Riemannian Optimization and the Computation of the Divergences and the Karcher Mean of Symmetric Positive Definite Matrices

Xinru Yuan[1], **Wen Huang**[2], Kyle A. Gallivan[1], and P.-A. Absil[3]

1,Florida State University 2,Rice University 3,Université Catholique de Louvain

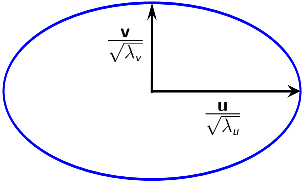
May 7, 2018
SIAM Conference on Applied Linear Algebra

Symmetric Positive Definite (SPD) Matrix

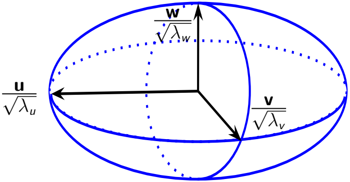
Definition

A symmetric matrix A is called positive definite iff all its eigenvalues are positive.

2 × 2 SPD matrix

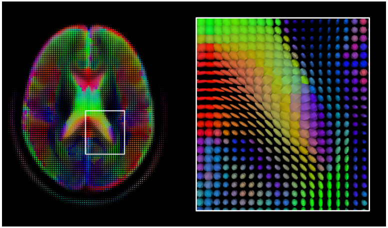


3 × 3 SPD matrix



Motivation of Averaging SPD Matrices

- Possible applications of SPD matrices
  - Diffusion tensors in medical imaging [CSV12, FJ07, RTM07]
  - Describing images and video [LWM13, SFD02, ASF^{+}05, TPM06, HWSC15]



- Motivation of averaging SPD matrices
  - Aggregate several noisy measurements of the same object
  - Subtask in interpolation methods, segmentation, and clustering

Averaging Schemes: from Scalars to Matrices

Let $A_1, \ldots, A_K$ be SPD matrices.

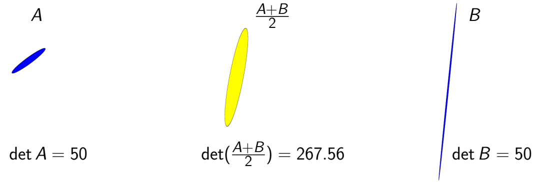
- Generalized arithmetic mean: $\frac{1}{K} \sum_{i=1}^{K} A_i$

  $\rightarrow$ Not appropriate in many practical applications

Let $A_1, \ldots, A_K$ be SPD matrices.

- Generalized arithmetic mean: $\frac{1}{K}\sum_{i=1}^{K} A_i$

  $\rightarrow$ Not appropriate in many practical applications

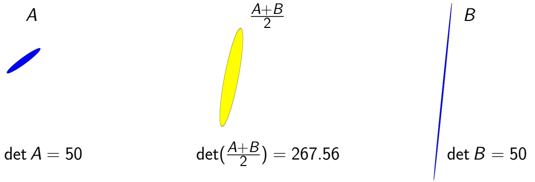


Averaging Schemes: from Scalars to Matrices

Let $A_1, \dots, A_K$ be SPD matrices.

- Generalized arithmetic mean: $\frac{1}{K} \sum_{i=1}^{K} A_i$

  $\rightarrow$ Not appropriate in many practical applications



- Generalized geometric mean: $(A_1 \cdots A_K)^{1/K}$

  $\rightarrow$ Not appropriate due to non-commutativity

  $\rightarrow$ How to define a matrix geometric mean?

Desired Properties of a Matrix Geometric Mean

The desired properties are given in the ALM list[1], some of which are:

- $G(A_{\pi(1)}, \dots, A_{\pi(K)}) = G(A_1, \dots, A_K)$ with π a permutation of $(1, \dots, K)$
- if $A_1, \dots, A_K$ commute, then $G(A_1, \dots, A_K) = (A_1, \dots, A_K)^{1/K}$
- $G(A_1, \dots, A_K)^{-1} = G(A_1^{-1}, \dots, A_K^{-1})$
- $\det(G(A_1, \dots, A_K)) = (\det(A_1) \cdots \det(A_K))^{1/K}$

[1] T. Ando, C.-K. Li, and R. Mathias, *Geometric means*, Linear Algebra and Its Applications, 385:305-334, 2004

Geometric Mean of SPD Matrices

- A well-known mean on the manifold of SPD matrices is the **Karcher mean** [Kar77]:

$$G(A_1, \ldots, A_K) = \underset{X \in \mathcal{S}_{++}^n}{\arg\min} \frac{1}{2K} \sum_{i=1}^{K} \delta^2(X, A_i), \tag{1}$$

where $\delta(X, Y) = \| \log(X^{-1/2} Y X^{-1/2}) \|_F$ is the geodesic distance under the affine-invariant metric

$$g(\eta_X, \xi_X) = \mathrm{trace}(\eta_X X^{-1} \xi_X X^{-1})$$

- The Karcher mean defined in (1) satisfies all the geometric properties in the ALM list [LL11]

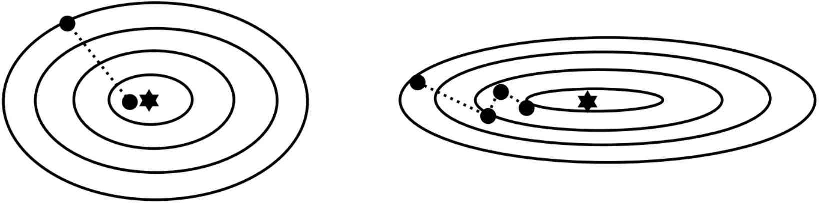
Algorithms

$$G(A_1, \dots, A_k) = \operatorname*{arg\,min}_{X \in \mathcal{S}^n_{++}} \frac{1}{2K} \sum_{i=1}^{K} \delta^2(X, A_i),$$

- Riemannian steepest descent [RA11, Ren13]
- Riemannian Barzilai-Borwein method [IP15]
- Riemannian Newton method [RA11]
- Richardson-like iteration [BI13]
- Riemannian steepest descent, conjugate gradient, BFGS, and trust region Newton methods [JVV12]
- Limited-memory Riemannian BFGS method [YHAG16]

Conditioning of the Objective Function

Hemstitching phenomenon
for steepest descent



well-conditioned Hessian   ill-conditioned Hessian

- Small condition number $\Rightarrow$ fast convergence
- Large condition number $\Rightarrow$ slow convergence

Conditioning of the Karcher Mean Objective Function

- **Riemannian metric**:

$$g_X(\xi, \eta) = \mathrm{trace}(\xi X^{-1} \eta X^{-1})$$

- **Euclidean metric**:

$$g_X(\xi, \eta) = \mathrm{trace}(\xi \eta)$$

Condition number κ of Hessian at the minimizer μ:

- Hessian of Riemannian metric:
  - $\kappa(H^R) \leq 1 + \frac{\ln(\max \kappa_i)}{2}$, where $\kappa_i = \kappa(\mu^{-1/2} A_i \mu^{-1/2})$
  - $\kappa(H^R) \leq 20$ if $\max(\kappa_i) = 10^{16}$
- Hessian of Euclidean metric:
  - $\frac{\kappa^2(\mu)}{\kappa(H^{\mathrm{R}})} \leq \kappa(H^{\mathrm{E}}) \leq \kappa(H^{\mathrm{R}}) \kappa^2(\mu)$
  - $\kappa(H^E) \geq \kappa^2(\mu)/20$

BFGS Quasi-Newton Algorithm: from Euclidean to Riemannian

- Update formula:

$$x_{k+1} = \underline{x_k + \alpha_k \eta_k}$$

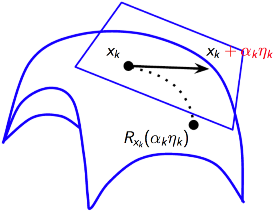
- Search direction:

$$\eta_k = -B_k^{-1} \operatorname{grad} f(x_k)$$

- B_k update:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k},$$

where $s_k = \underline{x_{k+1} - x_k}$, and $y_k = \underline{\operatorname{grad} f(x_{k+1}) - \operatorname{grad} f(x_k)}$



Optimization on a Manifold

replace by $R_{x_k}(\eta_k)$

- Update formula:

$$x_{k+1} = x_k + \alpha_k \eta_k$$

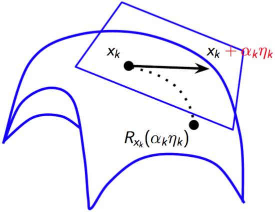
- Search direction:

$$\eta_k = -B_k^{-1} \operatorname{grad} f(x_k)$$

- B_k update:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k},$$

where $s_k = x_{k+1} - x_k$, and $y_k = \operatorname{grad} f(x_{k+1}) - \operatorname{grad} f(x_k)$



Optimization on a Manifold

replace by $R_{x_k}(\eta_k)$

- Update formula:

$$x_{k+1} = x_k \underline{+ \alpha_k \eta_k}$$

- Search direction:

$$\eta_k = -B_k^{-1} \operatorname{grad} f(x_k)$$

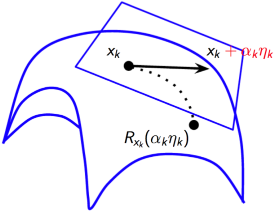
- B_k update:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k},$$

where $s_k = \underline{x_{k+1} - x_k}$, and $y_k = \underline{\operatorname{grad} f(x_{k+1}) - \operatorname{grad} f(x_k)}$

replaced by $R_{x_k}^{-1}(x_{k+1})$

on different tangent spaces



Optimization on a Manifold

Riemannian BFGS (RBFGS) Algorithm

- Update formula:

$$x_{k+1} = R_{x_k}(\alpha_k \eta_k) \text{ with } \eta_k = -\mathcal{B}_k^{-1} \operatorname{grad} f(x_k)$$

- $\mathcal{B}_k$ update [HGA15]:

$$\mathcal{B}_{k+1} = \tilde{\mathcal{B}}_k - \frac{\tilde{\mathcal{B}}_k s_k (\tilde{\mathcal{B}}_k s_k)^\flat}{(\tilde{\mathcal{B}}_k s_k)^\flat s_k} + \frac{y_k y_k^\flat}{y_k^\flat s_k},$$

where $s_k = \mathcal{T}_{\alpha_k \eta_k} \alpha_k \eta_k$, $y_k = \beta_k^{-1} \operatorname{grad} f(x_{k+1}) - \mathcal{T}_{\alpha_k \eta_k} \operatorname{grad} f(x_k)$, and $\tilde{\mathcal{B}}_k = \mathcal{T}_{\alpha_k \eta_k} \circ \mathcal{B}_k \circ \mathcal{T}_{\alpha_k \eta_k}^{-1}$.

- Stores and transports $\mathcal{B}_k^{-1}$ as a dense matrix

- Requires excessive computation time and storage space for large-scale problem

Limited-memory RBFGS (LRBFGS)

Riemannian BFGS:

- $\mathcal{B}_{k+1} = \tilde{\mathcal{B}}_k - \frac{\tilde{\mathcal{B}}_k s_k (\tilde{\mathcal{B}}_k s_k)^\flat}{(\tilde{\mathcal{B}}_k s_k)^\flat s_k} + \frac{y_k y_k^\flat}{y_k^\flat s_k}$,
  where $s_k = \mathcal{T}_{\alpha_k \eta_k} \alpha_k \eta_k$, $y_k = \beta_k^{-1} \operatorname{grad} f(x_{k+1}) - \mathcal{T}_{\alpha_k \eta_k} \operatorname{grad} f(x_k)$,
  and $\tilde{\mathcal{B}}_k = \mathcal{T}_{\alpha_k \eta_k} \circ \mathcal{B}_k \circ \mathcal{T}_{\alpha_k \eta_k}^{-1}$

Limited-memory Riemannian BFGS:

- Stores only the m most recent s_k and y_k
- Transports those vectors to the new tangent space rather than the entire matrix $\mathcal{B}_k^{-1}$
- Computational and storage complexity depends upon m

Implementations

- Representations of tangent vectors
- Retraction
- Vector transport

Implementations

- Representations of tangent vectors: $T_X \mathcal{S}^{\mathrm{n}}_{++} = \{S \in \mathbb{R}^{n \times n} | S = S^T\}$
  - Extrinsic representation: n^2-dimensional vector
  - Intrinsic representation: d-dimensional vector where $d = n(n+1)/2$

- Retraction

- Vector transport

Implementations

- Representations of tangent vectors: $T_X\,\mathcal{S}^{\mathrm{n}}_{++} = \{S \in \mathbb{R}^{n\times n} | S = S^T\}$
  - Extrinsic representation: n^2-dimensional vector
  - Intrinsic representation: d-dimensional vector where $d = n(n+1)/2$
- Retraction
  - Exponential mapping: $\mathrm{Exp}_X(\xi) = X^{1/2}\exp(X^{-1/2}\xi X^{-1/2})X^{1/2}$
- Vector transport
  - Parallel translation: $\mathcal{T}_{p_\eta}(\xi) = Q\xi Q^T$, with $Q = X^{\frac{1}{2}}\exp(\dfrac{X^{-\frac{1}{2}}\eta X^{-\frac{1}{2}}}{2})X^{-\frac{1}{2}}$

Implementations

- Representations of tangent vectors: $T_X \mathcal{S}^{n}_{++} = \{S \in \mathbb{R}^{n\times n} | S = S^T\}$
  - Extrinsic representation: n^2-dimensional vector
  - Intrinsic representation: d-dimensional vector where $d = n(n+1)/2$
- Retraction
  - Exponential mapping: $\mathrm{Exp}_X(\xi) = X^{1/2} \exp(X^{-1/2} \xi X^{-1/2}) X^{1/2}$
  - Second order approximation retraction [JVV12]:
    $$R_X(\xi) = X + \xi + \frac{1}{2}\xi X^{-1} \xi$$
- Vector transport
  - Parallel translation: $\mathcal{T}_{p_\eta}(\xi) = Q \xi Q^T$, with $Q = X^{\frac{1}{2}} \exp(\frac{X^{-\frac{1}{2}} \eta X^{-\frac{1}{2}}}{2}) X^{-\frac{1}{2}}$
  - Vector transport by parallelization [HAG15]: essentially an identity

Complexity Comparison for LRBFGS

Extrinsic approach:

- Function
- Riemannian gradient

Intrinsic approach:

- Function
- Riemannian gradient

Both approaches have the same complexities: $f + \nabla f$ cost

Complexity Comparison for LRBFGS

Extrinsic approach:

- Function
- Riemannian gradient
- Retraction
  - Evaluate $R_X(\eta)$

Intrinsic approach:

- Function
- Riemannian gradient
- Retraction
  - Compute η from $\tilde{\eta}^d$
  - Evaluate $R_X(\eta)$

$$\text{Intrinsic cost} = \text{Extrinsic cost} + 2n^3 + o(n^3)$$

Complexity Comparison for LRBFGS

Extrinsic approach:

- Function
- Riemannian gradient
- Retraction
- Riemannian metric
  - $6n^3 + o(n^3)$

Intrinsic approach:

- Function
- Riemannian gradient
- Retraction
- Reduces to Euclidean metric
  - $n^2 + o(n^2)$

Complexity Comparison for LRBFGS

Extrinsic approach:

- Function
- Riemannian gradient
- Retraction
- Riemannian metric
- $(2m)$ times of vector transport

Intrinsic approach:

- Function
- Riemannian gradient
- Retraction
- Reduces to Euclidean metric
- No explicit vector transport

Complexity Comparison for LRBFGS

Extrinsic approach:

- Function
- Riemannian gradient
- Retraction
- Riemannian metric
- $(2m)$ times of vector transport

Intrinsic approach:

- Function
- Riemannian gradient
- Retraction
- Reduces to Euclidean metric
- No explicit vector transport

Complexity comparison:

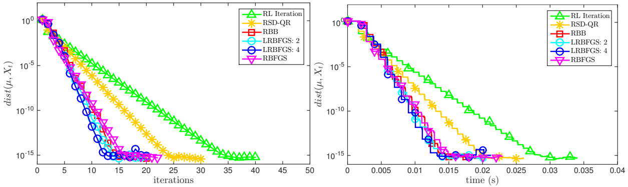
- $f + \nabla f +$
  $27n^3 + 12mn^2 +$
  $\underline{2m \times \text{Vector transport cost}}$

- $f + \nabla f +$
  $22n^3/3 + 4mn^2$

Numerical Results: Comparison of Different Algorithms

$K = 100$, size $= 3 \times 3$, $d = 6$

- $1 \leq \kappa(A_i) \leq 200$



- $10^3 \leq \kappa(A_i) \leq 10^7$

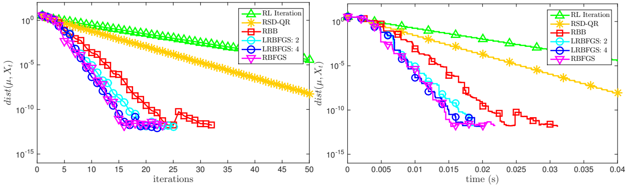
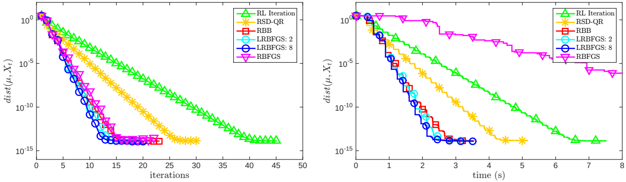


Figure: Evolution of averaged distance between current iterate and the exact Karcher mean with respect to time and iterations

Numerical Results: Comparison of Different Algorithms

$K = 30$, size $= 100 \times 100$, $d = 5050$

- $1 \leq \kappa(A_i) \leq 20$



- $10^4 \leq \kappa(A_i) \leq 10^7$

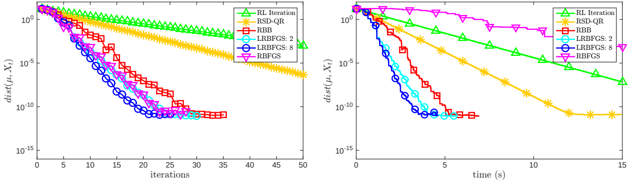


Figure: Evolution of averaged distance between current iterate and the exact Karcher mean with respect to time and iterations

Numerical Results: Riemannian vs. Euclidean Metrics

- $K = 100$, $n = 3$, and $1 \leq \kappa(A_i) \leq 10^6$.

- $K = 30$, $n = 100$, and $1 \leq \kappa(A_i) \leq 10^5$.

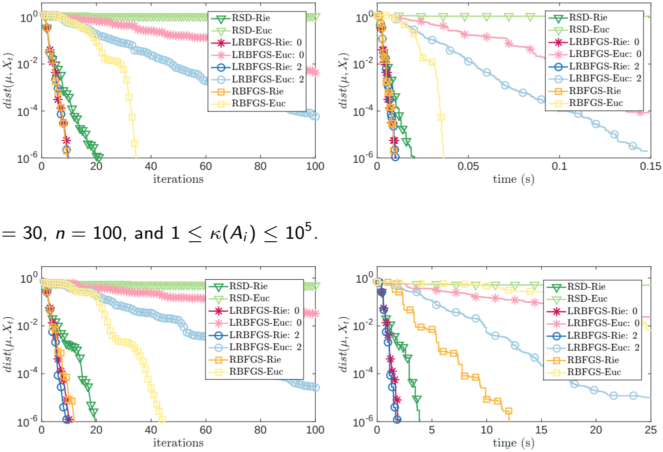


Figure: Evolution of averaged distance between current iterate and the exact Karcher mean with respect to time and iterations

Outline

- Karcher mean computation on $\mathcal{S}_{++}^{\mathsf{n}}$
- Divergence-based means on $\mathcal{S}_{++}^{\mathsf{n}}$
- L^1-norm median computation on $\mathcal{S}_{++}^{\mathsf{n}}$
- Application: Structure tensor image denoising
- Summary

Motivations

- Karcher mean

$$K(A_1, \ldots, A_K) = \underset{X \in \mathcal{S}^n_{++}}{\arg\min} \frac{1}{2K} \sum_{i=1}^{K} \delta^2(X, A_i), \tag{1}$$

  where $\delta(X, Y) = \| \log(X^{-1/2} Y X^{-1/2}) \|_F$

  - pros: holds desired properties
  - cons: high computational cost

- Use divergences as alternatives to the geodesic distance due to their computational and empirical benefits

- A divergence is like a distance except it lacks
  - triangle inequality
  - symmetry

LogDet α-divergence and Associated Mean

The LogDet α-divergence is defined as

$$G(A_1, \dots, A_k) = \underset{X \in \mathcal{S}^n_{++}}{\arg\min} \frac{1}{2K} \sum_{i=1}^{K} \delta^2_{\mathrm{LD},\alpha}(A_i, X) \,, \qquad (2)$$

where the LogDet α-divergence on $\mathcal{S}^{\mathrm{n}}_{++}$ is given by

$$\delta^2_{\mathrm{LD},\alpha}(X, Y) = \frac{4}{1-\alpha^2} \log \frac{\det(\frac{1-\alpha}{2} X + \frac{1+\alpha}{2} Y)}{\det(X)^{\frac{1-\alpha}{2}} \det(Y)^{\frac{1+\alpha}{2}}}$$

- The LogDet α-divergence is asymetric in general, except for $\alpha = 0$
- (2) defines the right mean. The left mean can be defined in a similar way.

Karcher Mean vs. LogDet α-divergence Mean

- Complexity comparison for problem-related operations

	function	gradient	total
LD α-div. mean	$\frac{2Kn^3}{3}$	$3Kn^3$	$\frac{11Kn^3}{3}$
Karcher mean	$18Kn^3$	$5Kn^3$	$23Kn^3$

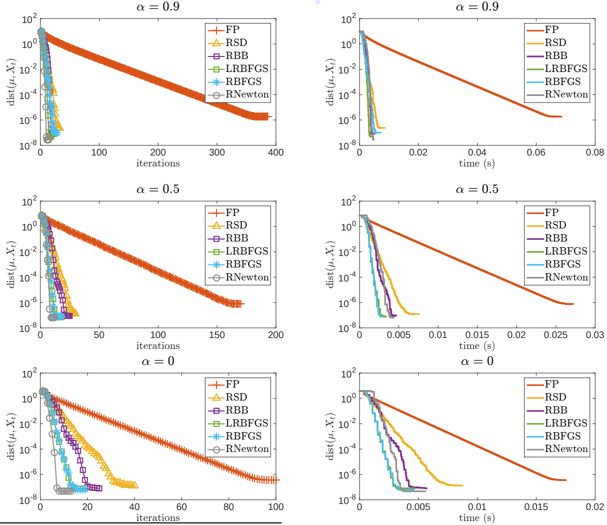
- Invariance properties

	scaling invariance	rotation invariance	congruence invariance	inversion invariance
LD α-div. mean	✓	✓	✓	✗
Karcher mean	✓	✓	✓	✓

Numerical Experiment: Comparions of Different Algorithms

$K = 100$, $n = 3$, and $10 \leq \kappa(A_i) \leq 10^6$

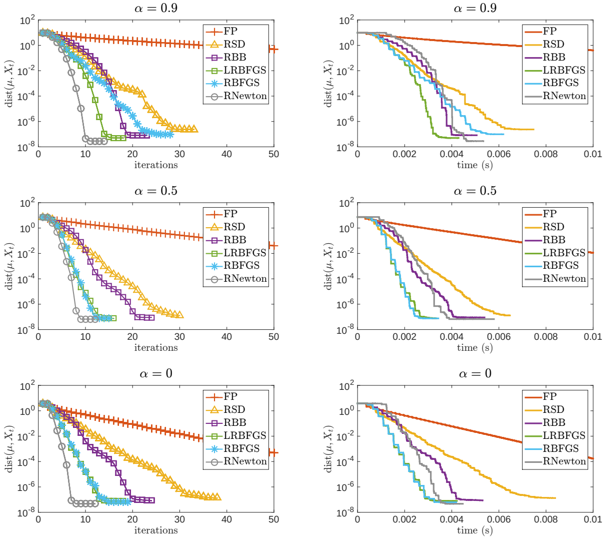
Fixed Point iteration[2]: stepsize $= \dfrac{1-\alpha}{2K}$



[2]Z. Chebbi and M. Moakher. Means of Hermitian positive-definite matrices based on the log-determinant -divergence function. Linear Algebra and its Applications, 436(7):1872C1889, 2012

Numerical Experiment: Comparions of Different Algorithms

- $K = 100$, $n = 3$, and $10 \leq \kappa(A_i) \leq 10^6$



Outline

- Karcher mean computation on $\mathcal{S}^{n}_{++}$
- Divergence-based means on $\mathcal{S}^{n}_{++}$
- L^1-norm median computation on $\mathcal{S}^{n}_{++}$
- Application: Structure tensor image denoising
- Conclusion

- The mean of a set of points is sensitive to outliers
- The median is robust to outliers

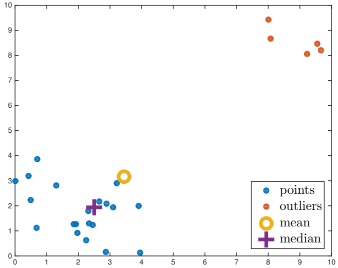


Figure: The geometric mean and median in $\mathbb{R}^2$ space.

Riemannian Median of SPD Matrices

The Riemannian median of a set of SPD matrices is defined as:

$$M(A_1, \ldots, A_K) = \arg\min_{X \in \mathcal{S}_{++}^n} \frac{1}{2K} \sum_{i=1}^{K} \delta(A_i, X),$$

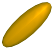
where $\delta(X, Y)$ is a distance or the square root of a divergence function.

- The cost function is non-smooth at $X = A_i$
- There may exist multiple local minima for some δ

Numerical Experiment

Original data ($K = 50$)

Outliers

outliers		0	1	5	10	25	50
δ_R	Median						
	Mean						

Outline

- Karcher mean computation on $\mathcal{S}^{\mathrm{n}}_{++}$
- Divergence-based means on $\mathcal{S}^{\mathrm{n}}_{++}$
- Riemannian L^1-norm median computation on $\mathcal{S}^{\mathrm{n}}_{++}$
- Application: Structure tensor image denoising
- Conclusion

Application: Structure Tensor Image Denoising

- A structure tensor image is a spatial structured matrix field

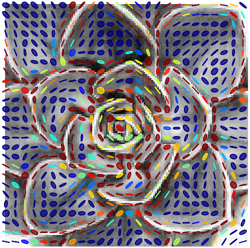
$$\mathcal{I} : \Omega \subset \mathbb{Z}^2 \to \mathcal{S}^{\mathrm{n}}_{++}$$

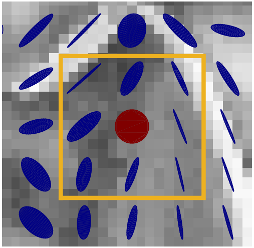
- Noisy tensor images are simulated by replacing the pixel values by an outlier tensor with a given probability *Pr*

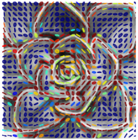
- Denoising is done by averaging matrices in the neighborhood of each pixel

- Mean Riemannian Error:

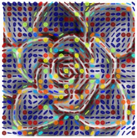
$$MRE = \frac{1}{\#\Omega} \sum_{(i,j)\in\Omega} \delta_R(\mathcal{I}_{i,j}, \tilde{\mathcal{I}}_{i,j})$$



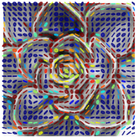




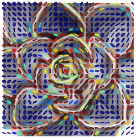
(a) Original image



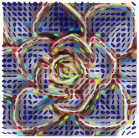
(b) A-mean



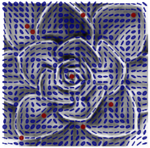
(c) K-mean



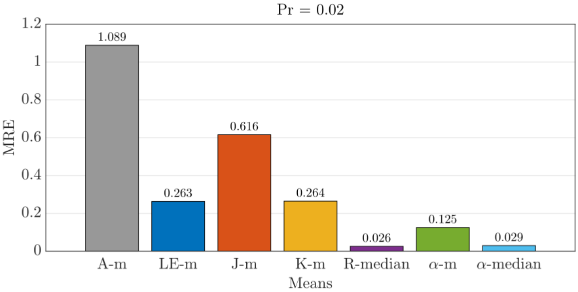
(d) R-median



(e) α-mean

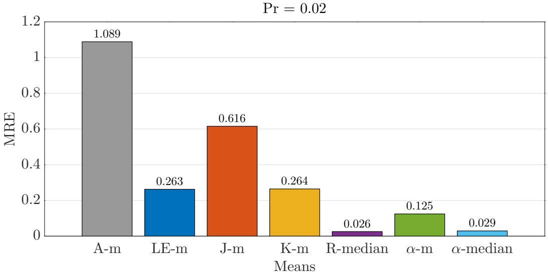


(f) Noisy image $Pr = 0.02$

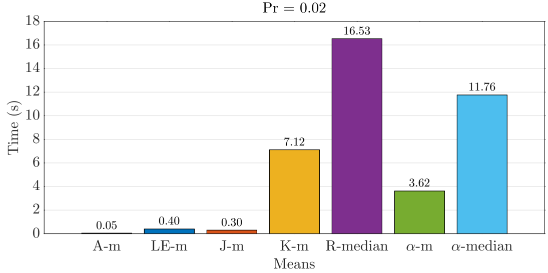


Structure Tensor Image Denoising: MRE and Time

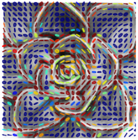
- MRE comparison



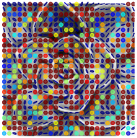
- Time comparison



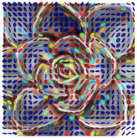
Structure Tensor Image Denoising: $Pr = 0.1$



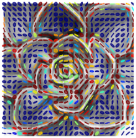
(g) Original image



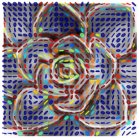
(h) A-mean



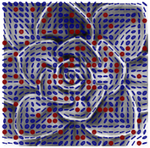
(i) K-mean



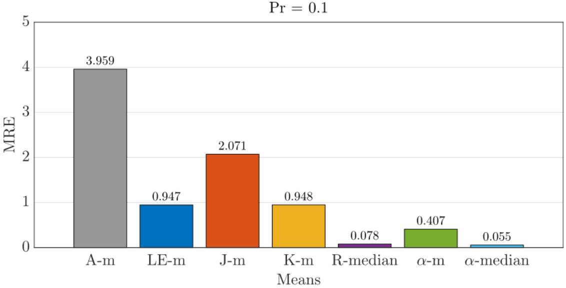
(j) R-median



(k) α-mean

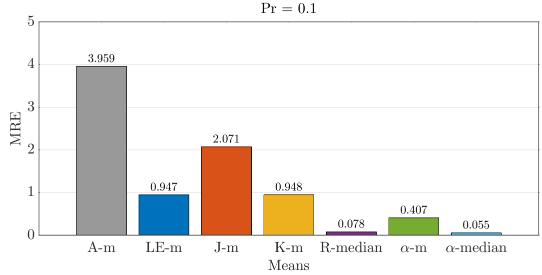


(l) Noisy image $Pr = 0.1$

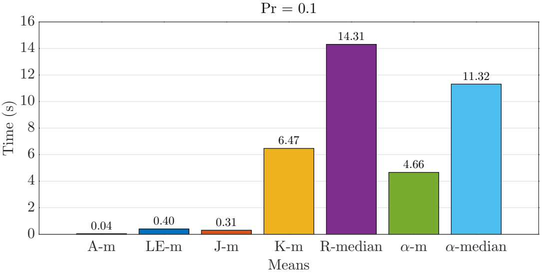


Structure Tensor Image Denoising: MRE and Time

- MRE comparison



- Time comparison



Summary

- Karcher mean for SPD matrices
  - Analyze the conditioning of the Hessian of the Karcher mean cost function
  - Apply a limited-memory Riemannian BFGS method to computing the SPD Karcher mean with efficient implementations
  - Recommend using LRBFGS as the default method for the SPD Karcher mean computation
- Other averaging techniques for SPD matrices
  - Investigate divergence-based means and Riemannian L^1-norm medians on $\mathcal{S}_{++}^{\mathrm{n}}$
  - Use recent development in Riemannian optimization to develop efficient and robust algorithm on $\mathcal{S}_{++}^{n}$
- Evaluate the performance of different averaging techniques in applications

Thank you!

References I

Ognjen Arandjelovic, Gregory Shakhnarovich, John Fisher, Roberto Cipolla, and Trevor Darrell.
Face recognition with image sets using manifold density divergence.
In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 581–588. IEEE, 2005.

D. A. Bini and B. Iannazzo.
Computing the Karcher mean of symmetric positive definite matrices.
Linear Algebra and its Applications, 438(4):1700–1710, 2013.

Guang Cheng, Hesamoddin Salehian, and Baba Vemuri.
Efficient recursive algorithms for computing the mean diffusion tensor and applications to DTI segmentation.
Computer Vision–ECCV 2012, pages 390–401, 2012.

References II

P. T. Fletcher and S. Joshi.
Riemannian geometry for the statistical analysis of diffusion tensor data.
Signal Processing, 87(2):250–262, 2007.

Wen Huang, P-A Absil, and Kyle A Gallivan.
A Riemannian symmetric rank-one trust-region method.
Mathematical Programming, 150(2):179–216, 2015.

Wen Huang, Kyle A Gallivan, and P-A Absil.
A Broyden class of quasi-Newton methods for Riemannian optimization.
SIAM Journal on Optimization, 25(3):1660–1685, 2015.

Zhiwu Huang, Ruiping Wang, Shiguang Shan, and Xilin Chen.
Face recognition on large-scale video in the wild with hybrid Euclidean-and-Riemannian metric learning.
Pattern Recognition, 48(10):3113–3124, 2015.

References III

Bruno Iannazzo and Margherita Porcelli.
The Riemannian Barzilai-Borwein method with nonmonotone line-search and the Karcher mean computation.
Optimization online, December, 2015.

B. Jeuris, R. Vandebril, and B. Vandereycken.
A survey and comparison of contemporary algorithms for computing the matrix geometric mean.
Electronic Transactions on Numerical Analysis, 39:379–402, 2012.

H. Karcher.
Riemannian center of mass and mollifier smoothing.
Communications on Pure and Applied Mathematics, 1977.

J. Lawson and Y. Lim.
Monotonic properties of the least squares mean.
Mathematische Annalen, 351(2):267–279, 2011.

References IV

Jiwen Lu, Gang Wang, and Pierre Moulin.
Image set classification using holistic multiple order statistics features and localized multi-kernel metric learning.
In *Proceedings of the IEEE International Conference on Computer Vision*, pages 329–336, 2013.

Q. Rentmeesters and P.-A. Absil.
Algorithm comparison for Karcher mean computation of rotation matrices and diffusion tensors.
In *19th European Signal Processing Conference*, pages 2229–2233, Aug 2011.

Q. Rentmeesters.
Algorithms for data fitting on some common homogeneous spaces.
PhD thesis, Universite catholiqué de Louvain, 2013.

References V

Y. Rathi, A. Tannenbaum, and O. Michailovich.
Segmenting images on the tensor manifold.
In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007.

Gregory Shakhnarovich, John W Fisher, and Trevor Darrell.
Face recognition from long-term observations.
In *European Conference on Computer Vision*, pages 851–865. Springer, 2002.

Oncel Tuzel, Fatih Porikli, and Peter Meer.
Region covariance: A fast descriptor for detection and classification.
In *European conference on computer vision*, pages 589–600. Springer, 2006.

Xinru Yuan, Wen Huang, P.-A. Absil, and Kyle A. Gallivan.
A Riemannian limited-memory BFGS algorithm for computing the matrix geometric mean.
Procedia Computer Science, 80:2147–2157, 2016.