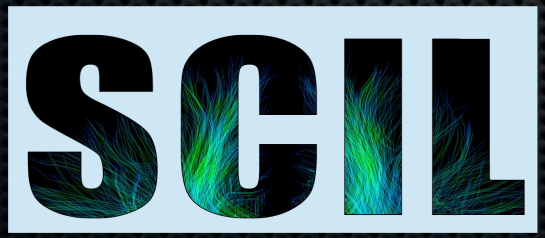
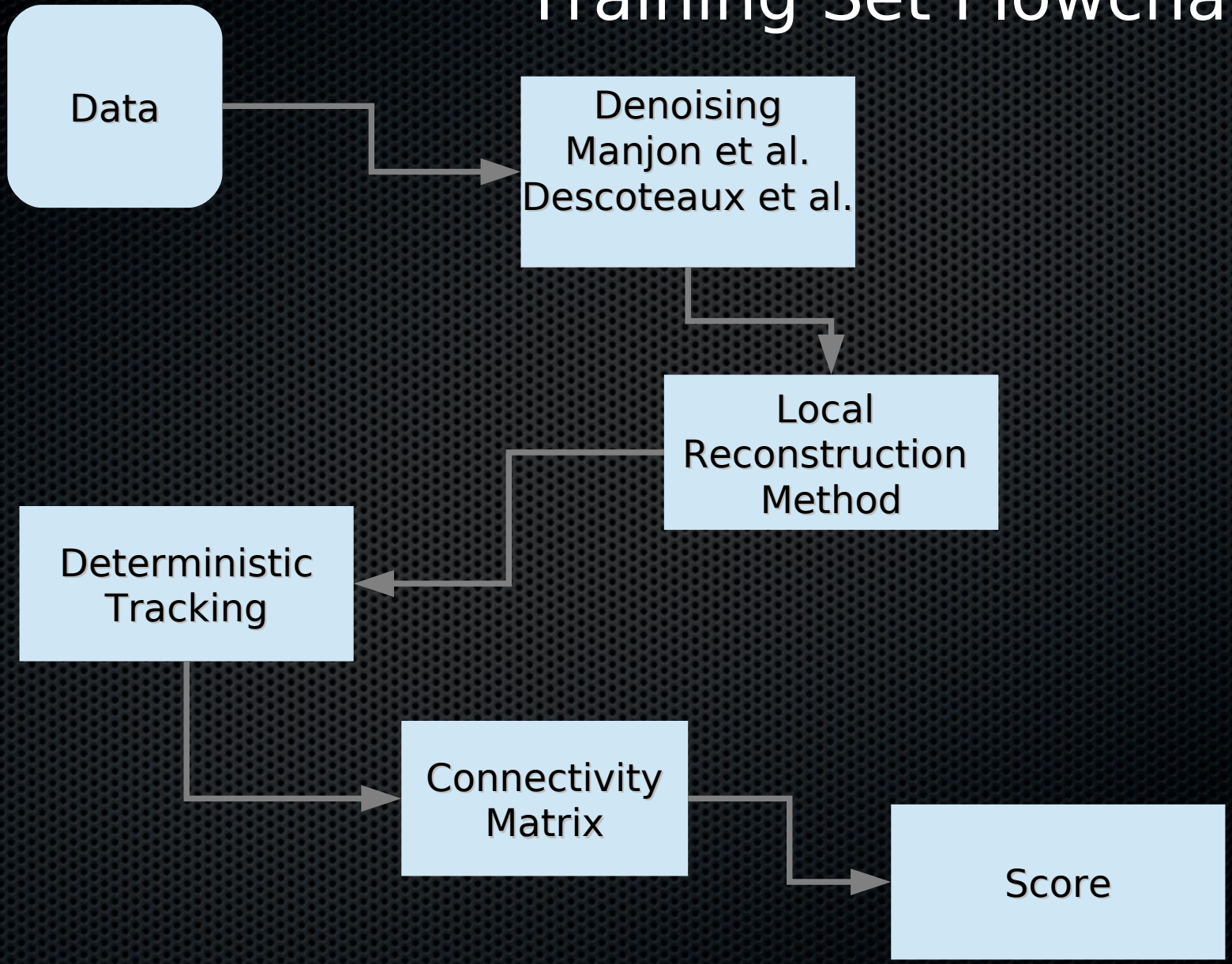
Evaluating 5 reconstruction methods in all three categories

Eleftherios Garyfallidis and Michael Paquette
Université de Sherbrooke,Quebec, Canada
scil.dinf.usherbrooke.ca



Training Set Flowchart



Local Reconstruction Methods

DTI/HARDI

CSD: Constrained Spherical Deconvolution

Tournier et al.

SDT: Spherical Deconvolution Transform

Descoteaux et al.

PSO: Multi-tensor with particle swarming.

Paquette et al.

Heavyweight

DSID: Diffusion Spectrum Imaging with Deconvolution

Canales-Rodriguez et al.

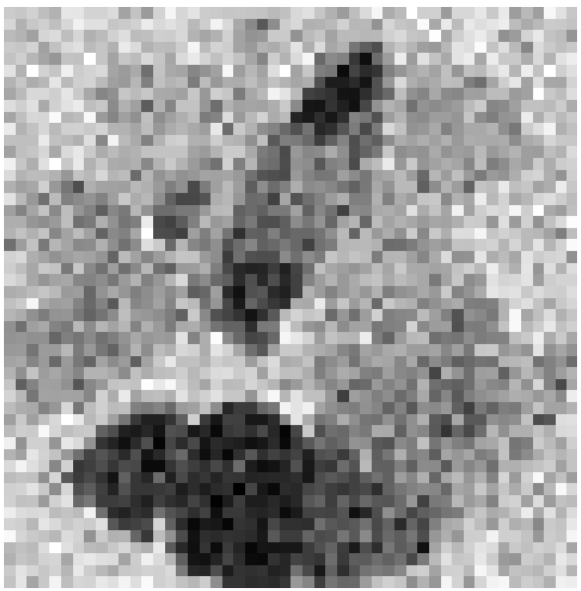
GQID: Generalized Q-sampling Imaging with Spherical Deconvolution

Garyfallidis PhD thesis

Denoising

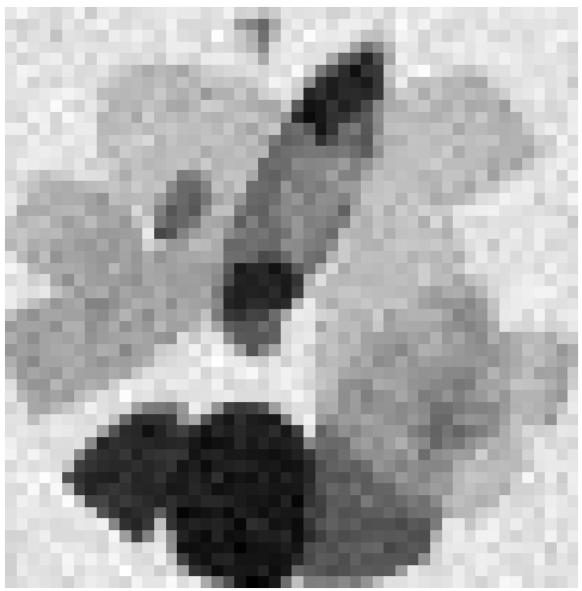
Adaptive non-local means denoising method

SNR = 10



SNR = 10 Denoised

SNR = 30



[Manjón, Coupé et al., 2010]

Multi-Tensor Model

The Multi-Tensor (MT) generative model:

$$S(b, g) = \sum_{i=0}^{N-1} f_i e^{-b g^t D_i g}$$

where
b, g are the gradient b-value and unit orientation,
f_i is the volume fraction of the i^{th} compartement,
D_i is the symmetric rank-2 tensor of the i^{th} compartement,
N is the number of compartement,

$S(b, g)$ is the signal generated by the MT model with parameters (f, D, N) at (b, g).

MT Fitting

- Constrained MT model:

$$S^C_{b,g} = f_0 e^{-bD_0} + \sum_{i=1}^{3} f_i e^{-bg^t D_i g}$$

constraints: 4 compartement (1 isotropic, 3 prolate).

MT Fitting

- Constrained MT model:

$$S^C_{b,g} = f_0 e^{-bD_0} + \sum_{i=1}^{3} f_i e^{-bg^t D_i g}$$

  constraints: 4 compartement (1 isotropic, 3 prolate).

- Fit by minimizing $\sum_k (S^c_{b_k,g_k} - Y_{b_k,g_k})^2$

  where Y is the measured diffusion signal.

Particle Swarm Optimization (PSO)

PSO is a stochastic global optimization method,

[http://www.itm.uni-stuttgart.de/research/pso_opt/bilder/pso.gif]

Particle Swarm Optimization (PSO)

PSO is a stochastic global optimization method,

- Np random position in the parameters space (called particles),

[http://www.itm.uni-stuttgart.de/research/pso_opt/bilder/pso.gif]

Particle Swarm Optimization (PSO)

PSO is a stochastic global optimization method,

- Np random position in the parameters space (called particles),
- Updated at every timestep by a velocity,

[http://www.itm.uni-stuttgart.de/research/pso_opt/bilder/pso.gif]

Particle Swarm Optimization (PSO)

PSO is a stochastic global optimization method,

- Np random position in the parameters space (called particles),
- Updated at every timestep by a velocity,
- Computed with information from the whole swarm of particles.

[http://www.itm.uni-stuttgart.de/research/pso_opt/bilder/pso.gif]

PSO

- Swarm: $\Omega_j^{t+1} = \Omega_j^t + v_j^{t+1}$

[Kennedy and Eberhart, 1995]

PSO

- Swarm: $\Omega_j^{t+1} = \Omega_j^t + v_j^{t+1}$
- Velocity: $v_j^{t+1} = wv_j^t + \phi_p r_p(p_j^t - \Omega_j^t) + \phi_g r_g(g^t - \Omega_j^t)$

[Kennedy and Eberhart, 1995]

PSO

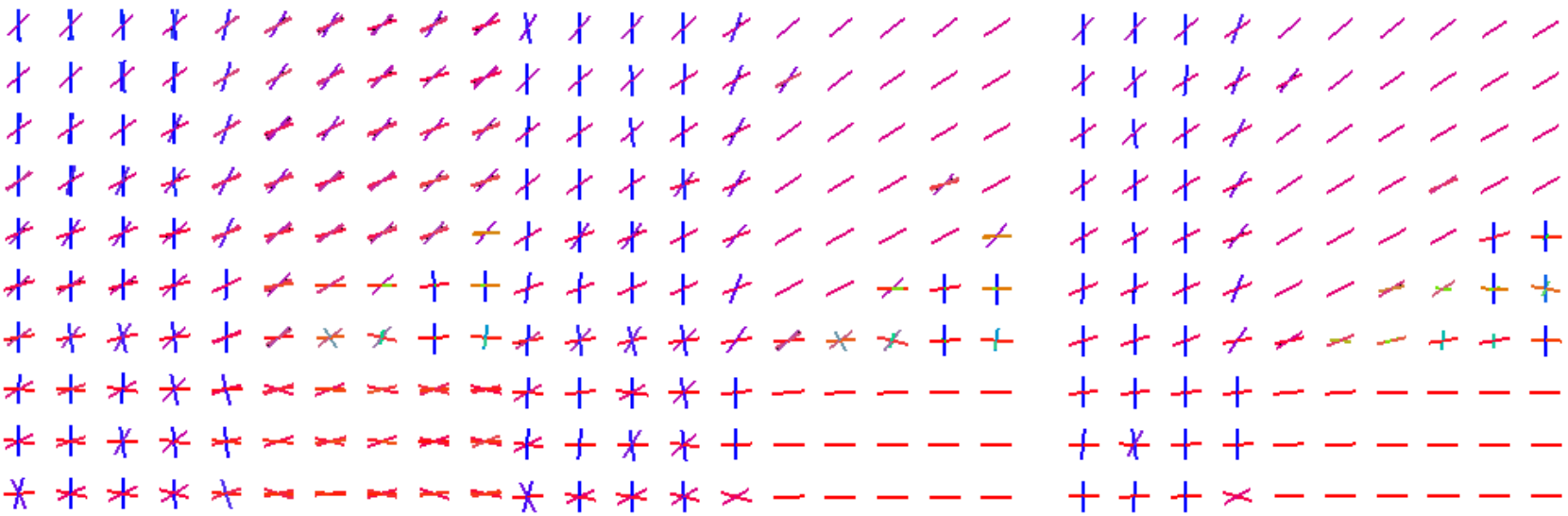
- Swarm: $\Omega_j^{t+1} = \Omega_j^t + v_j^{t+1}$
- Velocity: $v_j^{t+1} = wv_j^t + \phi_p r_p(p_j^t - \Omega_j^t) + \phi_g r_g(g^t - \Omega_j^t)$

- where
  w, ϕ_p and ϕ_g are user tuned parameters,
  p_j^t is the j^{th} particle's best known position at iteration t,
  g^t is the swarm's best known position at iteration t,
  r_p, r_g are uniform random variable.

[Kennedy and Eberhart, 1995]

Model Complexity

3 Tensor Fit | Angular Threshold | Neighborhood Threshold



DSI real ODF

$$\psi_{DSI}(\hat{\mathbf{u}}) = \int_0^{\infty} P(r\hat{\mathbf{u}}) r^2 dr$$

Signal to Propagator relationship

$$S(\mathbf{q}) = S_0 \int P(\mathbf{r}) \exp(i2\pi\mathbf{q}\cdot\mathbf{r}) d\mathbf{r}$$

$$\downarrow \quad Q(\mathbf{r}) = S_0 P(\mathbf{r})$$

$$S(\mathbf{q}) = \int Q(\mathbf{r}) \exp(i2\pi\mathbf{q}\cdot\mathbf{r}) d\mathbf{r}$$

$$\downarrow \quad S(\mathbf{q}) = S(-\mathbf{q})$$

$$Q(\mathbf{r}) = \int S(\mathbf{q}) \exp(-2\pi\mathbf{q}\cdot\mathbf{r}) d\mathbf{q}$$

$$\downarrow \quad \text{cosine transform}$$

$$Q(\mathbf{r}) = \int S(\mathbf{q}) \cos(2\pi\mathbf{q}\cdot\mathbf{r}) \mathbf{q}$$

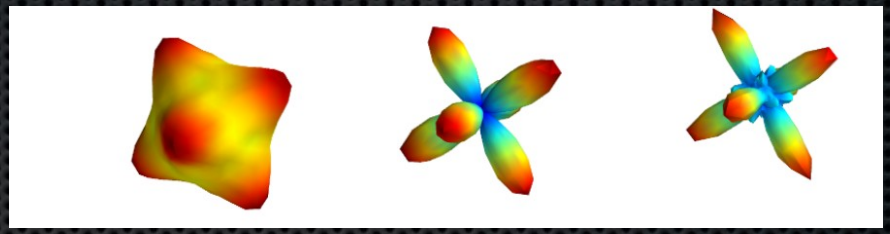
GQI2 real ODF

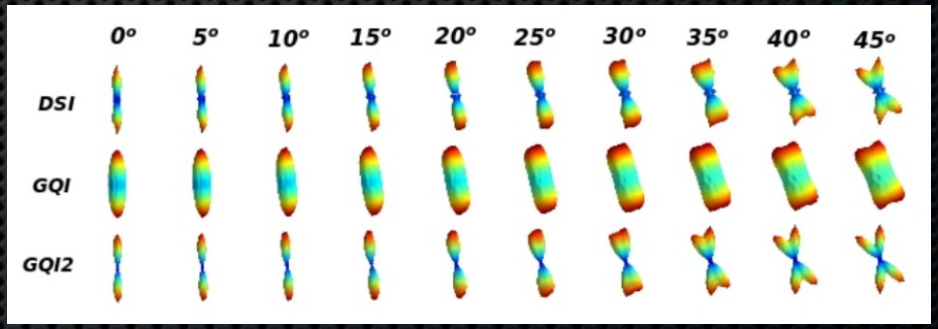
$$\psi_{GQI2}(\hat{\mathbf{u}}) = \int_0^{\lambda} Q(r\hat{\mathbf{u}}) r^2 dr$$

$$\psi_{GQI2}(\hat{\mathbf{u}}) = \lambda^3 \int S(\mathbf{q}) H(2\pi\lambda\mathbf{q}\cdot\hat{\mathbf{u}}) d\mathbf{q}$$

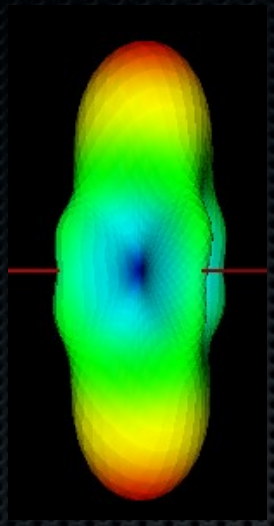
$$H(x) = \begin{cases} \frac{2\cos(x)}{x^2} + \frac{(x^2-2)\sin(x)}{x^3}, & x \neq 0 \\ 1/3, & x = 0 \end{cases}$$

GQI GQI2 DSI





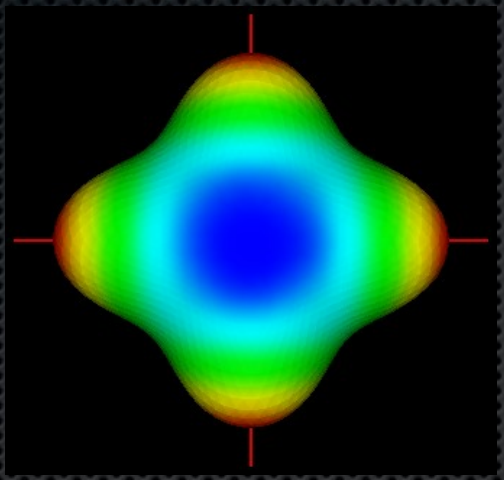
Spherical Deconvolution Transform



HARDI Signal

c_j

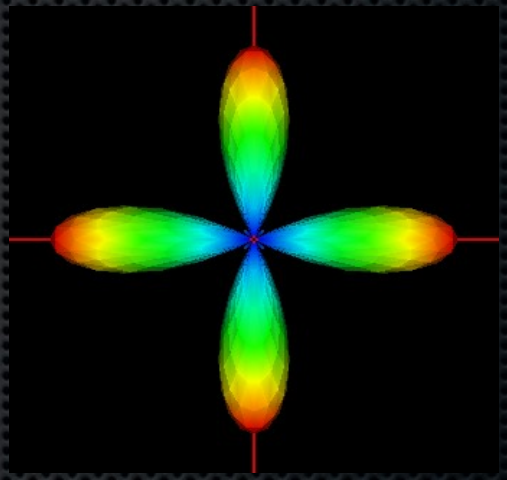
FRT



Diffusion ODF

$2\pi P_{l(j)}(0)$

Deconvolution sharpening transform

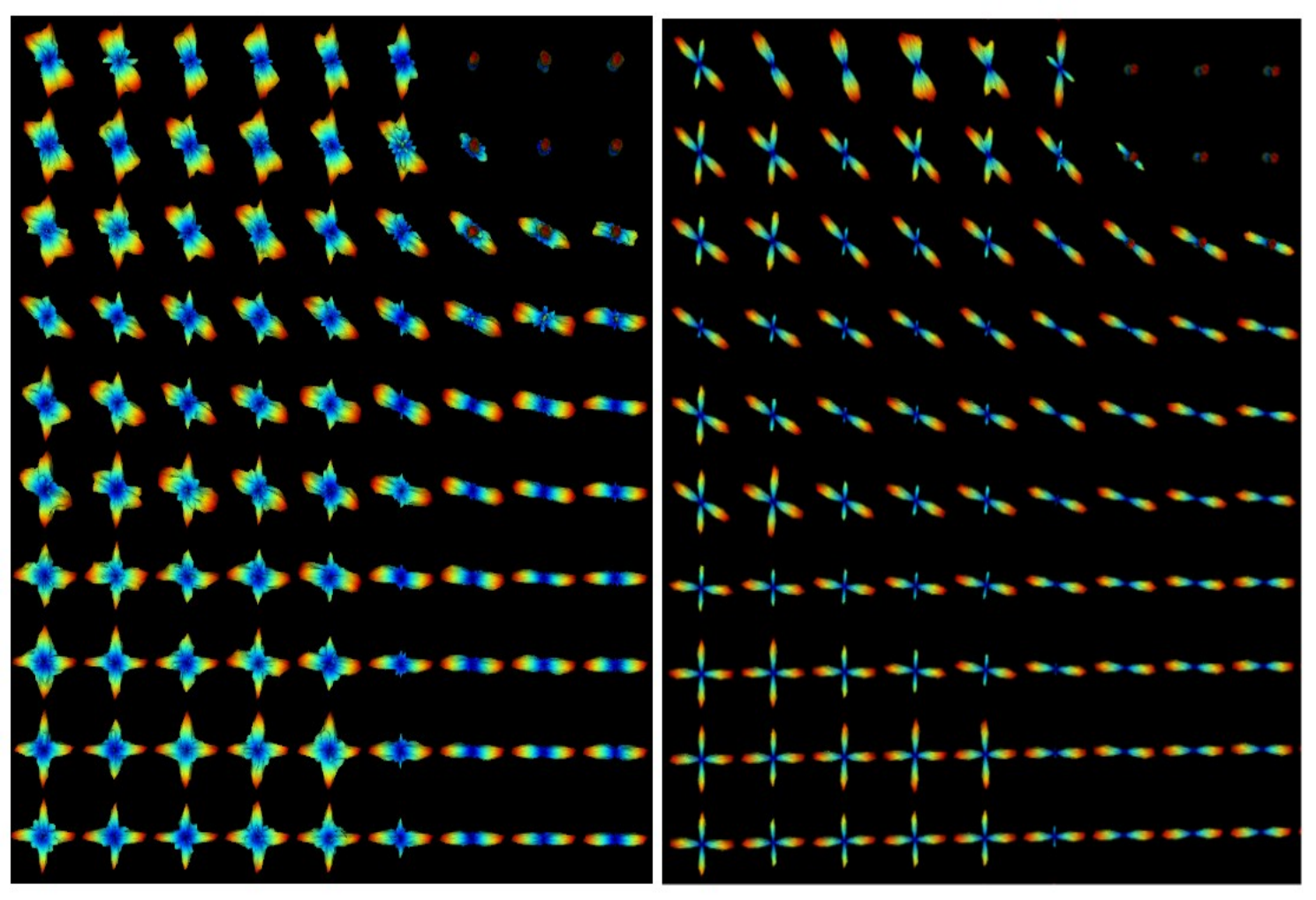


Fiber ODF

$1/ f_j$

GQI2 vs GQID

(GQI2 with deconvolution)




DIPY

dipy.org

Dipy — dipy 0.6.0.dev documentation - Google Chrome

dipy.org

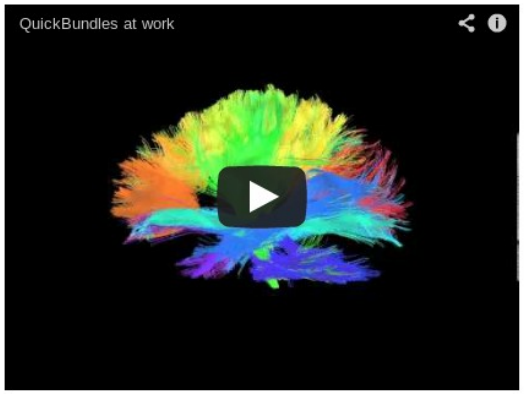
Mendeley | nipy/dipy | Garyfallidis/dipy | fos/fos | Numpy Doc | Builders | reST | reST2 | pep8

Diffusion Imaging In Python

Dipy is a **free** and **open source** software project for **diffusion** *magnetic resonance imaging* (dMRI) **analysis**.

Highlights

In Dipy we care about methods which can solve complex problems efficiently and robustly. QuickBundles is one of the many state-of-the art algorithms found in Dipy. It can be used to simplify large datasets of streamlines. See our examples and try QuickBundles with your data *examples_index*. Here is a video of QuickBundles applied on a simple dataset.



Announcements

- **Release!** Version 0.6.0 21/02/2013
- **Dipy 3rd Sprint**, Berkeley, CA, April 2013
- **ISBI 2013 will be based on Dipy**, February 2013

Getting Started

Here is a simple example showing how to calculate *color_FA*. We use a

Site Navigation

Documentation
Development

NIPY Community

Community Home
NIPY Projects
Mailing List
License

Table Of Contents

Diffusion Imaging In Python
- Highlights
- Announcements
- Getting Started
- To infinity and beyond

Next topic

Documentation

This Page

Show Source

Search mailing list archive

Go

Search this site

Go

fnins-06-00175 (2).pdf | eleftherios_pict....jpg | Show all downloads...

Dipy Developers (Core team)

Matthew
Berkeley

Eleftherios
Cambridge, UK

Ian
MRC CBU Cambridge, UK

Ariel
Stanford

Bago
UCSF

Stefan
South Africa

Dipy Developers (Full list)

- Eleftherios Garyfallidis, University of Sherbrooke, QC, CA
- Ian Nimmo-Smith, MRC Cognition and Brain Sciences Unit, Cambridge, UK
- Matthew Brett, University of California, Berkeley, CA, US
- Bago Amirbekian, University of California, San Fransisco, CA, US
- Ariel Rokem, Stanford University, CA, US
- Stefan Van der Walt, Stellenbosch University, ZA
- Maxime Descoteaux, University of Sherbrooke, QC, CA
- Michael Paquette, University of Sherbrooke, QC, CA
- Christopher Nguyen, University of California, Los Angeles, CA, US
- Emanuele Olivetti, NeuroInformatics Laboratory (NILab), Trento, IT
- Yaroslav Halchenco, PBS Department, Dartmouth, NH, US
- Samuel St-Jean, University of Sherbrooke, QC, CA
- Your name here

We combine expertise in computer science, mathematics, psychology and medicine.

Mission

The purpose of dipy is to make it easier to do better diffusion MR imaging research.

- clearly written
- clearly explained
- a good fit for the underlying ideas
- a natural home for collaboration

We hope that, if we fail to do this, you will let us know and we will try and make it better.

Truly Open

BSD License

Documentation

Unit tests

Examples

Tutorials

Buildbots

Git/Github

Travis

Dependencies

Numpy

Scipy

Cython

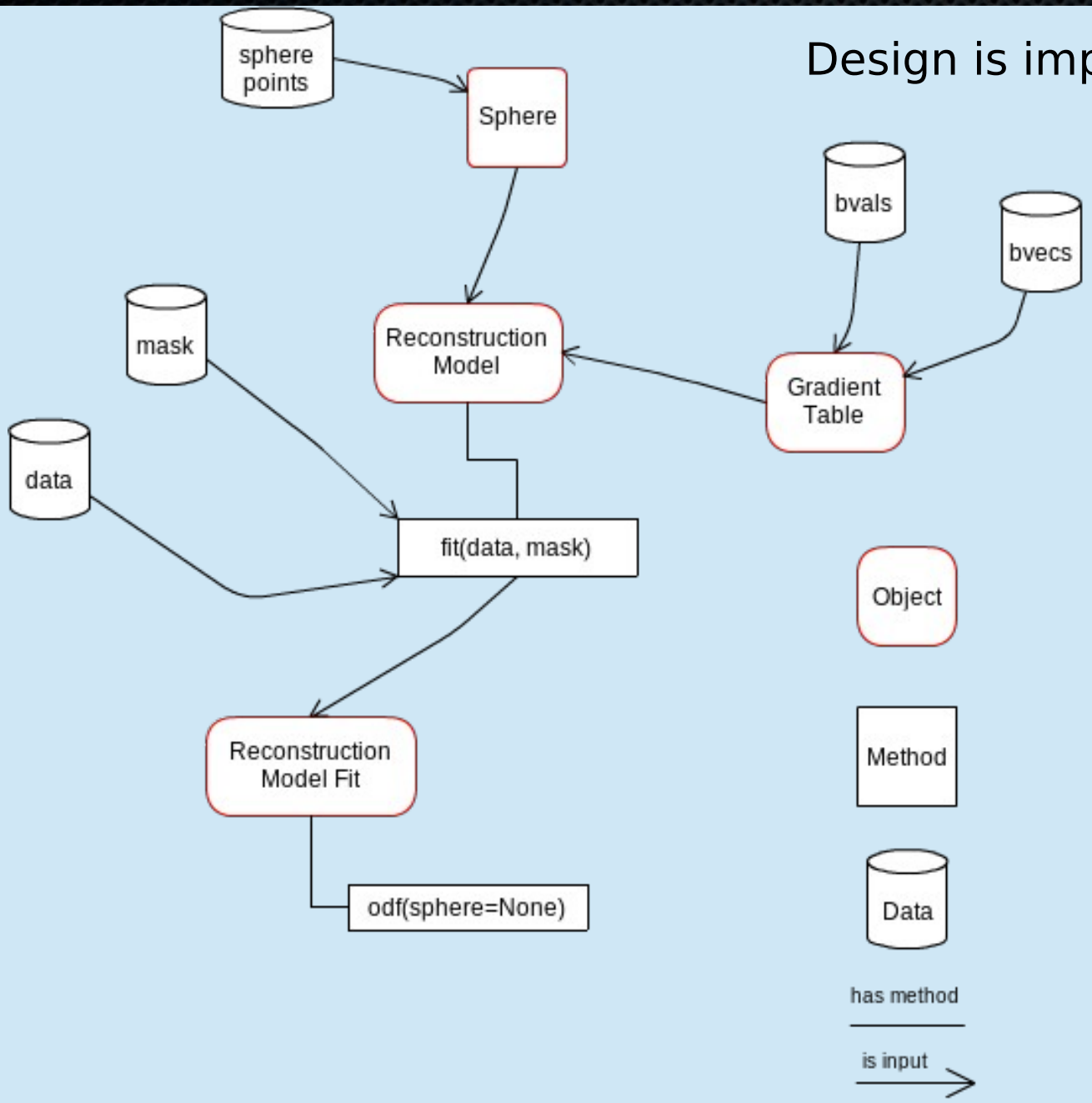
Nibabel

(Optional)

Python vtk

Pytables

Design is important



List of algorithms

Reconstruction

- Diffusion Spectrum Imaging (Wedeen)
- Generalized Q-Sampling Imaging (Yeh)
- Diffusion Spectrum Imaging with Deconvolution (Canales-Rodriguez)
- Qball (Descoteaux)
- Constant Solid Angle (Aganj)
- Opdt (Tristan-Vega)
- Diffusion Tensor (Basser)
- Simulations
- ++ CSD + MultiTensor on their way.

Tracking

- Deterministic
- Probabilistic
- Statistics (length etc.)
- Streamline counts

Input / Output

- nifti
- dicom (partly supported)
- trackvis
- hdf5
- fast pickling
- numpy io

List of algorithms

Segmentation
- QuickBundles

Volume
- Fast volume traversal (ndindex)
- Reslice volumes

Registration
- Warp streamlines

And your module here

Get involved and share some code!

Dipy 0.6 is out.

Thank you!