

# Supplementary Material : “Diffusion tensor smoothing through weighted Karcher means”

Owen Carmichael<sup>†</sup>, Jun Chen<sup>§</sup>, Debashis Paul<sup>§</sup> and Jie Peng<sup>§</sup>

<sup>†</sup> Departments of Neurology and Computer Science, University of California, Davis

<sup>§</sup>Department of Statistics, University of California, Davis

July 4, 2013

## S-1 Recursive algorithm for weighted Karcher mean under affine-invariant geometry

We propose a simple recursive method for approximating the weighted Karcher mean under affine-invariant geometry. This algorithm does not involve any gradient or Hessian computations and thus is computationally fast. It is based on the following observation: for a sequence of points  $z_1, \dots, z_n$  in the Euclidean space  $\mathbb{R}^p$ , one can compute their weighted mean  $m := (\sum_{i=1}^n w_i z_i) / (\sum_{i=1}^n w_i)$  for  $w_1, \dots, w_n > 0$  through an  $n$ -step recursive procedure:

(i) Set  $m_1 = z_1$ , and  $j = 1$ ;

(ii) Compute  $m_{j+1} = m_j + \frac{w_{j+1}}{\sum_{i=1}^{j+1} w_i} (z_{j+1} - m_j)$ ;

(iii) If  $j + 1 < n$ , set  $j = j + 1$  and go to (ii). Otherwise, set  $m = m_n$  and stop.

In step (ii),  $m_{j+1}$  is on the line segment connecting  $m_j$  and  $z_{j+1}$ . Note that straight lines are geodesics in the Euclidean space. Thus, as a generalization to a Riemannian manifold  $\mathcal{M}$ , this step can be modified as follows:

(ii)' Compute  $m_{j+1} = \gamma(1 - \frac{w_{j+1}}{\sum_{i=1}^{j+1} w_i})$ , where  $\gamma(\cdot)$  is the geodesic on  $(\mathcal{M}, g)$  such that  $\gamma(0) = m_j$  and  $\gamma(1) = z_{j+1}$ .

Note that, in general,  $m$  depends on the ordering of the points  $z_1, \dots, z_n$  unless  $z_i$ 's all lie on a geodesic. When approximating the solution to equation (12) (Section 3), we first pre-order the tensors  $X_i$ 's based on the Euclidean distances of  $s_i$ 's from  $s$ , i.e.,  $z_1 := X_1$ , where  $s_1$  is closest to  $s$  among all  $s_i$ 's, etc. For a manifold, the geodesic  $\gamma(\cdot)$  between two (nearby) points  $X, Y$  such that  $\gamma(0) = X$  and  $\gamma(1) = Y$  can be expressed through the exponential map and the logarithm map:  $\gamma(t) = \exp_X(t \log_X(Y))$ . For the tensor space with affine-invariant metric, we have: for  $X, Y \in \mathcal{P}_N$  and  $S \in T_X(\mathcal{P}_N)$

$$\exp_X(S) = X^{1/2} \exp(X^{-1/2} S X^{-1/2}) X^{1/2}, \quad \log_X(Y) = X^{1/2} \log(X^{-1/2} Y X^{-1/2}) X^{1/2}. \quad (\text{S-1})$$

Compared with the gradient-based methods, the above algorithm has the advantages of being numerically stable and computationally efficient. Important properties of this recursive algorithm

include: (a) It will yield the weighted affine-invariant mean if all the tensors lie along a geodesic, or when all the tensors have the same orientation. (b) Under the setting of Section 3, the quantity resulting from this algorithm differs from the weighted affine-invariant mean only in the third order term. The approximation is especially good in the context tensor smoothing since the distribution of the weights is highly skewed as voxels nearer to the target voxel have larger weights. This has been experimentally verified by comparing the results using the recursive algorithm to those using a conjugate gradient procedure.

## S-2 Simulation studies

### S-2.1 Description of the simulated tensor field for Simulation I

Band orientation	Slice #	Band #	Indices for band boundaries	Tensors on band
Horizontal	1 and 2	1	20, 35	diag(0.25, 16, 0.25)
		2	60, 75	diag(0.5, 4, 0.5)
		3	90, 105	diag(0.7, 2, 0.7)
	3 and 4	1	40, 50	diag(0.25, 16, 0.25)
		2	80, 90	diag(0.5, 4, 0.5)
		3	110, 120	diag(0.7, 2, 0.7)
Vertical	1 and 2	1	20, 35	diag(16, 0.25, 0.25)
		2	60, 75	diag(4, 0.5, 0.5)
		3	90, 105	diag(2, 0.7, 0.7)
	3 and 4	1	40, 50	diag(0.25, 16, 0.25)
		2	80, 90	diag(0.5, 4, 0.5)
		3	110, 120	diag(0.7, 2, 0.7)

Table S-1: Description of the tensor field (Simulation I)

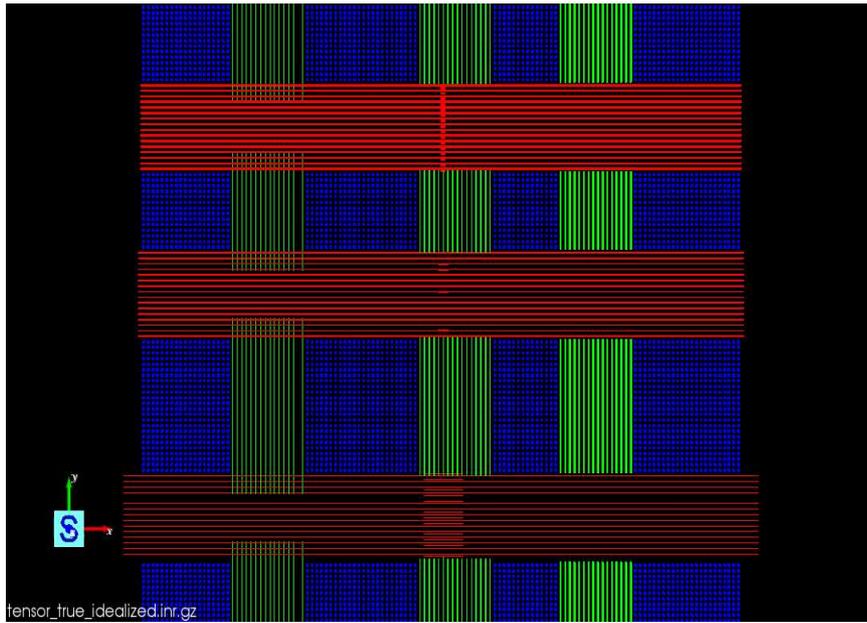


Figure S-1: Schematic plot of the tensor field (one slice) for Simulation I

## S-2.2 Linear vs. nonlinear regression estimates under Simulation I

Table S-2.2 below reports the median and median absolute deviation (MAD) of the estimation errors (in terms of the affine-invariant distance to the true tensor) for linear and nonlinear regression procedures across the bands (anisotropic tensors), the background (isotropic tensors) and the whole tensor field under Simulation I. The main findings are the following.

- (i) Nonlinear regression of the raw DWI data gives more accurate estimate of the tensors compared to the linear regression regardless of noise level and region. The improvement is only slight on the background. However, the improvement is significant on the bands.
- (ii) Comparison under the two simulation schemes (9 gradients each used once and twice, respectively) shows that the results for both linear and nonlinear regression improve when the number of gradient directions increases.
- (iii) Results across different noise levels show that the performance of both linear and nonlinear regression degrades when the noise level increases.

Table S-2: Comparison between linear regression and nonlinear regression on DWI data. Reported numbers are the medians of the error norms (in parenthesis, MAD of error norms) in three regions: whole set, bands and background.

<b>Nine gradient directions each used once</b>				
Noise level	Method	Whole set	Bands	Background
$\sigma = 10$	linear	0.1049 (0.0460)	0.3207 (0.2409)	0.0757 (0.0185)
	nonlinear	0.0991 (0.0380)	0.1823 (0.1177)	0.0757 (0.0185)
$\sigma = 50$	linear	0.5462 (0.2523)	1.6672 ( 1.3299)	0.3850 (0.0982)
	nonlinear	0.5141 (0.2148)	1.0617 (0.7383)	0.3829 (0.0970)
$\sigma = 100$	linear	1.2382 (0.6738)	10.9 ( 5.844)	0.819 (0.2592)
	nonlinear	1.1318 (0.5615)	2.8713 (2.3365)	0.8009 (0.2424)
<b>Nine gradient directions each repeated twice</b>				
Noise level	Method	Whole set	Bands	Background
$\sigma = 10$	linear	0.073891 (0.0322)	0.229592 (0.1734)	0.053692 (0.0130)
	nonlinear	0.069904 (0.0267)	0.129959 (0.0844)	0.053679 (0.0130)
$\sigma = 50$	linear	0.383068 (0.1770)	1.330171 (1.0666)	0.271789 (0.0681)
	nonlinear	0.359311 (0.1481)	0.828572 (0.5926)	0.269491 (0.0672)
$\sigma = 100$	linear	0.825685 (0.4091)	2.489265 (2.0479)	0.566317 (0.1579)
	nonlinear	0.758624 (0.3443)	1.726173 (1.2410)	0.548341 (0.1484)