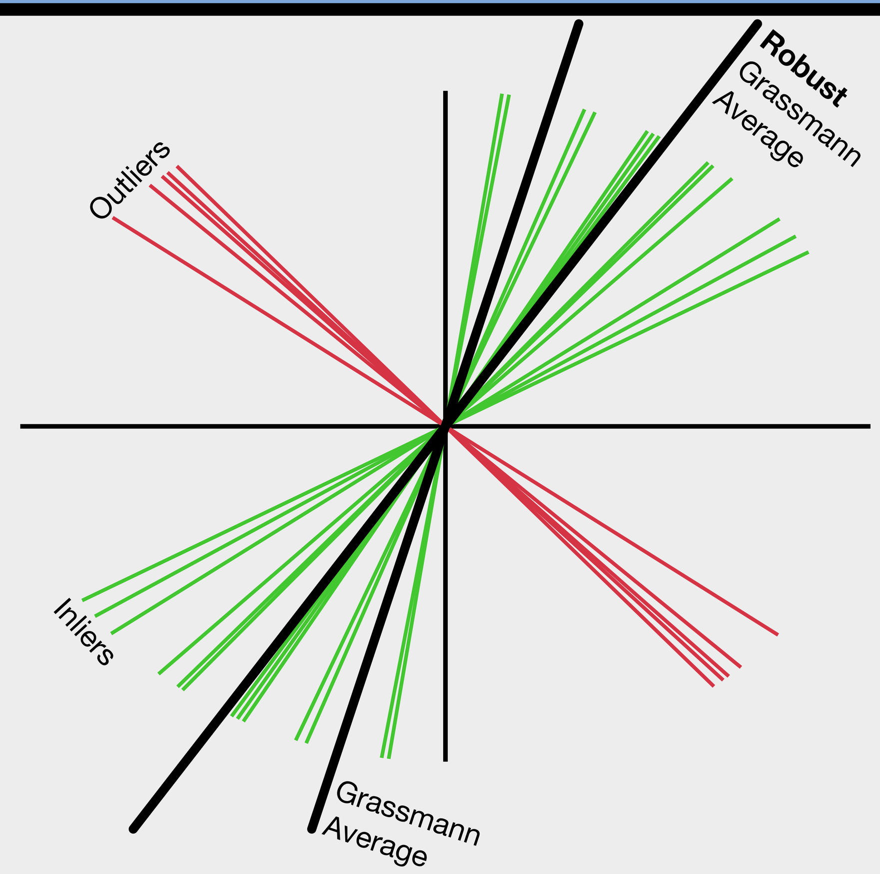
$$\mathbf{q}_{\mathrm{PCA}} = \underset{\mathbf{v}\in S^{D-1}}{\arg\max} \sum_{n=1}^{N} (\mathbf{x}_n^T\mathbf{v})^2$$

For Gaussian data, GA coincides with PCA:

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

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

5) Idea #2: Averages are easy to make robust



One of the most well-understood robust statistics is the robust average.
As we are merely computing a subspace average, we can easily phrase a **robust Grassmann average:**

Algorithm: Robust Grassmann Average (RGA)

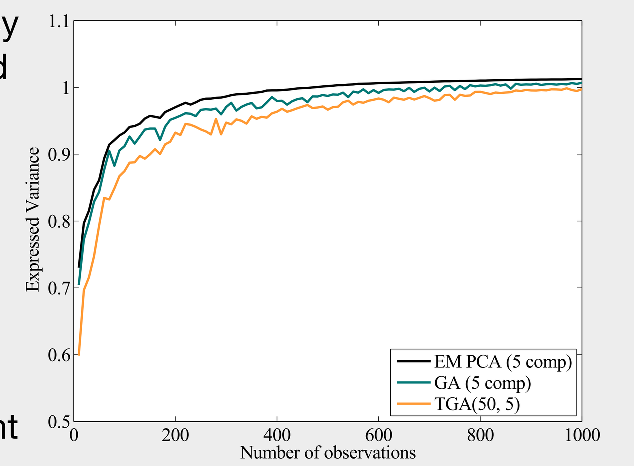
$$w_n \leftarrow \mathrm{sign}(\mathbf{u}_n^T\mathbf{q}_{i-1})\|\mathbf{x}_n\|, \quad \mathbf{q}_i \leftarrow \frac{\boldsymbol{\mu}_{\mathrm{rob}}(w_{1:N}, \mathbf{u}_{1:N})}{\|\boldsymbol{\mu}_{\mathrm{rob}}(w_{1:N}, \mathbf{u}_{1:N})\|},$$

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

In computer vision we are mostly concerned with pixel-level outliers, so we suggest a simple pixel-level trimmed average. This can be computed with the same complexity as a standard average.

6) Results

Statistical efficiency (Gaussian data of 30 dimensions)



The right figure investigate the statistical efficiency of the proposed estimators. We note the standard robustness-efficiency tradeoff.

Below we consider a film restoration task where noisy frames are denoised by projecting to a robust (RGA) subspace.

The estimated noise is transfered to new films to provide quantitative results.

We further model backgrounds with changing light using a robust subspace.

Finally, we show scalability by computing the leading 20 robust components of the entire Star Wars IV movie (a task beyond other methods).

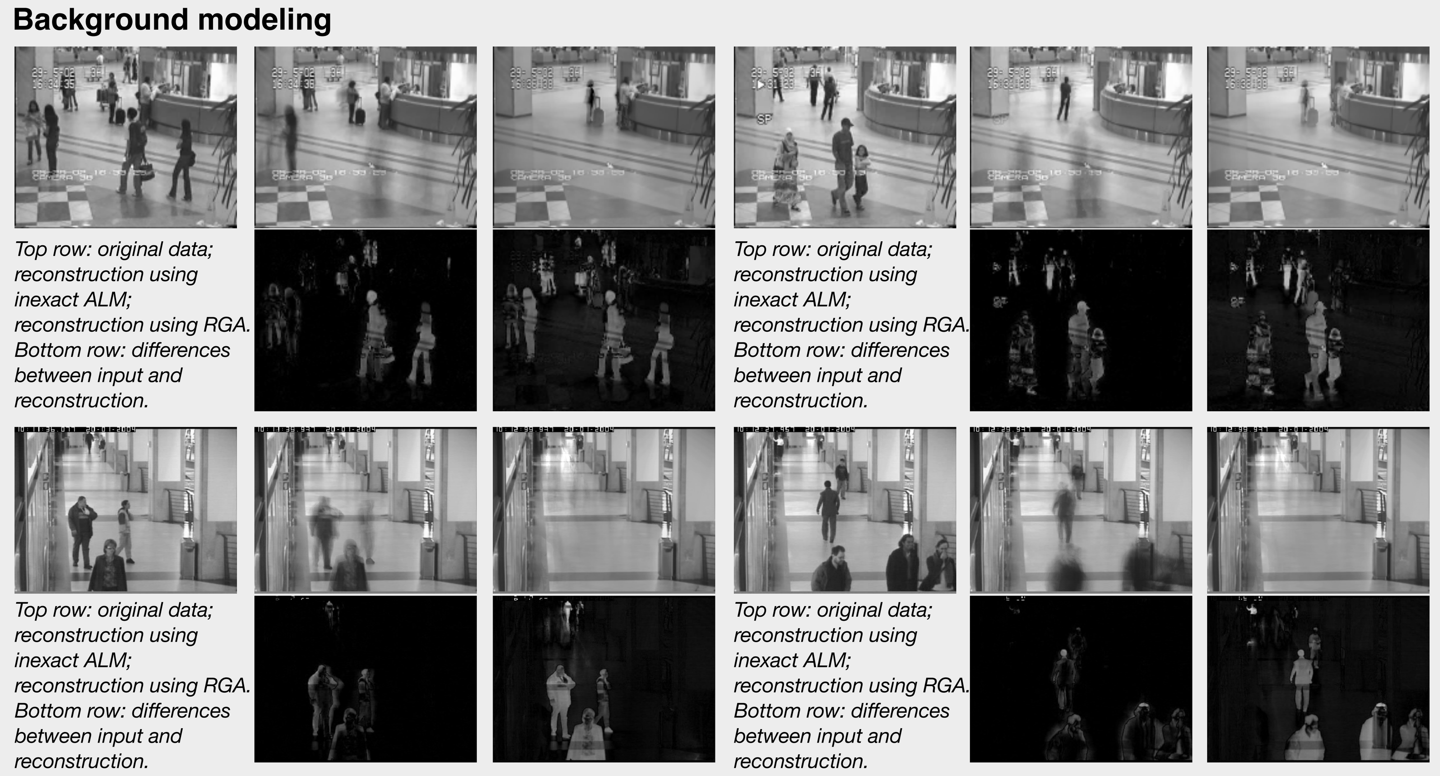
Film restoration (Nosferatu, 1922)

Top row: original data; reconstruction using inexact ALM; reconstruction using approach from De la Torre & Black; reconstruction using RGA. Bottom row: differences between input and reconstruction.

Pixel-level outliers (Noise from Nosferatu)

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

Background modeling



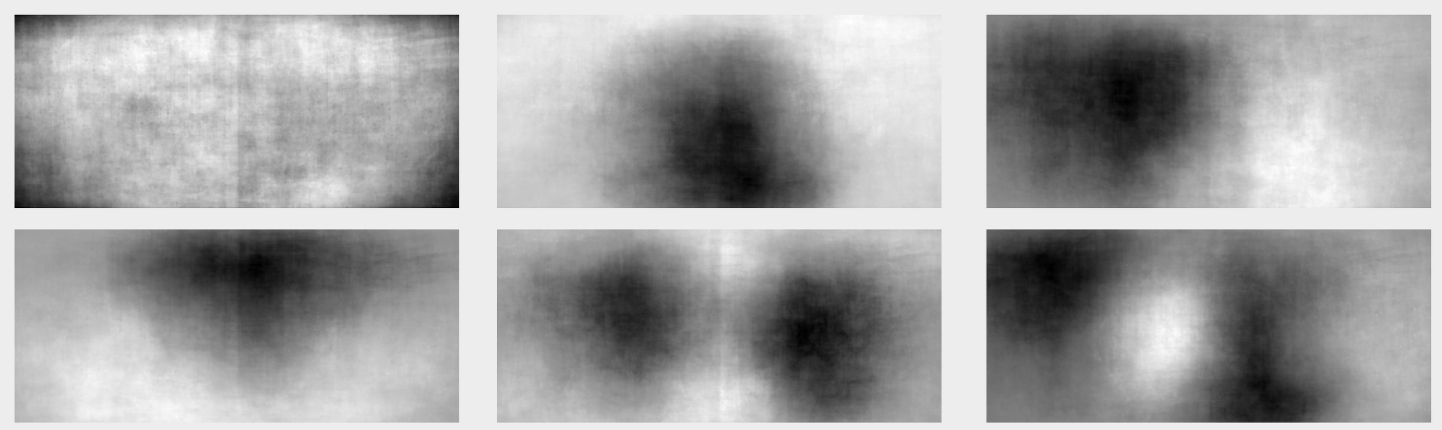
Top row: original data; reconstruction using inexact ALM; reconstruction using RGA. Bottom row: differences between input and reconstruction.

Top row: original data; reconstruction using inexact ALM; reconstruction using RGA. Bottom row: differences between input and reconstruction.

Top row: original data; reconstruction using inexact ALM; reconstruction using RGA. Bottom row: differences between input and reconstruction.

Top row: original data; reconstruction using inexact ALM; reconstruction using RGA. Bottom row: differences between input and reconstruction.

Star Wars IV



More Information

At out web page we have source code (Matlab and C++) along with further results, the paper and its supplementary material:
`http://ps.is.tue.mpg.de/project/Robust_PCA`