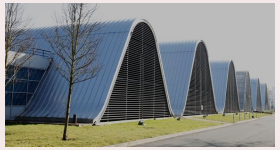


MEDUSA: an HPC-based simulation environment to create decoders of white matter microstructure

Cyril Poupon[12] , Alexis Brullé[12] , Anas Bachiri[1], **Ivy Uszynski**[1]

[1]*BAOBAB, NeuroSpin, Paris-Saclay University, CNRS, CEA*
[2] *AIDAS CEA-FZJ Joint Institute*

2022, September, 19th

AI Data Analytics and Scalable Simulations

Outline

Introduction: why building realistic digital white matter phantoms ?

Existing digital white matter phantom generators

The MEDUSA approach: combining realism and computational efficiency

Large scale simulations with the MEDUSA simulator

Conclusion

Introduction: why building realistic digital white matter phantoms ?

Existing digital white matter phantom generators

The MEDUSA approach: combining realism and computational efficiency

Large scale simulations with the MEDUSA simulator

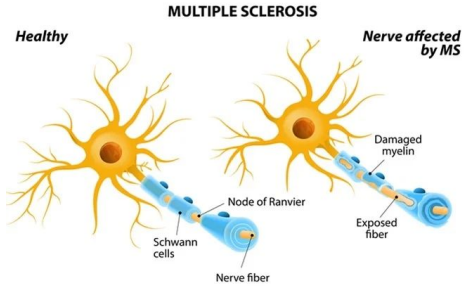
Conclusion

Introduction: why building realistic digital white matter phantoms ?

- Popular microstructural white matter models (AxCaliber, ActiveAxon, NODDI, (SANDI,) …) rely on analytical models which: *Assaf et al 2008; Alexander 2008; Zhang et al 2011; Palombo et al, 2020*
  - use over simplistic representations of white matter microstructure (neurites=cylinders or sticks)
  - embed the glia into the extracellular space

- Efficient to probe the neurite density, the mean axon diameter or the mean neurite orientation dispersion.

- Brain pathologies (neurodegenerative, neurodevelopmental, psychiatric, tumoral) can be associated to other microstructural disorders:
  - damage to myelin (multiple sclerosis)

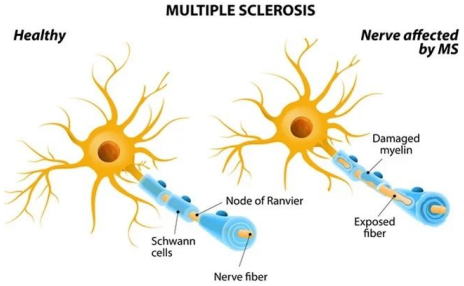


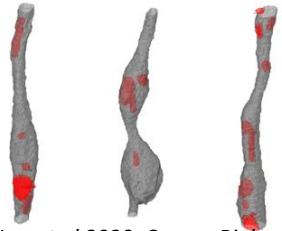
Introduction: why building realistic digial white matter phantoms ?

- Popular microstructural white matter models (AxCaliber, ActiveAxon, NODDI, (SANDI,) …) rely on analytical models which: *Assaf et al 2008; Alexander 2008; Zhang et al 2011; Palombo et al, 2020*
  - use over simplistic representations of white matter microstructure (neurites=cylinders or sticks)
  - embed the glia into the extracellular space

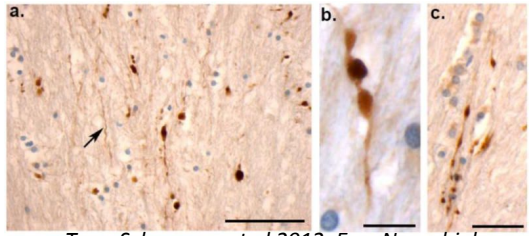
- Efficient to probe the neurite density, the mean axon diameter or the mean neurite orientation dispersion.

- Brain pathologies (neurodegenerative, neurodevelopmental, psychiatric, tumoral) can be associated to other microstructural disorders:
  - damage to myelin (multiple sclerosis)
  - axon caliber variations (beading of axons in stroke, varicosities after brain trauma)





Lee et al 2020, Comm. Biol.



Tang-Schommer et al 2012, Exp. Neurobiol.

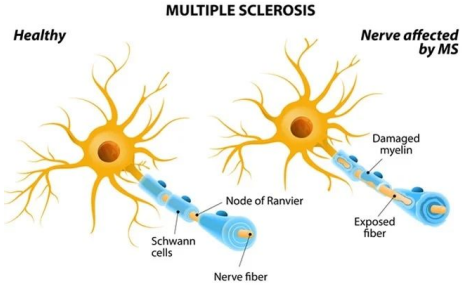
Wu et al et al 2021, Aging and Disease

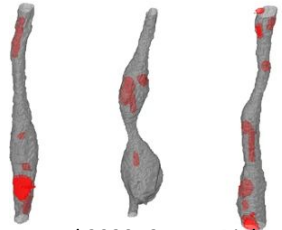
Introduction: why building realistic digital white matter phantoms ?

- Popular microstructural white matter models (AxCaliber, ActiveAxon, NODDI, (SANDI,) …) rely on analytical models which: *Assaf et al 2008; Alexander 2008; Zhang et al 2011; Palombo et al, 2020*
  - use over simplistic representations of white matter microstructure (neurites=cylinders or sticks)
  - embed the glia into the extracellular space

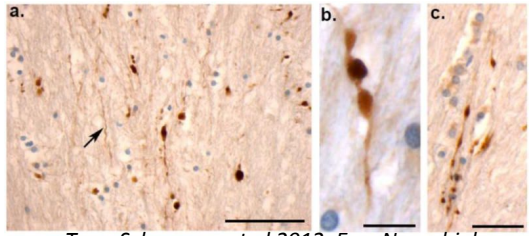
- Efficient to probe the neurite density, the mean axon diameter or the mean neurite orientation dispersion.

- Brain pathologies (neurodegenerative, neurodevelopmental, psychiatric, tumoral) can be associated to other microstructural disorders:
  - damage to myelin (multiple sclerosis)
  - axon caliber variations (beading of axons in stroke, varicosities after brain trauma)
  - activation of microglia in white matter lesions (gliosis, astrocytosis in vascular diseases).

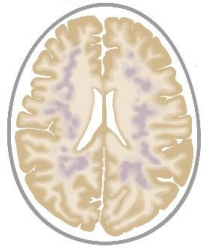


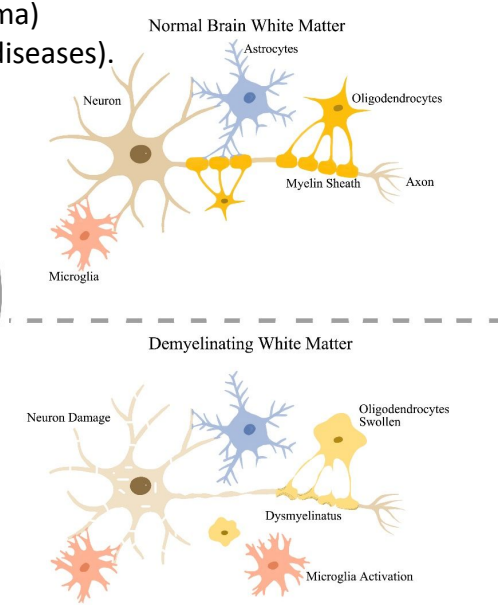


Lee et al 2020, Comm. Biol.



Tang-Schommer et al 2012, Exp. Neurobiol.

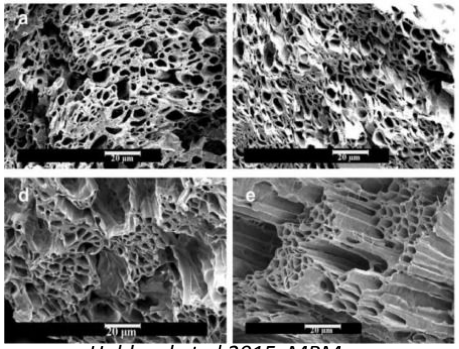




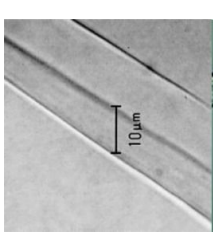
Wu et al et al 2021, Aging and Disease

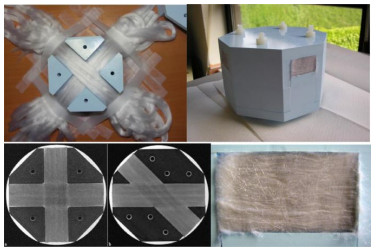
Introduction: why building realistic digital white matter phantoms ?

- Digital versus physical white matter phantoms ?
  - Post-mortem tissues can be used
    - Not easy to get in humans (body donation programs, ethical issues, not all pathological tissue cases)
    - Except for animals, tissue integrity can be questioned
    - Fixation-induced modifications of the tissue microstructure
  - Hardware phantoms can be designed
    - Use of synthetic fibers (acrylic, dyneema, …)
    - Electrospun hollow fibers
    - Melt spinning fibers
    - Not easy control over the geometry, oversimplistic

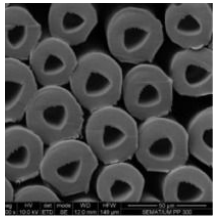


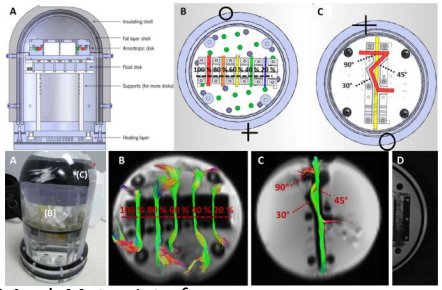
Hubbard et al 2015, MRM



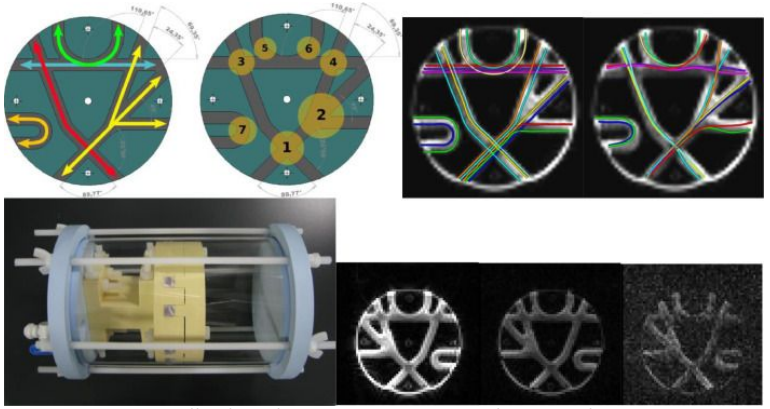


Poupon et al 2008, MRM





Guise et al 2016, ACS Appl. Mater. Interfaces



Fillard et al 2011, NeuroImage, Fiber Cup Phantom

Outline

Introduction: why building realistic digital white matter phantoms ?

Existing digital white matter phantom generators

The MEDUSA approach: combining realism and computational efficiency

Large scale simulations with the MEDUSA simulator

Conclusion

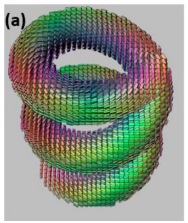
Existing digital white matter phantom generators

- 2 review papers about physical and digital white matter phantoms:
  - Fieremans et al. 2018. NeuroImage. Physical and Numerical Phantoms for the validation of brain microstructural MRI: A Cookbook
  - Drobnjak et al. 2021. Physical and digital phantoms for validating tractography and assessing artifacts

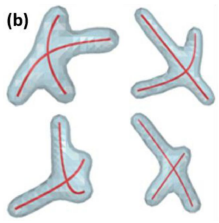
- They point out the question of scale:
  - digital phantoms dedicated to tractography validation
    - large field of view (similar to the brain size)
    - good representation of orientation distribution functions of white matter fiber bundles
  - digital phantoms dedicated to tissue microstructure investigation
    - limited (due to complexity) field of view (eg 1mm^3 voxel)
    - precise and realistic cellular content

Existing digital white matter phantom generators

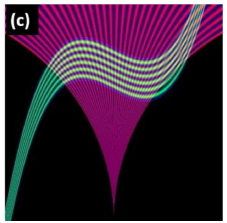
- Overview of existing digital white matter phantoms dedicated to tractography validation



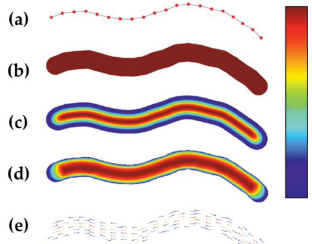
Tend helix tool,
DTI model
Teem 2021



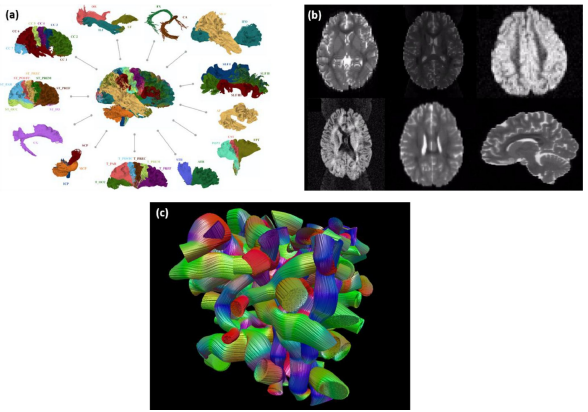
Randomized fiber crossings
Cetingul et al 2012,
IEEE EMBC



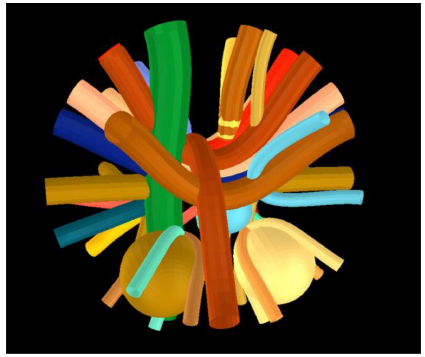
Crossing and fanning
phantom + DTI model
Aganj et al 2011, MIA



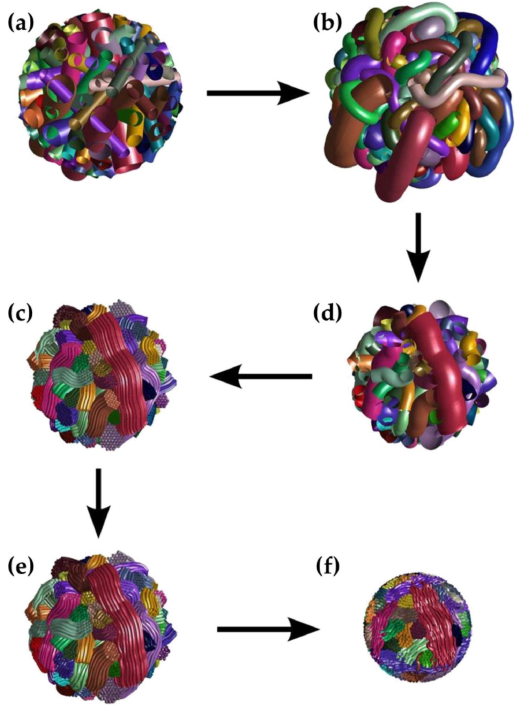
Mathematical framework to
design bundles, DTI model
Leemans et al 2005, MRM



Fiberfox (MITK), ISMRM Tractography Challenge 2015
Streamlines + stick, tensor, astrostick, ball or dot models
including imaging artifacts
Neher et al 2014, MRM



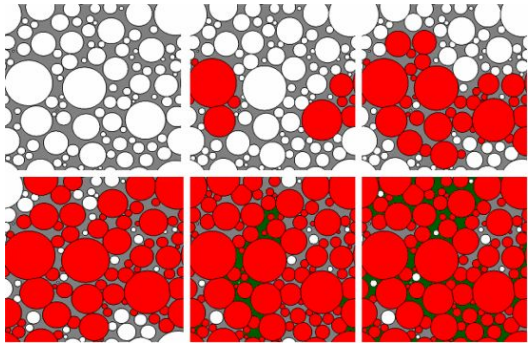
Phantomas, 2nd HARDI Recons. challenge, Tubes + CHARMED model
http://www.emmanuelcaruyer.com/phantomas.php.



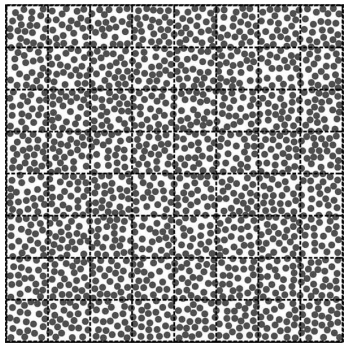
Numerical FIber Generator (NFG)
Tubes + Prolate tensors
Cloose et al 2009, NeuroImage

Existing digital white matter phantom generators

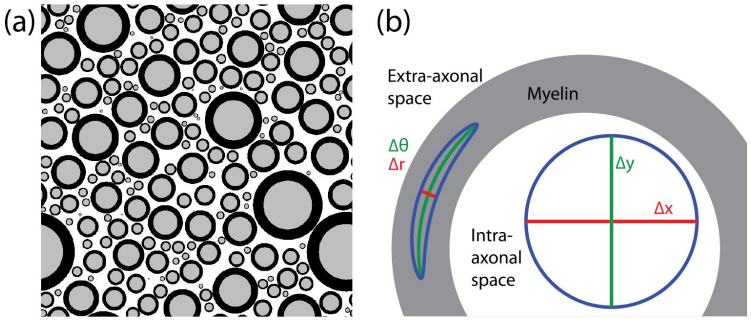
- Overview of existing digital white matter phantoms dedicated to tissue microstructure investigation



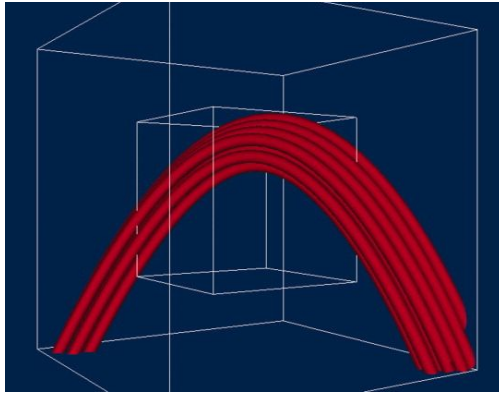
Camino (UCL) 2D & 3D, axon calibration
Monte-Carlo-based simulator
Hall et al 2009, IEEE TMI



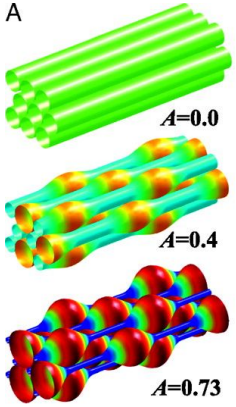
Two-compartment exchange model, 2D
Monte-Carlo-based simulator
Fieremans et al 2010, NMR Biomed.



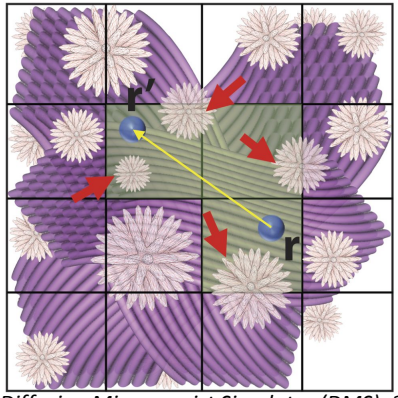
Influence of myelin water, 2D
Monte-Carlo-based simulator
Harkins et al 2016, Physics in Medicine Biology



Complex polygonal fiber configurations 3D
Monte-Carlo-based simulator
Ball et al 2009, MRM



Axonal beading 3D
Monte-Carlo-based simulator
Budde et al 2010, PNAS

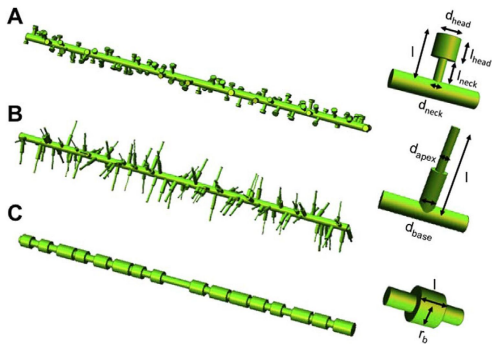


Diffusion Microscopist Simulator (DMS), 3D
Monte-Carlo-based simulator
Yeh et al 2013, Plos-ONE

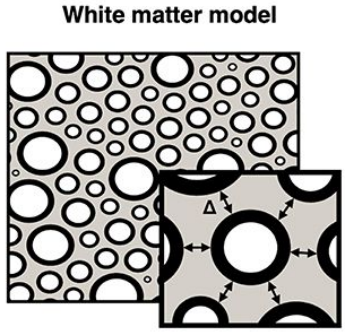
Improving realism of fibers, 3D
Time-dependency study
Ginsburger et al 2018, Frontiers

Existing digital white matter phantom generators

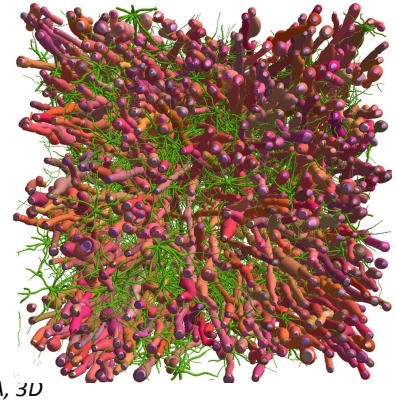
- Overview of existing digital white matter phantoms dedicated to tissue microstructure investigation



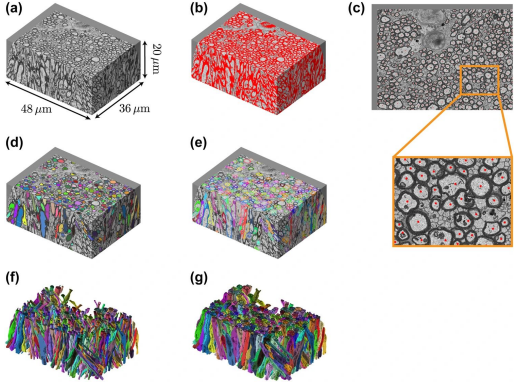
Sensitivity to spine and leaflets, 3D
Monte-Carlo-based simulator
Palombo et al 2018, NeuroImage



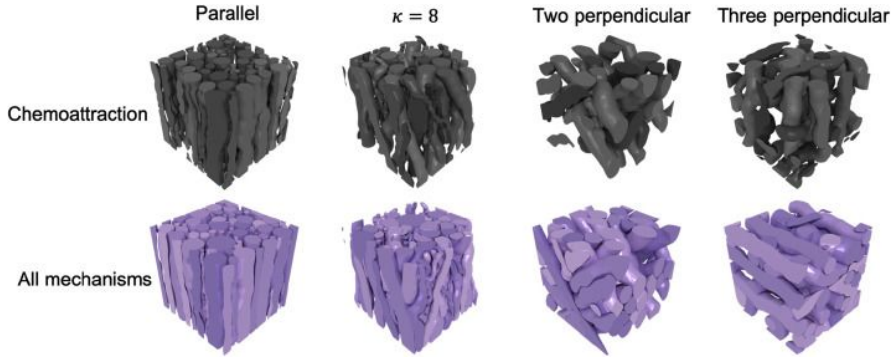
Axon packing simulator, 2D
https://github.com/neuropoly/axonpacking.
Mingasson et al 2017, Frontiers



MEDUSA, 3D
Generative geometry and Monte-Carlo-based simulator
Ginsburger et al 2019, NeuroImage



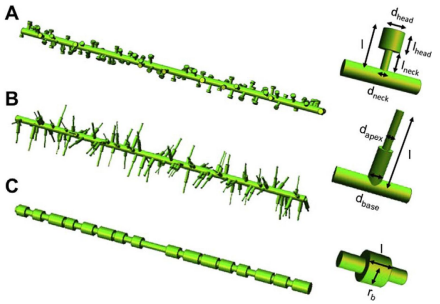
Diameter variation and axonal orientation dispersion
Monte-Carlo-based simulator from electron microscopy
Lee et al 2019, BRain Struct. And Function



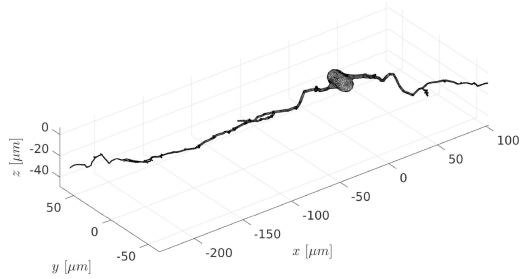
ConFiG, 3D
Generative fibre growth of axons
Palombo et al 2020, NeuroImage

Existing digital white matter phantom generators

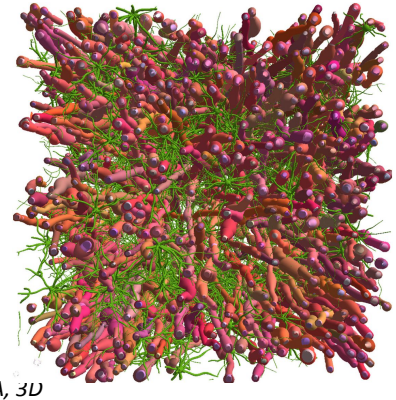
- Overview of existing digital white matter phantoms dedicated to tissue microstructure investigation



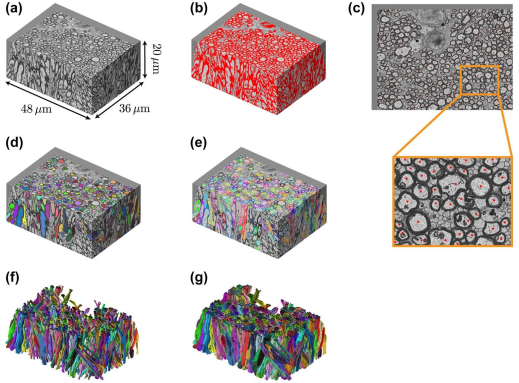
Sensitivity to spine and leaflets, 3D
Monte-Carlo-based simulator
Palombo et al 2018, NeuroImage



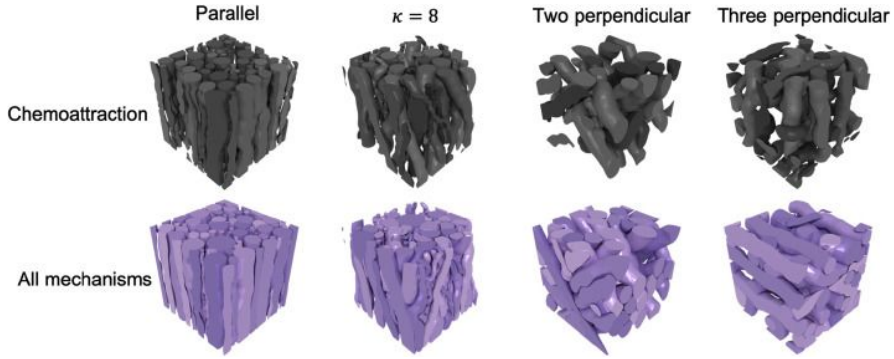
SpinDoctor, 3D
Simulation of single neurons, finite elements
Fang et al 2020, NeuroImage



MEDUSA, 3D
Generative geometry and Monte-Carlo-based simulator
Ginsburger et al 2019, NeuroImage



Diameter variation and axonal orientation dispersion
Monte-Carlo-based simulator from electron microscopy
Lee et al 2019, BRain Struct. And Function



ConFiG, 3D
Generative fibre growth of axons
Palombo et al 2020, NeuroImage

Introduction: why building realistic digital white matter phantoms ?

Existing digital white matter phantom generators

The MEDUSA approach: combining realism and computational efficiency

Large scale simulations with the MEDUSA simulator

Conclusion

The MEDUSA approach: combining realism and computational efficiency

Key components of a diffusion simulator:

- a simulator of **cell membrane geometry and cell populations**
  - scalability: single cell, cell populations within a voxel, whole brain
  - online collections (NeuroMorpho.Org, https://h01-release.storage.googleapis.com/gallery.html) / generative models
  - a packing and overlapping removal algorithm / a growing under constraints algorithm
  - some physical membrane properties (mechanical, permeability, hydrophobia, …)
  - representation of membrane surfaces (Cartesian grids, surface meshes, sphere atoms, …)
- a simulator of the **diffusion process of water molecules or metabolites**
  - Monte-Carlo simulation
  - Finite element discretization and adaptive time integration to solve the Bloch-Torrey partial differential equation
  - interaction at membrane surface
- a simulator of **diffusion-weighted NMR pulse sequences** and of the **diffusion NMR signal decay**
  - genericity wrt pulse sequence writing and access to a dictionary of diffusion sequences
  - synthesis of the signal decay in case of Monte-Carlo simulation

The MEDUSA approach: combining realism and computational efficiency

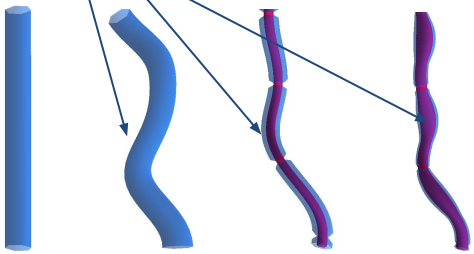
Key components of the MEDUSA (Microstructure Environment Design Using Sphere Atoms) simulator:

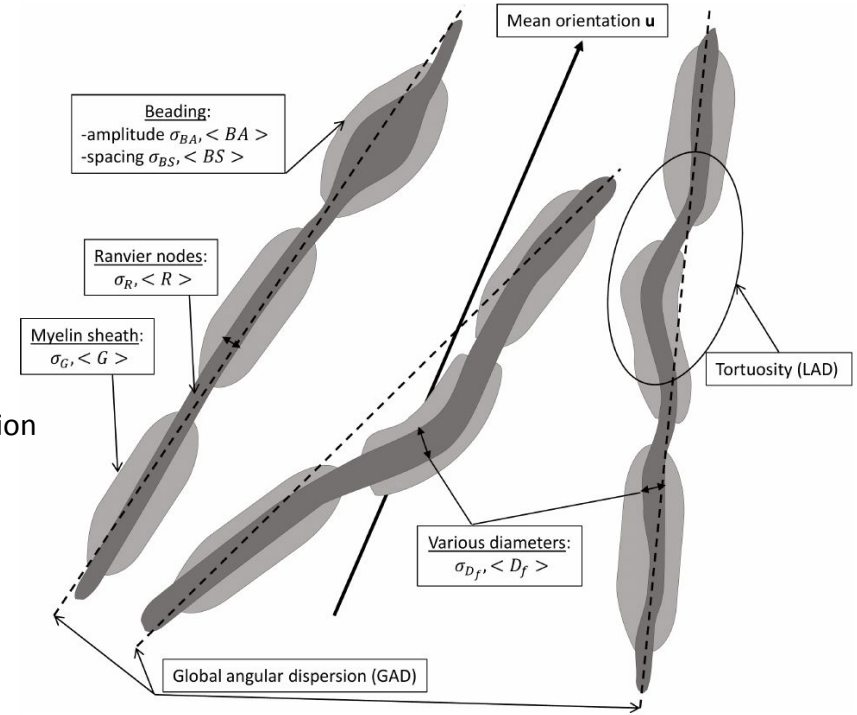
- a simulator of **cell membrane geometry and cell populations**
  - scalability: single cell, cell populations within a voxel, whole brain
  - online collections (NeuroMorpho.Org, https://h01-release.storage.googleapis.com/gallery.html) / generative models
  - a packing and overlapping removal algorithm / a growing under constraints algorithm
  - some physical membrane properties (mechanical, permeability, hydrophobia, …)
  - representation of membrane surfaces (Cartesian grids, surface meshes, sphere atoms, …)
- a simulator of the **diffusion process of water molecules or metabolites**
  - Monte-Carlo simulation
  - Finite element discretization and adaptive time integration to solve the Bloch-Torrey partial differential equation
  - interaction at membrane surface
- a simulator of **diffusion-weighted NMR pulse sequences** and of the **diffusion NMR signal decay**
  - a generic pulse sequence writing and access to a dictionary of diffusion sequences
  - a phase integration to synthesize the signal decay (for Monte-Carlo approach only)

MEDUSA **generative models** of cells:

Geometrical characteristics of each cell model are drawn from distributions whose generative parameters are chosen by the operator.

- **fiber population** (of myelinated axons)
  - axons
    - diameter distribution
    - global angular dispersion
    - tortuosity
    - diameter variation / beading
    - permeability
    - mean orientation
  - myelin sheath
    - g-ratio distribution
    - internodal length to node width distribution
  - volume fraction
  - has tortuosity or not
  - has myelin sheath or not
  - has beading or not



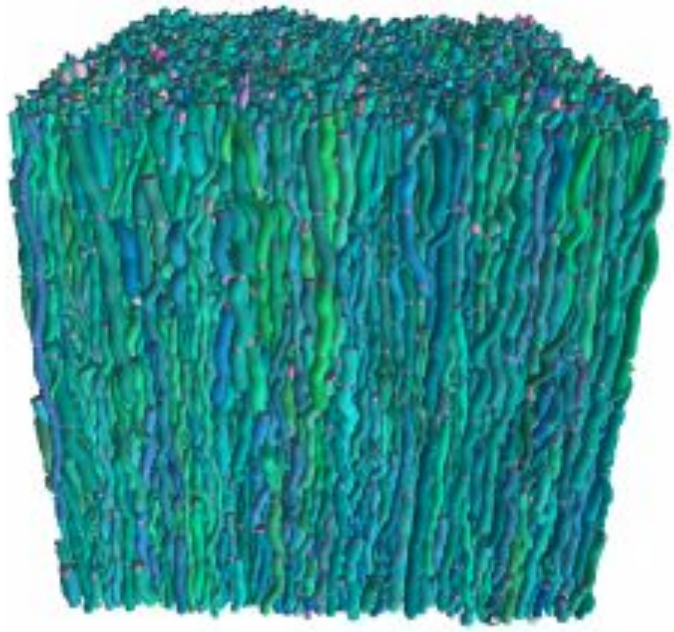


The MEDUSA approach: combining realism and computational efficiency

geometry.json


```
attributes =
{
  "apply_add_details": 1,
  "apply_atom_regularization": 1,
  "apply_remove_overlaps": 1,
  "field_of_view": [
    -10.0,
    -10.0,
    -10.0,
    10.0,
    10.0,
    10.0
  ],
  "grid_resolution": 0.25,
  "maximum_atom_count": 5000000,
  "maximum_force_norm": 0.25,
  "minimum_repulsion_force_attenuation_percentage": 10.0,
  "populations": {
    "fiber_population_01": {
      "parameters": {
        "axon": {
          "beading": {
            "interbeading_length_distribution": {
              "parameters": {
                "mean": 12.0,
                "stddev": 4.0
              },
              "type": "gamma-distribution"
            },
            "magnitude_ratio_distribution": {
              "parameters": {
                "mean": 1.0826034921544718,
                "stddev": 0.001
              },
              "type": "gamma-distribution"
            },
            "width_distribution": {
              "parameters": {
                "mean": 3.0,
                "stddev": 0.8
              },
              "type": "gamma-distribution"
            }
          },
          "diameter_distribution": {
            "parameters": {
              "mean": 0.5,
              "multiplicator": 1000.0,
              "stddev": 0.05,
              "upper_boundary": 10.0
            },
            "type": "gamma-distribution"
          },
          "global_angular_dispersion_in_degrees": 0.09432968145757836,
          "has_beading": 0,
          "has_tortuosity": 1,
          "mean_orientation": [
            0.0,
            0.0,
            1.0
          ],
          "permeability_distribution": {
            "parameters": {
              "mean": 2.0,
              "stddev": 1.0
            },
            "type": "gamma-distribution"
          },
          "tortuosity": {
            "angular_dispersion_in_degrees": 2.5529705916310665,
            "magnitude": 0.02,
            "wave_length": 25.0
          }
        },
        "has_myelin_sheath": 1,
        "myelin_sheath": {
          "g_ratio_distribution": {
            "parameters": {
              "mean": 0.6,
              "stddev": 0.06
            },
            "type": "gamma-distribution"
          },
          "has_ranvier_nodes": 1,
          "internodal_length_to_node_width_distribution": {
            "parameters": {
              "mean": 40.0,
              "stddev": 0.5
            },
            "type": "gamma-distribution"
          }
        },
        "sphere_atom_oversampling_ratio": 20.0,
        "volume_fraction": 0.11284638560138271
      },
      "type": "fiber-population"
    }
  },
  "repulsion_force_moving_average_window_size": 10,
  "repulsion_force_stddev_percentage_threshold": 1.0,
  "sphere_atom_overlap_solver_maximum_iteration_count": 500
}
```


GkgExecuteCommand **Medusa**
-geometryInputDictionary geometry.json
-geometryOutputRaw fibers.medusageometry



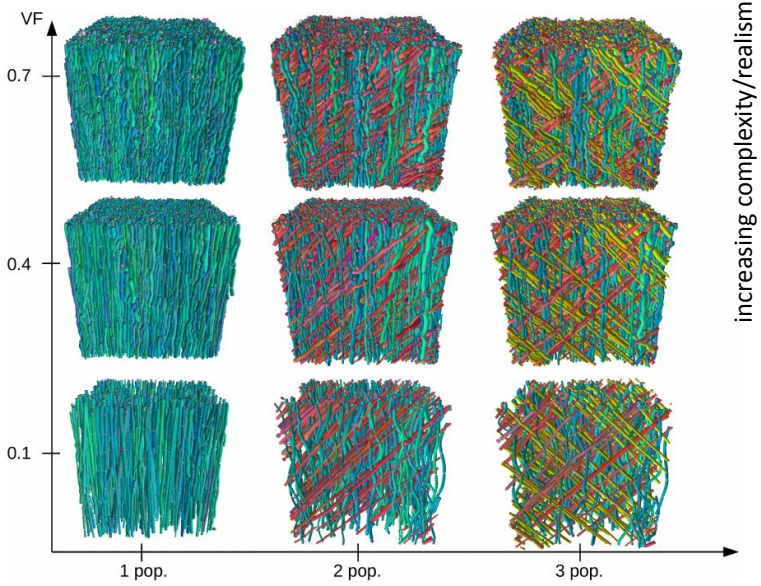
fibers.medusageometry

The MEDUSA approach: combining realism and computational efficiency

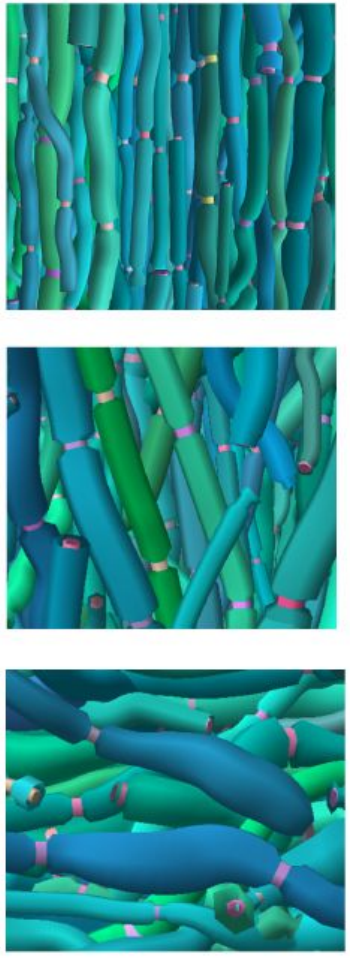
geometry.json

```
attributes =
{
  "apply_add_details": 1,
  "apply_atom_regularization": 1,
  "apply_remove_overlaps": 1,
  "field_of_view": [
    -10.0,
    -10.0,
    -10.0,
    10.0,
    10.0,
    10.0
  ],
  "grid_resolution": 0.25,
  "maximum_atom_count": 5000000,
  "maximum_force_norm": 0.25,
  "minimum_repulsion_force_attenuation_percentage": 10.0,
  "populations": {
    "fiber_population_01": {
      "parameters": {

        .......

      },
      "type": "fiber-population"
    },
    "fiber_population_02": {
      "parameters": {

        .......

      },
      "type": "fiber-population"
    },
    "fiber_population_03": {
      "parameters": {

        .......

      },
      "type": "fiber-population"
    }

  },
  "repulsion_force_moving_average_window_size": 10,
  "repulsion_force_stddev_percentage_threshold": 1.0,
  "sphere_atom_overlap_solver_maximum_iteration_count": 500
}
```

GkgExecuteCommand **Medusa**
-geometryInputDictionary geometry.json
-geometryOutputRaw fibers.medusageometry



increasing complexity/realism



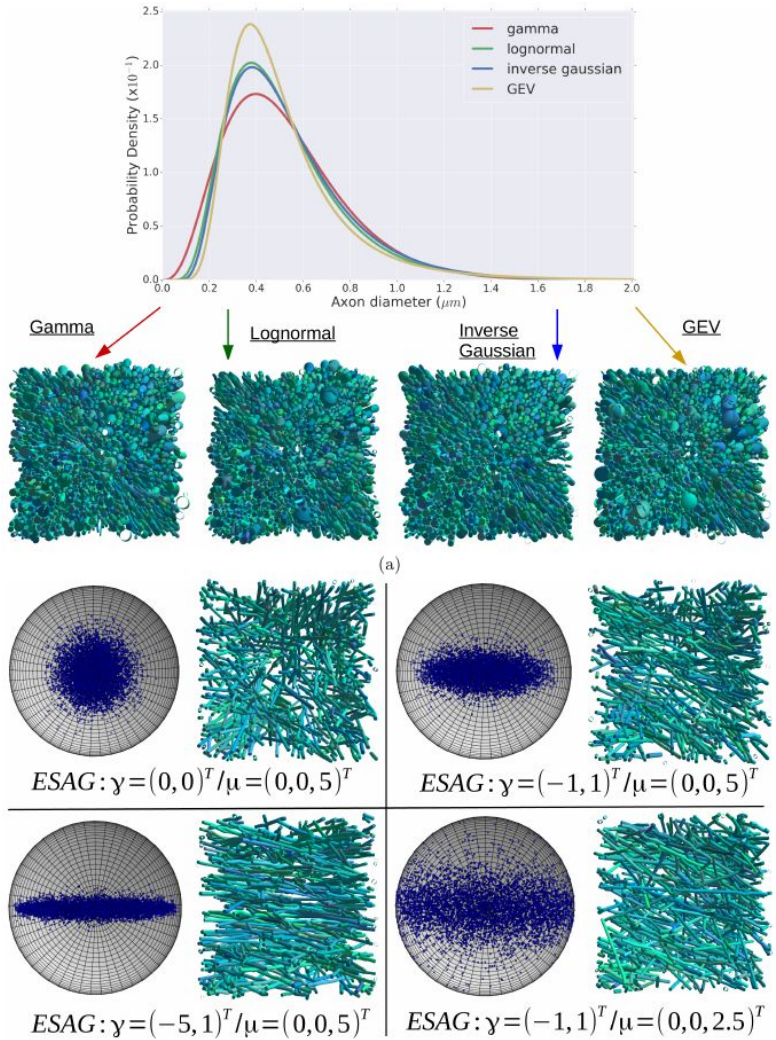
The MEDUSA approach: combining realism and computational efficiency

Example of fine tuning of the diameter distribution type with:

- Gamma distributions
- LogNormal distributions
- Inverse Gaussian distribution
- Generalized Extreme Value distributions

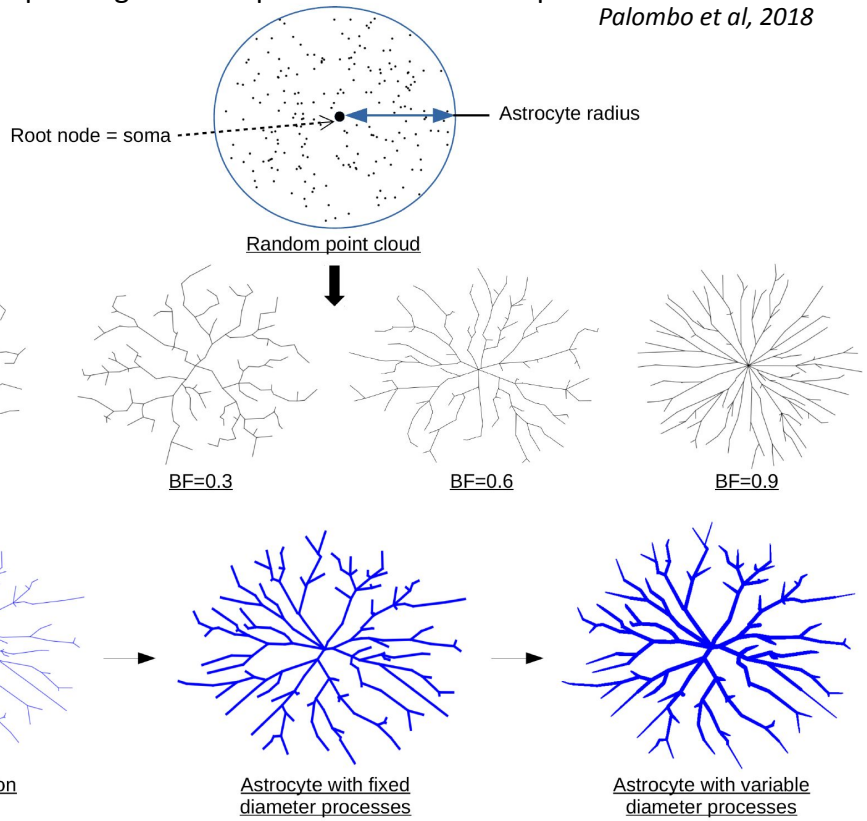
And of the global angular dispersion with:

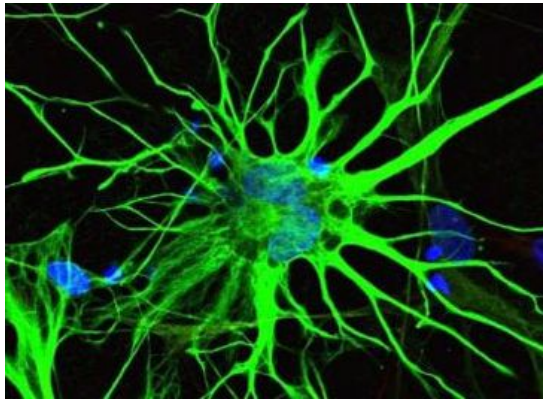
- Watson distributions
- Elliptical Symmetric Angular Gaussian distributions

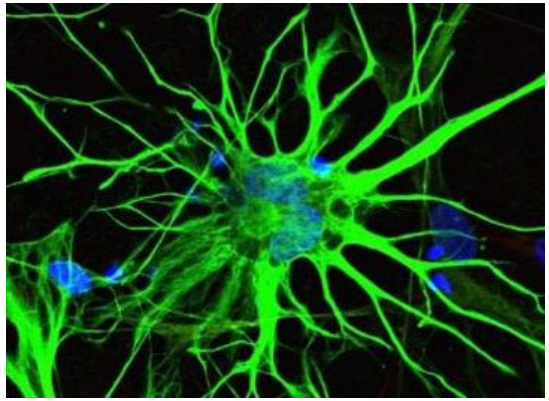


- **astrocyte population** reconstructed from a Minimum Spanning Tree computed over a random point cloud
  - soma
    - diameter distribution
    - permeability distribution
  - process
    - diameter distribution
    - balancing factor
    - node count distribution
    - tortuosity
    - permeability distribution
  - total diameter distribution
  - volume fraction

Palombo et al, 2018

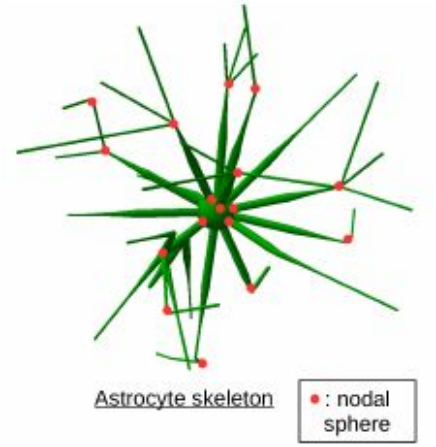




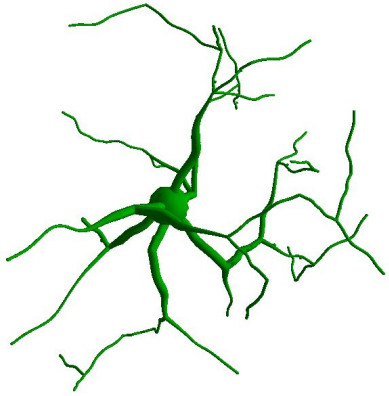


- **astrocyte population**
  - soma
    - diameter distribution
    - permeability distribution
  - process
    - diameter distribution
    - balancing factor
    - node count distribution
    - tortuosity
    - permeability distribution
  - total diameter distribution
  - volume fraction

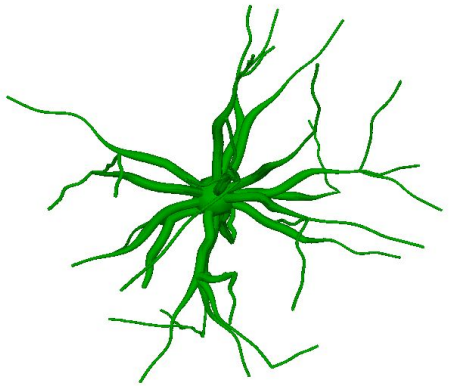
straight branches



fibrous astrocyte type I



fibrous astrocyte type II

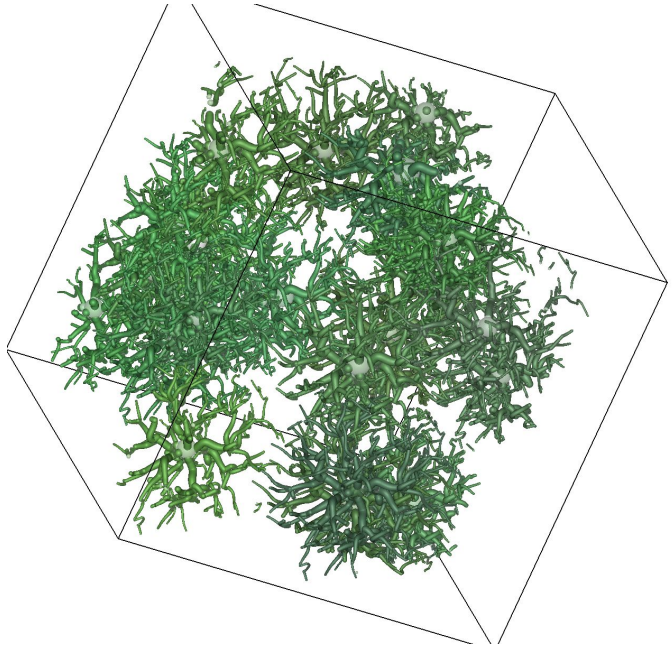


The MEDUSA approach: combining realism and computational efficiency

geometry.json

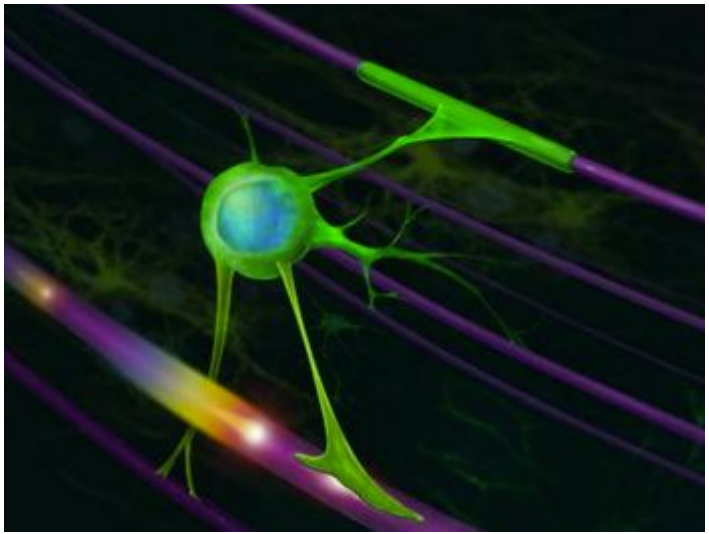
```
attributes =
{
  "apply_add_details": 1,
  "apply_atom_regularization": 1,
  "apply_remove_overlaps": 1,
  "field_of_view": [
    -10.0,
    -10.0,
    -10.0,
    10.0,
    10.0,
    10.0
  ],
  "grid_resolution": 0.25,
  "maximum_atom_count": 5000000,
  "maximum_force_norm": 0.25,
  "minimum_repulsion_force_attenuation_percentage": 10.0,
  "populations": {
    "astrocyte-population-01" : {
      "parameters": {
        "soma" : {
          "diameter_distribution" : {
            "type" : "gamma-distribution",
            "parameters" : {
              "mean" : 2.0,
              "stddev" : 0.2,
              "multiplicator" : 1e3,
              "upper_boundary" : 10.0
            }
          },
          "permeability_distribution" : {
            "type" : "gamma-distribution",
            "parameters" : {
              "mean" : 0.5,
              "stddev" : 0.1
            }
          }
        },
        "process" : {
          "atom_oversampling_ratio" : 10.0,
          "minimum_segment_point_count" : 20,
          "diameter_distribution" :{
            "type" : "gamma-distribution",
            "parameters" : {
              "mean" : 0.7,
              "stddev" : 0.1,
              "multiplicator" : 1e3,
              "upper_boundary" : 1.0
            }
          },
          "balancing_factor" : 0.7,
          "node_count_distribution" : {
            "type" : "gamma-distribution",
            "parameters" : {
              "mean" : 100.0,
              "stddev" : 4.0,
              "multiplicator" : 1e3,
              "upper_boundary" : 200.0
            }
          },
          "has_tortuosity" : 1,
          "tortuosity" : {
            "magnitude" : 0.9,
            "angular_dispersion_in_degrees" : 1.5,
            "wave_length" : 10.0
          },
          "permeability_distribution" : {
            "type" : "gamma-distribution",
            "parameters" : {
              "mean" : 0.5,
              "stddev" : 0.1
            }
          },
          "total_diameter_distribution" : {
            "type" : "gamma-distribution",
            "parameters" : {
              "mean" : 25.0,
              "stddev" : 5.0,
              "multiplicator" : 1e3,
              "upper_boundary" : 50.0
            }
          },
          "volume_fraction" : 0.01,
        },
        "type" : "astrocyte-population"
      }
    }
    "repulsion_force_moving_average_window_size": 10,
    "repulsion_force_stddev_percentage_threshold": 1.0,
    "sphere_atom_overlap_solver_maximum_iteration_count": 500
}
```

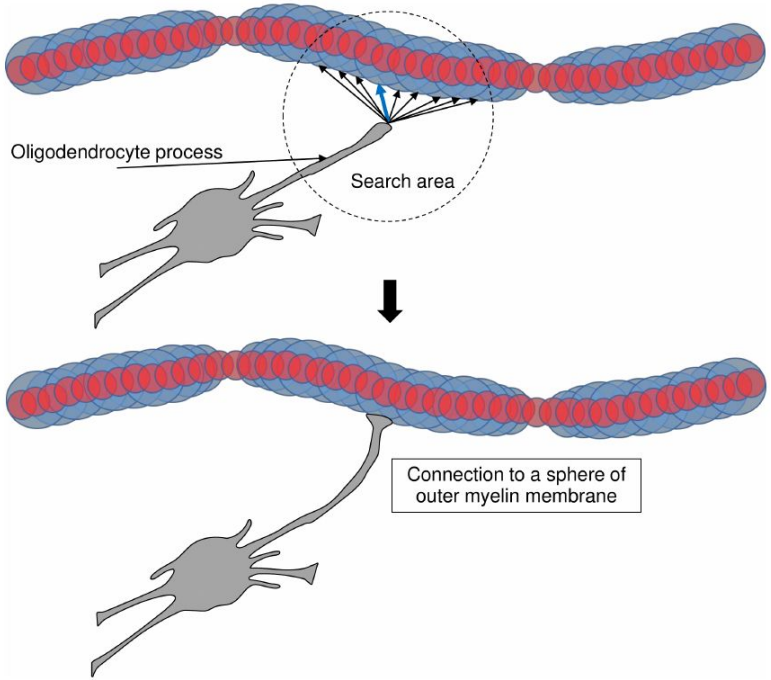
GkgExecuteCommand **Medusa**
-geometryInputDictionary geometry.json
-geometryOutputRaw astrocytes.medusageometry



astrocytes.medusageometry

- **oligodendrocyte population** similar to astrocytes (no branching) but with connections to fibers
  - soma
    - diameter distribution
    - permeability distribution
  - process
    - diameter distribution
    - node count distribution
    - tortuosity
    - permeability distribution
  - myelinated fiber search radius distribution
  - volume fraction



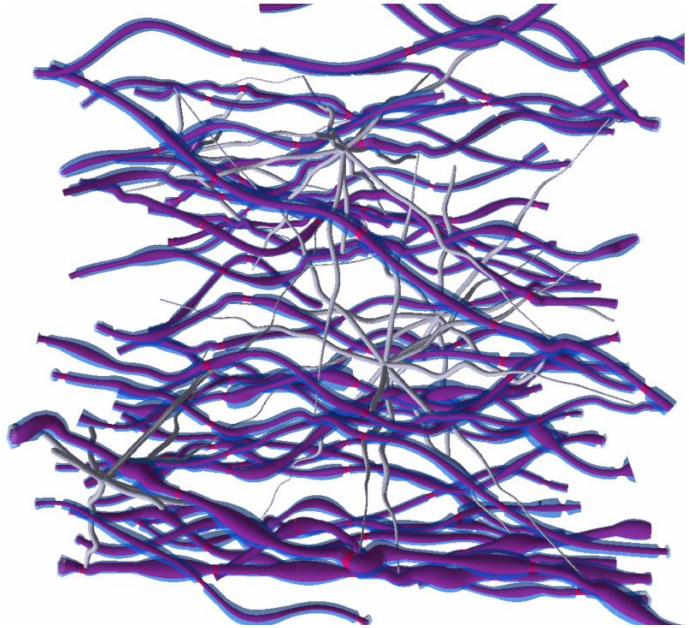


The MEDUSA approach: combining realism and computational efficiency

geometry.json

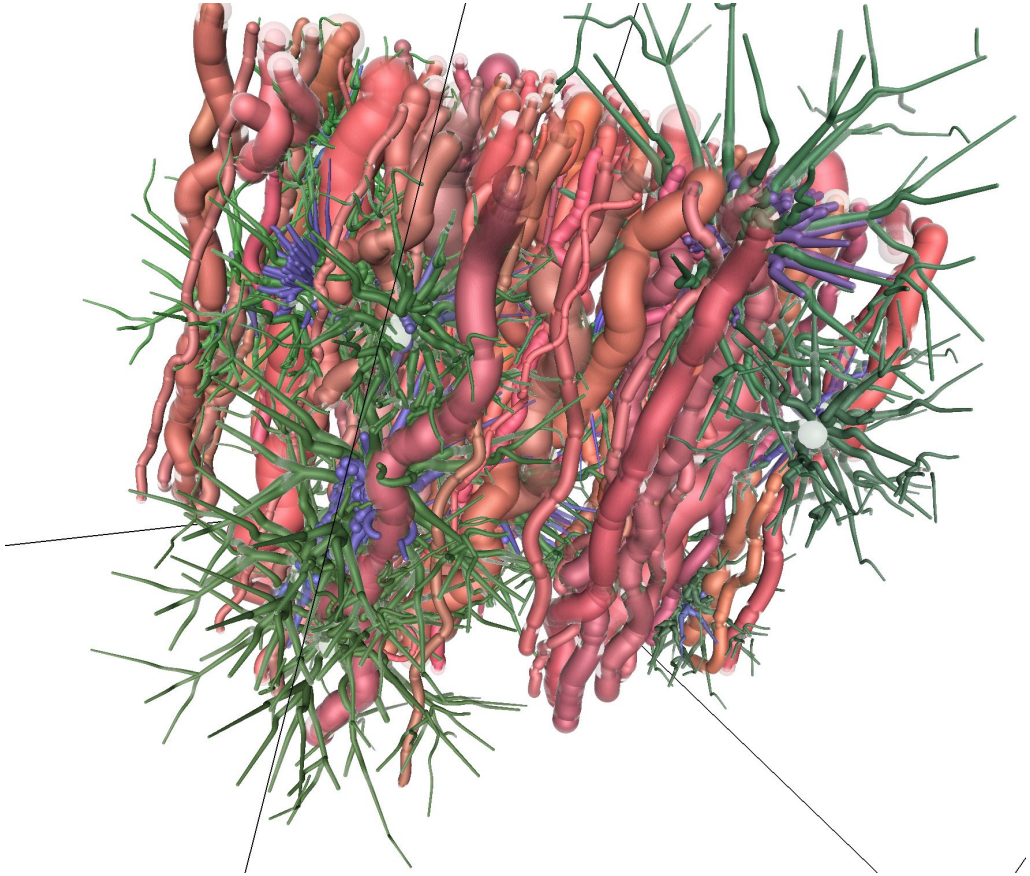
```
attributes =
{
  "apply_add_details": 1,
  "apply_atom_regularization": 1,
  "apply_remove_overlaps": 1,
  "field_of_view": [
    -10.0,
    -10.0,
    -10.0,
    10.0,
    10.0,
    10.0
  ],
  "grid_resolution": 0.25,
  "maximum_atom_count": 5000000,
  "maximum_force_norm": 0.25,
  "minimum_repulsion_force_attenuation_percentage": 10.0,
  "populations": {
     "oligodendrocyte-population-01" : {
      "parameters": {
        "soma" : {
          "diameter_distribution" : {
            "type" : "gamma-distribution",
            "parameters" : {
              "mean" : 1.5,
              "stddev" : 0.7,
              "multiplicator" : 1e3,
              "upper_boundary" : 3.0
            }
          },
          "permeability_distribution" : {
            "type" : "gamma-distribution",
            "parameters" : {
              "mean" : 0.5,
              "stddev" : 0.1
            }
          }
        },
        "process" : {
          "atom_oversampling_ratio" : 10.0,
          "minimum_segment_point_count" : 20,
          "diameter_distribution" :{
            "type" : "gamma-distribution",
            "parameters" : {
              "mean" : 0.7,
              "stddev" : 0.1,
              "multiplicator" : 1e3,
              "upper_boundary" : 1.0
            }
          },
          "node_count_distribution" : {
            "type" : "gamma-distribution",
            "parameters" : {
              "mean" : 50.0,
              "stddev" : 4.0,
              "multiplicator" : 1e3,
              "upper_boundary" : 100.0
            }
          },
          "has_tortuosity" : 1,
          "tortuosity" : {
            "magnitude" : 0.2,
            "angular_dispersion_in_degrees" : 1.5,
            "wave_length" : 15.0
          },
          "permeability_distribution" : {
            "type" : "gamma-distribution",
            "parameters" : {
              "mean" : 0.5,
              "stddev" : 0.1
            }
          }
        },
        "myelinated_fiber_search_radius_distribution" : {
          "type" : "gamma-distribution",
          "parameters" : {
            "mean" : 20.0,
            "stddev" : 1.5,
            "multiplicator" : 1e3,
            "upper_boundary" : 40.0
          }
        },
        "volume_fraction" : 0.01,
      },
      "type" : "oligodendrocyte-population"
    },
    "fiber_population_01": {
      "parameters": {

        .......

      },
      "type": "fiber-population"
    }
  },
  "repulsion_force_moving_average_window_size": 10,
  "repulsion_force_stddev_percentage_threshold": 1.0,
  "sphere_atom_overlap_solver_maximum_iteration_count": 500
}
```

GkgExecuteCommand **Medusa**
-geometryInputDictionary geometry.json
-geometryOutputRaw
fibers-and-oligodendrocytes.medusageometry



fibers-and-oligodendrocytes.medusageometry

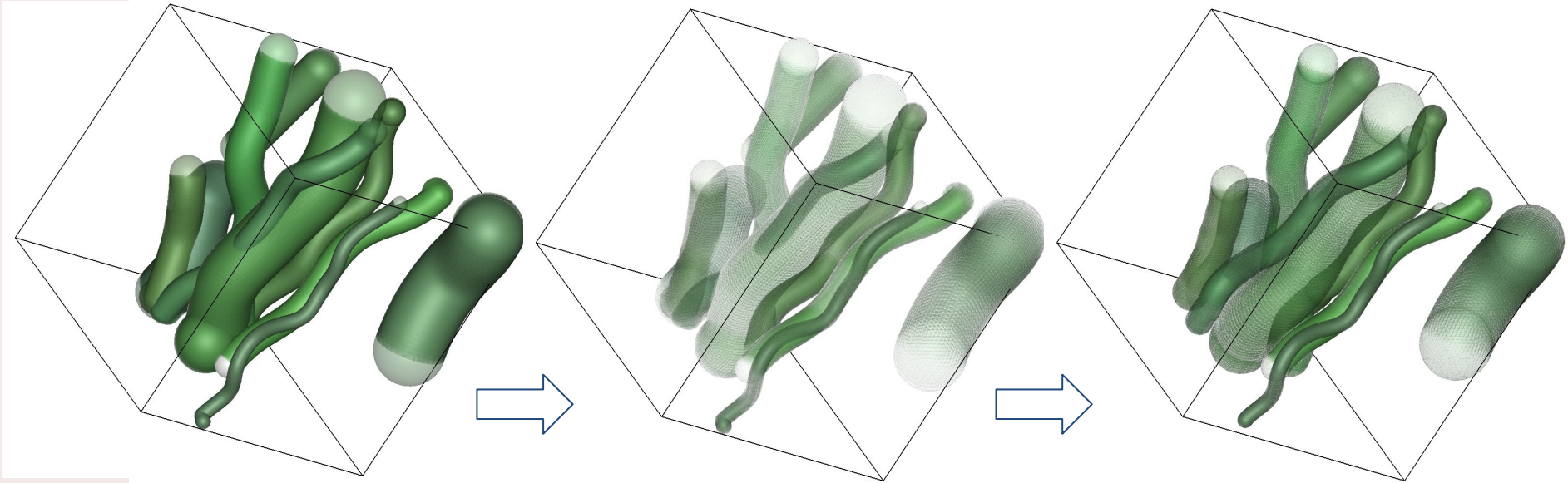
MEDUSA generative models of cells: all in one example including fibers, astrocytes and oligodendrocytes



MEDUSA **packing and overlapping removal** algorithm **with sphere atoms**

- MEDUSA proceeds using a three-fold strategy:
  - randomly generating cells without any overlapping consideration
  - decomposing cells into highly overlapping sphere atoms
  - removing overlaps using a force-based sphere packing algorithm using a 1D sweep and prune algorithm

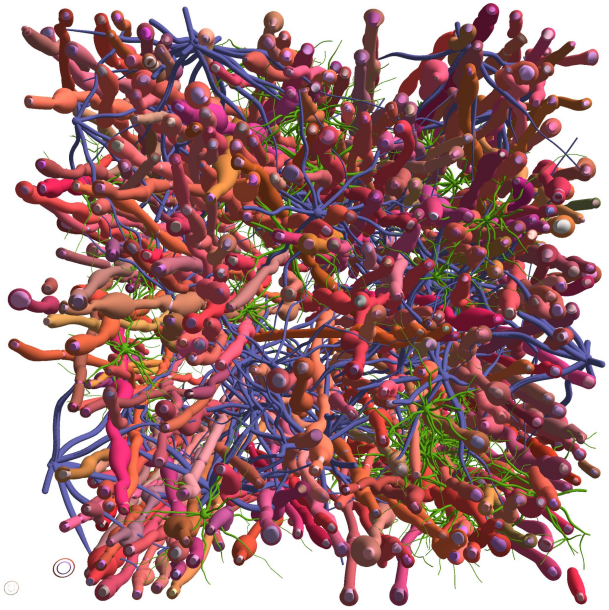
$$O(s_1, s_2) = \max(0, r_1 + r_2 - d(x_1, x_2)) \qquad F(s_1, s_2) = \frac{O(s_1, s_2)}{2} \frac{x_1 - x_2}{|x_1 - x_2|} \qquad F_{tot}(s_1) = \sum_{s \in S} F(s_1, s)$$

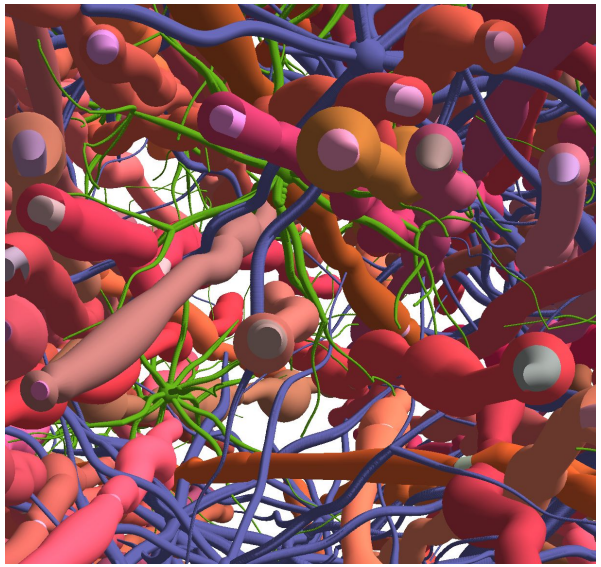


overlapped cells | sphere atom decomposition of cells | overlap removal applying forces to sphere atoms

MEDUSA **packing and overlapping removal** algorithm **with sphere atoms**

- The advantage of the MEDUSA decomposition of cells is four-fold:
  - a more parsimonious representation of cells than mesh surfaces
  - a generic approach to decompose cells into sphere atoms (no need to come back to the cell type afterwards if each atom store id(s) to its parent cell and population)
  - an efficient way to compute distance of any object to cells, cell overlaps, and lookup-tables with Cartesian grids storing sphere atoms
  - an efficient way to render scenes in OpenGL with GL lists



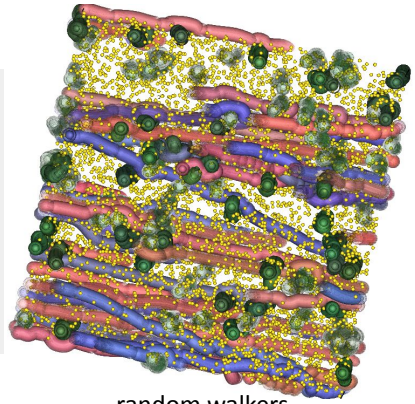


MEDUSA **diffusion process** simulation

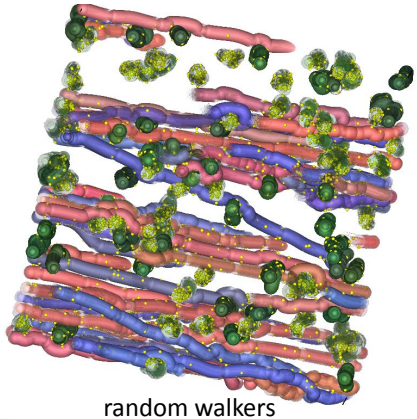
- Monte-Carlo approach
  - distribution of random walkers **everywhere**, **intracellular**, **extracellular**
  - possibility to combine several Monte-Carlo sessions to avoid memory overload
- Computationally efficient with the Sphere Atom decomposition using LUT (eg simple distance to spheres)

diffusion-process.json

```
attributes = {
  'particle_initialization' : 'everywhere',
  'session_count' : 10,
  'particle_diffusivity_in_um2_per_us' : 2e-3,
  'particle_count' : 100000,
  'grid_resolution_in_um' : ( 1.0, 1.0, 1.0 ), # in general, it is ( precision *
axon_diameter )^2 / ( 6 * particle_diffusivity ) with precision = 0.1
  'time_step_in_us' : 1.0,
  'total_duration_in_us' : 240000.0
}
```

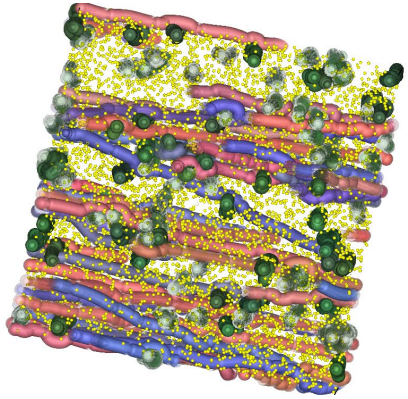


random walkers
everywhere



random walkers
intracellular only

random walkers
extracellular only



GkgExecuteCommand **Medusa**
-geometryInputDictionary geometry.json
-geometryOutputRaw fibers.medusageometry
-diffusionProcessInputDictionary diffusion-process.json
-diffusionProcessOutputParticles particles.pointcloud

particles.pointcloud

MEDUSA **diffusion process** simulation

- Monte-Carlo approach
  - distribution of random walkers **everywhere**, **intracellular**, **extracellular**
  - possibility to combine several Monte-Carlo sessions to avoid memory overload
- Computationally efficient with the Sphere Atom decomposition using LUT (eg simple distance to spheres)

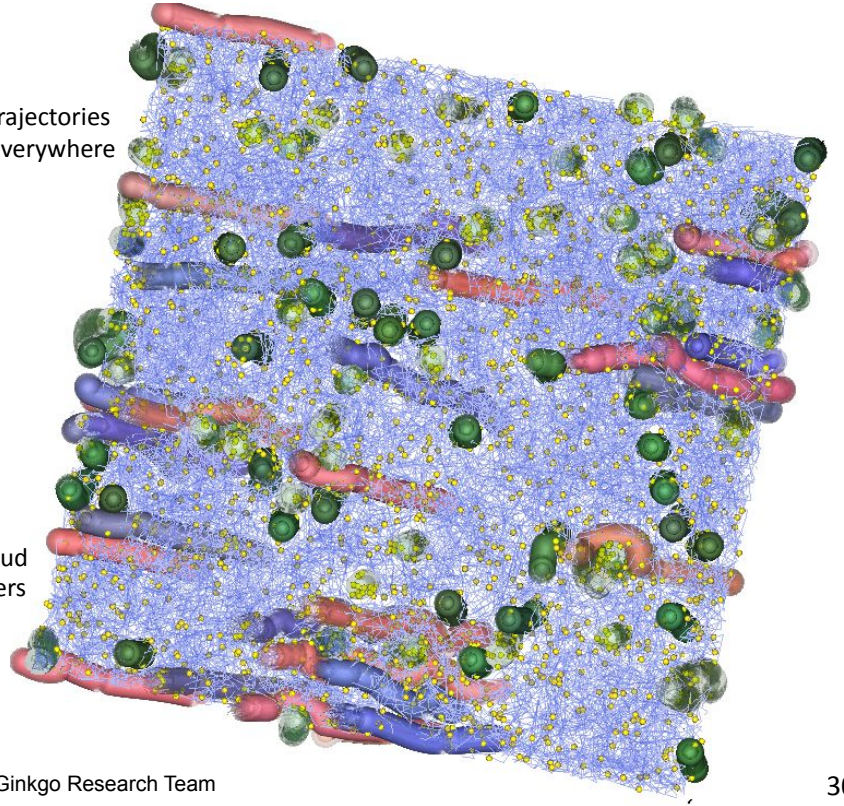
diffusion-process.json

```
attributes = {
  'particle_initialization' : 'everywhere',
  'session_count' : 10,
  'particle_diffusivity_in_um2_per_us' : 2e-3,
  'particle_count' : 100000,
  'grid_resolution_in_um' : ( 1.0, 1.0, 1.0 ), # in general, it is ( precision *
axon_diameter )^2 / ( 6 * particle_diffusivity ) with precision = 0.1
  'time_step_in_us' : 1.0,
  'total_duration_in_us' : 240000.0
}
```

GkgExecuteCommand **Medusa**
-geometryInputDictionary geometry.json
-geometryOutputRaw fibers.medusageometry
-diffusionProcessInputDictionary diffusion-process.json
-diffusionProcessOutputParticles particles.pointcloud
-diffusionProcessOutputTrajectories trajectories.clusters

particles.pointcloud
trajectories.clusters

trajectories
everywhere



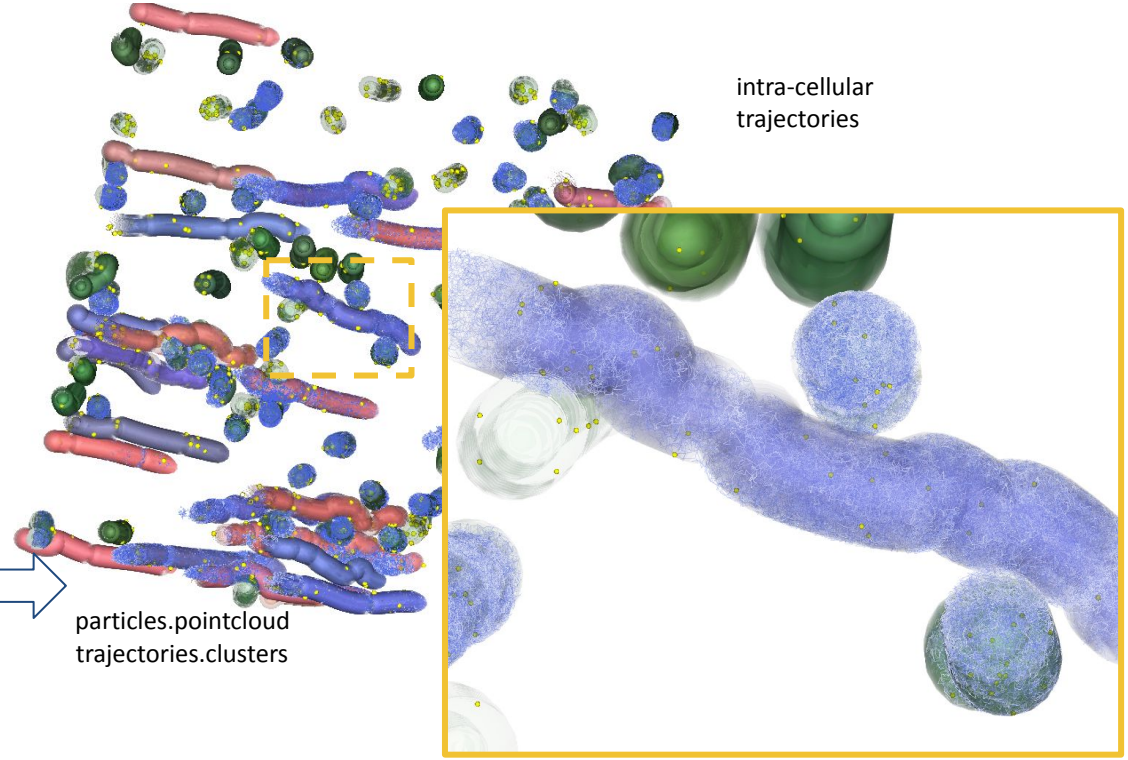
MEDUSA **diffusion process** simulation

- Monte-Carlo approach
  - distribution of random walkers **everywhere**, **intracellular**, **extracellular**
  - possibility to combine several Monte-Carlo sessions to avoid memory overload
- Computationally efficient with the Sphere Atom decomposition using LUT (eg simple distance to spheres)

diffusion-process.json

```
attributes = {
  'particle_initialization' : 'everywhere',
  'session_count' : 10,
  'particle_diffusivity_in_um2_per_us' : 2e-3,
  'particle_count' : 100000,
  'grid_resolution_in_um' : ( 1.0, 1.0, 1.0 ), # in general, it is ( precision *
axon_diameter )^2 / ( 6 * particle_diffusivity ) with precision = 0.1
  'time_step_in_us' : 1.0,
  'total_duration_in_us' : 240000.0
}
```

GkgExecuteCommand **Medusa**
-geometryInputDictionary geometry.json
-geometryOutputRaw fibers.medusageometry
-diffusionProcessInputDictionary diffusion-process.json
-diffusionProcessOutputParticles particles.pointcloud
-diffusionProcessOutputTrajectories trajectories.clusters



intra-cellular
trajectories

particles.pointcloud
trajectories.clusters

MEDUSA **diffusion NMR signal attenuation** simulation **for prescribed diffusion-weighted pulse sequences**

- By default, we **never export particle clouds and trajectories** to preserve memory and disk space
- Various diffusion-weighted sequence schemes implemented in MEDUSA: **PGSE, Sine-OGSE, Cosine-PGSE, t-Sine-OGSE, t-Cosine-OGSE**

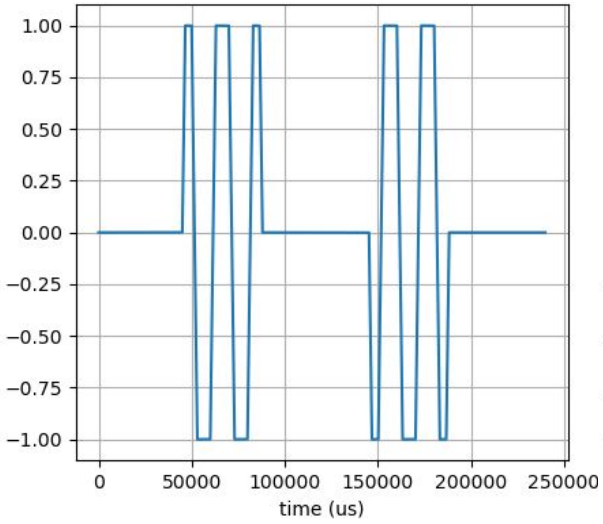
mri-sequences.json

```
attributes = {
  'sequence-01' : {
    'type' : 'ogse',
    'parameters' : {
      'echo_time_in_ms' : 240.0,
      'waveform_type' : 'trapezoid-cosine',
      'time_offset_to_diffusion_module_in_ms' : 45.0,
      'period_in_ms' : 20.0,
      'period_count' : 2,
      'big_delta_in_ms' : 100.3,
      'maximum_gradient_slew_rate_in_tesla_per_m_per_s': 200,
      'gradient_time_resolution_in_us' : 10.0,
      'gradient_magnitudes' : (
        ( 300.0, 100.0, 80.0 ),
        ( 50.0, 20.0 )
       ),
      'gradient_orientations' : (
        ( ( 1.0, 0.0, 0.0 ), ( 0.0, 1.0, 0.0 ), ( 0.0, 0.0, 1.0 ) ),
        ( ( 1.0, 0.0, 0.0 ), ( 0.0, 1.0, 0.0 ) )
       )
     }
   },
  'sequence-02' :  {
    'type' : 'pgse',
    'parameters' : {
      'echo_time_in_ms' : 240.0,
      'time_offset_to_diffusion_module_in_ms' : 45.0,
      'little_delta_in_ms' : 40.3,
      'big_delta_in_ms' : 100.3,
      'maximum_gradient_slew_rate_in_tesla_per_m_per_s': 200,
      'gradient_time_resolution_in_us' : 10.0,
      'gradient_magnitudes' :  (
        ( 300.0, 100.0, 80.0 ),
        ( 50.0, 20.0 )
       ),
      'gradient_orientations' : (
        10,   # instead of orientations, you can give the orientation #
        6
       )
     }
   }
}
```

```
GkgExecuteCommand Medusa
-geometryInputDictionary geometry.json
-diffusionProcessInputDictionary diffusion-process.json
-mriSequenceInputDictionary mri-sequences.json
-mriSequenceOutputNormalizedGradientProfileDirectory
normalized_gradient_profiles
```

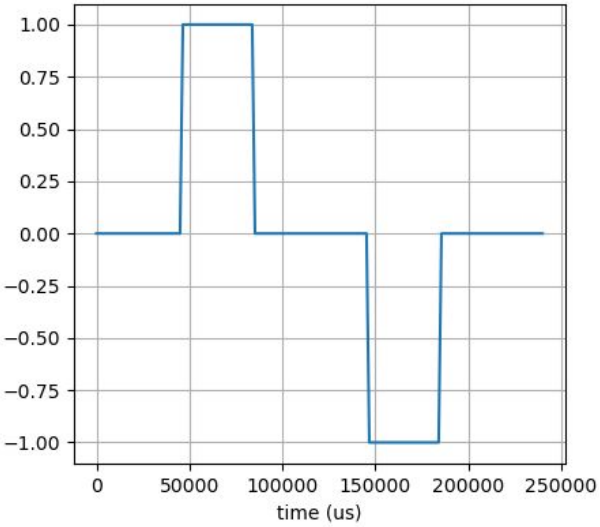
normalized_gradient_profiles/sequence_01.py

gradient profile of t-Cosine-OGSE sequence 01



normalized_gradient_profiles/sequence_02.py

gradient profile of PGSE sequence 02

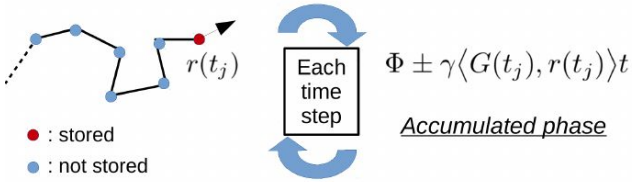


MEDUSA **diffusion NMR signal attenuation** simulation **for prescribed diffusion-weighted pulse sequences**

- **Phase integration** performed **at each time step** for all particles and all target sequence tunings (few memory limitation)
- **Memory consumption** during the joint Monte-Carlo simulation & phase integration : 100k random walkers x 464 phase accumulators = **186 MBytes**!

mri-sequences.json

```
attributes = {
  'sequence-01' : {
    'type' : 'ogse',
    'parameters' : {
      'echo_time_in_ms' : 240.0,
      'waveform_type' : 'trapezoid-cosine',
      'time_offset_to_diffusion_module_in_ms' : 45.0,
      'period_in_ms' : 20.0,
      'period_count' : 2,
      'big_delta_in_ms' : 100.3,
      'maximum_gradient_slew_rate_in_tesla_per_m_per_s': 200,
      'gradient_time_resolution_in_us' : 10.0,
      'gradient_magnitudes' : (
        ( 300.0, 100.0, 80.0 ),
        ( 50.0, 45.0, 35.0, 30.0, 20.0 )
       ),
      'gradient_orientations' : (
        ( ( 1.0, 0.0, 0.0 ), ( 0.0, 1.0, 0.0 ), ( 0.0, 0.0, 1.0 ) ),
        ( ( 1.0, 0.0, 0.0 ), ( 0.0, 1.0, 0.0 ) )
       )
     }
   },
  'sequence-02' :  {
    'type' : 'pgse',
    'parameters' : {
      'echo_time_in_ms' : 240.0,
      'time_offset_to_diffusion_module_in_ms' : 45.0,
      'little_delta_in_ms' : 40.3,
      'big_delta_in_ms' : 100.3,
      'maximum_gradient_slew_rate_in_tesla_per_m_per_s': 200,
      'gradient_time_resolution_in_us' : 10.0,
      'gradient_magnitudes' :  (
        ( 300.0, 200.0, 100.0, 80.0 ),
        ( 50.0, 20.0 )
       ),
      'gradient_orientations' : (
        90,   # instead of orientations, you can give the orientation count
        40
       )
     }
   }
}
```



GkgExecuteCommand **Medusa**
-geometryInputDictionary geometry.json
-diffusionProcessInputDictionary diffusion-process.json
-mriSequenceInputDictionary mri-sequences.json
-mriSequenceOutputDiffusionNMRAttenuation
diffusion-mri-attenuation.nii

diffusion-mri-attenuation.nii

sequence-01: 3 gradient magnitudes x 3 gradient orientations
\+ 5 gradient magnitudes x 3 gradient orientations
sequence-02: 4 gradient magnitudes x 90 gradient orientations
\+ 2 gradient magnitudes x 40 gradient orientations

9+15+360+80 = 464 diffusion NMR signal attenuations

Introduction: why building realistic digital white matter phantoms ?

Existing digital white matter phantom generators

The MEDUSA approach: combining realism and computational efficiency

Large scale simulations with the MEDUSA simulator

Conclusion

MEDUSA **computational efficiency**:

- code implemented in C++ with Kokkos toolkit compatible with CPU and GPU architectures
- computation times:
  - typically from a dozen of seconds (GPU version) to a minute (CPU version) to generate a virtual geometry sample at the mesoscale (~50um) embedding thousands of cells
  - typically from a dozen of minutes (GPU version) to an hour (CPU version) to perform the joint Monte-Carlo simulation and diffusion NMR signal attenuation computation
- code available from the Ginkgo GitLab repository:

  https://framagit.org/cpoupon/gkg/-/tree/master/simulation/src/plugin/gkg-simulation-plugin-functors/Medusa

Launching **MEDUSA simulation campaigns on supercomputers**:

- Available MEDUSA **Docker container with meta-json** configuration files
- Existing **syntax to link parameters** within and between populations

simulation-campaign.json

```
{
 "two_fiber_simulation_campaign": {
  "target_sample_count": 100000,
  "geometry" : {
    "field_of_view":[
     -10.0,
     -10.0,
     -10.0,
     10.0,
     10.0,
     10.0
    ],
    "grid_resolution":{
     "linked": {
       "type": "scaling_function",
       "parameters": {
       "coefficient": 2.0,
       "linked_paths": [["two_fiber_simulation_campaign",
                 "geometry",
                 "populations",
                 "fiber_population_01",
                 "parameters",
                 "axon",
                 "diameter_distribution",
                 "parameters",
                 "mean"]]
      }
     }
    },
    "apply_remove_overlaps" : 1,
    "apply_add_details" : 1,
    "apply_atom_regularization" : 1,
    "sphere_atom_overlap_solver_maximum_iteration_count": 250,
    "maximum_force_norm": 0.25,
    "maximum_atom_count": 10000000,
    "repulsion_force_moving_average_window_size" : 10,
    "minimum_repulsion_force_attenuation_percentage" : 10.0,
    "repulsion_force_stddev_percentage_threshold" : 1.0,
```

```
    "populations": {
     "fiber_population_01": {
      "type": "fiber-population",
      "parameters": {
       "sphere_atom_oversampling_ratio": 10.0,
       "axon": {
        "diameter_distribution": {
         "type": "gamma-distribution",
         "parameters": {
          "mean": {
           "random": {
            "lower_value": 0.4,
            "upper_value": 0.4
           }
          },
          "stddev": {
           "linked": {
            "type": "scaling_function",
            "parameters": {
             "coefficient": 0.1,
             "linked_paths": [[
              "two_fiber_simulation_campaign",
              "geometry",
              "populations",
              "fiber_population_01",
              "parameters",
              "axon",
              "diameter_distribution",
              "parameters",
              "mean"
             ]]
            }
           }
          },
          "multiplicator": 1e3,
          "upper_boundary": 10.0
         }
        },
        "global_angular_dispersion_in_degrees": {
         "random": {
          "lower_value": 0.0,
          "upper_value": 10.0
         }
        },
```

```
        "has_tortuosity": 1,
        "tortuosity": {
         "magnitude": 0.05,
         "angular_dispersion_in_degrees": {
          "random": {
           "lower_value": 0.0,
           "upper_value": 10.0
          }
         },
         "wave_length": 25.0
        },
        "has_beading": 0,
        "beading": {
         "interbeading_length_distribution": {
          "type": "gamma-distribution",
          "parameters": {
           "mean": 12.0,
           "stddev": 4.0
          }
         },
         "width_distribution": {
          "type": "gamma-distribution",
          "parameters": {
           "mean": 3.0,
           "stddev": 0.8
          }
         },
         "magnitude_ratio_distribution": {
          "type": "gamma-distribution",
          "parameters": {
           "mean": {
            "random": {
             "lower_value": 1.0,
             "upper_value": 1.9
            }
           },
           "stddev": 0.001
          }
         }
        },
        "mean_orientation": [
         0.0,
         0.0,
         1.0
        ],
```

```
...
     "fiber_population_02": {
      "type": "fiber-population",
      "parameters": {
       "sphere_atom_oversampling_ratio": 10.0,
       "axon": {
        "diameter_distribution": {
         "type": "gamma-distribution",
         "parameters": {
          "mean": {
           "random": {
            "lower_value": 0.3,
            "upper_value": 0.3
           }
          },
...
          "mean_orientation": {
           "linked": {
            "type": "random_orientation_in_plane",
            "parameters": {
             "linked_paths": [
              [
               "two_fiber_simulation_campaign",
               "geometry",
               "populations",
               "fiber_population_01",
               "parameters",
               "axon",
               "mean_orientation"
              ]
             ]
            }
           }
          },
...
         }
        }
       }
      }
     }
    }
}
```

Introduction: why building realistic digital white matter phantoms ?

Existing digital white matter phantom generators

The MEDUSA approach: combining realism and computational efficiency

Large scale simulations with the MEDUSA simulator

Conclusion

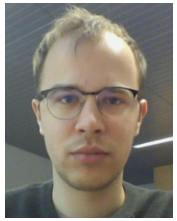
- MEDUSA is an all-in-one virtual geometry sample / diffusion Monte-Carlo / diffusion NMR signal attenuation simulator but any simulator can be used independently
- Relies on an sphere atom (and beyond sphere) decomposition of cells drastically speeding-up the generation of virtual samples and the Monte-Carlo simulation
- Highly parallelized for CPU or GPU supercomputers
- Clean software C++ design to allow further improvements (even more realistic cell models, alternative atoms, …)
- Large scale simulations only requiring the definition of meta-jsons
- Web-service under development in the frame of the EBRAINS infrastructure

Acknowledgements

- To the organizers of the BrainComp 2022 Workshop

- To the scientists and PhD students who contributed to the MEDUSA software:
  - from the BAOBAB/Ginkgo team (NeuroSpin, Paris-Saclay University, CNRS, CEA, France)
    - Cyril Poupon
    - Anas Bachiri
    - Alexis Brullé
    - Kevin Ginsburger (alumni)



  - from the INM1/Fiber Architecture Group (Forschungszentrum Jülich, Wuppertal University, Germany)
    - Markus Axer
    - Felix Matuschke


- This project/research has received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreement No. 945539 (Human Brain Project SGA3) and from the AIDAS CEA-FZJ Joint Institute.