Graichen, Uwe; Eichardt, Roland; Haueisen, Jens:

SpharaPy: a Python toolbox for spatial harmonic analysis of non-uniformly sampled data

Original published in:	SoftwareX. - Amsterdam [u.a.] : Elsevier. - 10 (2019), art. 100289, 7 pp.
Original published:	2019-08-09
ISSN:	2352-7110
DOI:	10.1016/j.softx.2019.100289
[*Visited:*	2019-11-13]

SoftwareX 10 (2019) 100289

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original Software Publication

SpharaPy: A Python toolbox for spatial harmonic analysis of non-uniformly sampled data

Uwe Graichen [*], Roland Eichardt, Jens Haueisen

Technische Universität Ilmenau, Department of Computer Science and Automation, Institute of Biomedical Engineering and Informatics, POB 100565, 98684 Ilmenau, Germany

ARTICLE INFO

Article history:
Received 2 May 2019
Received in revised form 18 July 2019
Accepted 19 July 2019

Keywords:
Spatial harmonic analysis
Eigenanalysis of the discrete Laplace–Beltrami operator
Generalized spatial Fourier analysis
Spatial filtering
Online filtering
EEG Data Processing

ABSTRACT

SpharaPy is a Python implementation of a new approach for spatial harmonic analysis (SPHARA). SPHARA extends the classical spatial Fourier analysis to non-uniformly positioned samples on arbitrary surfaces in $\mathbb{R}^3$. The basis functions (BF) used by SPHARA are determined by the eigenanalysis of the discrete Laplace–Beltrami operator, which is defined on a triangular mesh specified by the spatial sampling points. The SpharaPy Python toolbox provides classes and functions to compute the SPHARA BF for data analysis and synthesis as well as classes to design and apply spatial filters. An illustrative example of applying the SpharaPy package in the field of biosignal processing using electroencephalography data is presented.

Code metadata

Current code version	1.0.12
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2019_164
Code Ocean compute capsule	https://codeocean.com/capsule/2590457/tree
Legal Code License	BSD 3-Clause "New" or "Revised" license (BSD-3-Clause)
Code versioning system used	git
Software code languages, tools, and services used	Python (>= 3.6)
Compilation requirements, operating environments & dependencies	numpy (>= 1.16.1), scipy (>= 1.2.0), matplotlib (>= 3.0.2)
If available Link to developer documentation/manual	https://spharapy.readthedocs.io
Support email for questions	uwe.graichen@tu-ilmenau.de

1. Motivation and significance

Discrete Fourier analysis is very common in digital signal and image processing, and it is a fundamental tool in many applications. In common digital image data, the pixels are arranged on a flat surface in a Cartesian or rectangular grid. For such data, the basis functions (BF) for the Fourier transformation are usually implicitly specified by the transformation rule (cf. [1]).

In many applications, the sensors for data acquisition are not located on a flat surface and cannot be represented by Cartesian or regular grids. However, a spatial Fourier analysis for this kind of data is useful. An example from the field of biomedical engineering, where sensors are non-regularly arranged on a curved surface, is electroencephalography (EEG). In EEG, the sensors are placed at known (predefined and/or tracked) positions on the surface of the head. The positions of the sensors in these systems can be described by means of triangular meshes. Because of the non-regular sensor arrangement, standard 2D Fourier analysis cannot be used for the spatial analysis of these multisensor data.

In this article, we present a Python implementation of SPHARA, a new method for SPatial HARmonic Analysis of multisensor data [2]. SPHARA can be considered as a generalization of the spatial Fourier analysis for spatially arbitrary sensor arrangements. The SPHARA BF are computed as the eigenvectors of the Laplace–Beltrami operator, which is defined on the meshed surface of the sensor positions. Using this approach, BF

[*] Corresponding author.
E-mail address: uwe.graichen@tu-ilmenau.de (U. Graichen).

https://doi.org/10.1016/j.softx.2019.100289

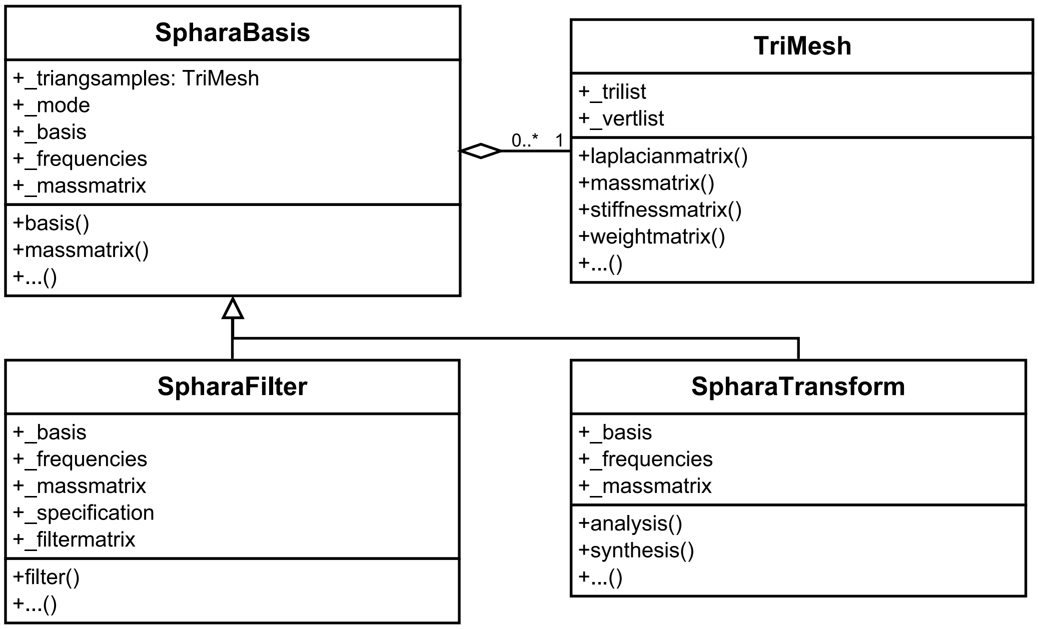

Fig. 1. The simplified class diagram of the SpharaPy package: The class `TriMesh` describes the spatial configuration of the sample points and is an aggregation of the `SpharaBasis` class. `SpharaBasis` is used to determine the BF. `SpharaTransform` and `SpharaFilter` are subclasses of `SpharaBasis`. The `@property` and the `setter` of the attributes are not shown in this class diagram.

of spatial harmonics for arbitrary arrangements of sensors can be generated. The recorded multisensor data are decomposed by projections into the space of these SPHARA BF.

The presented Python-Toolbox is particularly suitable for the spatial harmonic analysis of data measured with irregularly arranged sensors. Eigensystems of Laplace operators and Laplace–Beltrami operators are also applied in graph theory [3] as well as computer graphics and shape analysis [4–9].

2. Software description

2.1. Software architecture

SpharaPy is an object-oriented implementation of the SPHARA approach in Python. The SpharaPy package consists of five modules `trimesh`, `spharabasis`, `spharatransform`, `spharafilter` and `datasets`. The simplified class diagram of the classes implemented in the SpharaPy package is shown in Fig. 1.

The `trimesh` module provides the `TriMesh` class, which is used to specify the configuration of the spatial sample points. The `TriMesh` class is an aggregation of the `SpharaBasis` class. The class `SpharaBasis` is implemented in the module `spharabasis` and can be employed to determine SPHARA BF for spatially irregularly sampled functions with topology described by an instance of the `TriMesh` class. `SpharaBasis` is the base class, with two derived subclasses, `SpharaTransform` and `SpharaFilter` (see Fig. 1). The module `spharatransform` provides the class `SpharaTransform` to perform the SPHARA transformation, facilitating the SPHARA analysis and synthesis of spatially irregularly sampled data. The module `spharafilter` provides the class `SpharaFilter`, which can be employed for spatial filtering using a SPHARA basis. In addition, it provides methods for designing and applying different types of filters to spatially irregularly sampled data. The `datasets` module is an interface for the example data sets, which are included in the SpharaPy package.

To avoid unnecessary use of memory and computing power, a lazy approach is used to calculate the data attributes of class instances, such as BF or filter matrices. The attributes are not calculated until they are required for the first time. They are then saved and used as long as the underlying attributes do not change.

2.2. Software functionalities

The SpharaPy package provides essentially four classes to specify triangle meshes, to calculate spatial harmonic BF, for data analysis, and for spatial filtering.

The class `TriMesh` contains routines to define triangle meshes describing an arbitrary spatial multisensor arrangement. It also provides methods for determining matrices derived from the triangular mesh that are required to compute the SPHARA BF, such as weight, mass, stiffness, and Laplacian matrices (cf. [2,3,10,11]).

Using the class `SpharaBasis`, a generalized spatial Fourier basis (SPHARA basis) is computed by eigenanalysis of the discrete Laplace–Beltrami operator, which is defined on the triangular mesh. For discretizations using the inverse Euclidean or the unit weighting scheme, the following Laplace eigenvalue problem has to be solved

$$L\vec{x}_i = \lambda_i \vec{x}_i \,, \tag{1}$$

with the Laplacian matrix L (comprises the discretization of the Laplace–Beltrami operator), the eigenvalues λ_i, and the eigenvectors $\vec{x}_i$. If the FEM discretization is used the Laplacian matrix L is computed by $L = B^{-1} S$, with mass matrix B and stiffness matrix S. Inversion of the mass matrix B can be avoided by solving the generalized eigenvalue problem

$$S\vec{x}_i = \lambda_i B\vec{x}_i \,. \tag{2}$$

Various discretization methods are provided by the class `SpharaBasis` to determine the Laplace–Beltrami operator, the FEM approach, the inverse Euclidean, and unit weighting of the edges in the triangular grid [2,3,10,12]. If the Laplace–Beltrami operator is discretized with unit weighting of the edges of the triangular grid, then the coordinates of the positions of the vertices are not considered, but adjacencies, defined by edges in the triangle mesh. For inhomogeneous triangular meshes, the weighting function of the edges has to be adapted according to the mesh geometry. A common choice is to use the inverse of the Euclidean distance [4]. Another approach to adapt the discretization of the Laplace–Beltrami operator to inhomogeneous

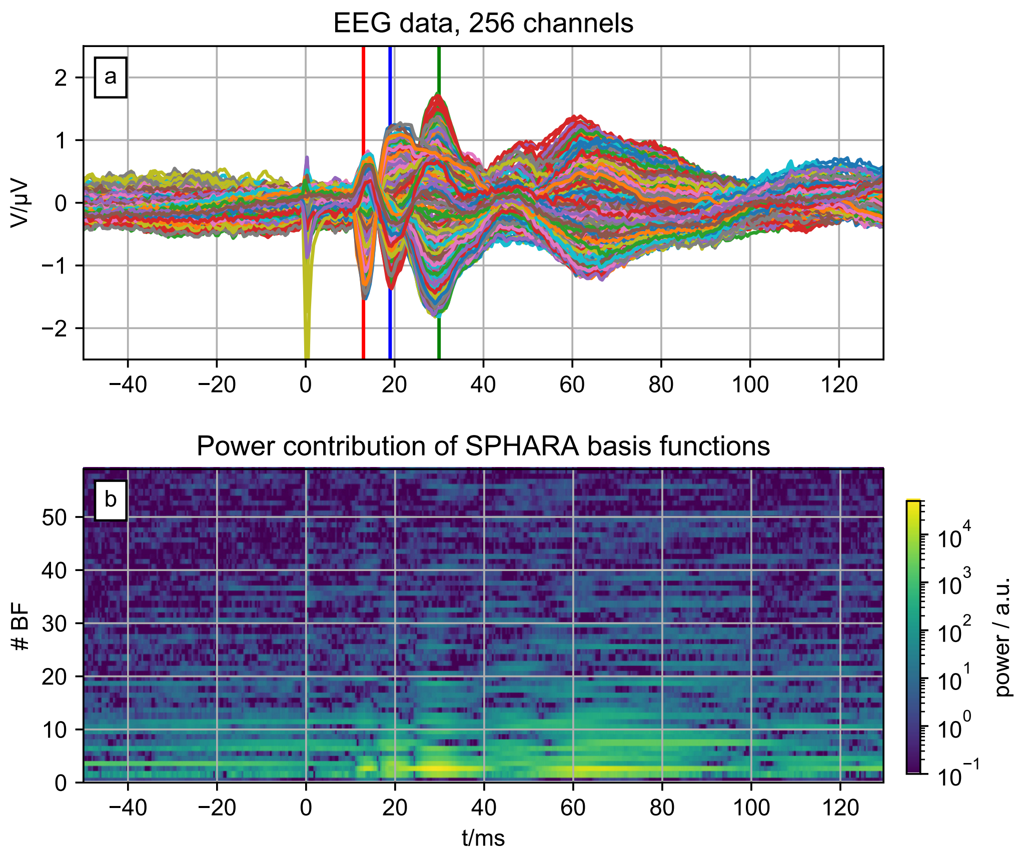

Fig. 2. A butterfly plot of all channels of the EEG data (a) is compared to the visualization of the power contributions of the first 60 low-frequency SPHARA BF (b). Each EEG channel in the butterfly plot is represented in a different color, the components P14, N20 and N30 are marked by red, blue and green vertical lines respectively (a). Only the first 60 out of 256 BF are used for the visualization, as the low-frequency BF carry most of the signal power (b).. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

spatial sampling is the FEM approach. In this case, the Laplace–Beltrami operator is generated using a mass matrix and a stiffness matrix determined from the properties of the triangular mesh [2,10,13]. The eigenvalues λ_i can be considered as discrete spatial frequencies. Using the inverse Euclidean and unit weighting the corresponding eigenvectors $\vec{x}_i$ form a harmonic orthonormal basis regarding the scalar product of the vector spaces. The eigenvectors $\vec{x}_i$ computed by the FEM approach are orthogonal regarding the B-relative scalar product, cf. Eq. (4). Further details on the properties of the discrete Laplace–Beltrami operators and the resulting SPHARA bases are given in Appendix, cf. also [2,14].

The class `SpharaTransform` provides methods for the analysis (forward transformation) of discrete data defined on the vertices of the triangular mesh and the synthesis (inverse transform) of SPHARA coefficients. If the inverse Euclidean or unit weighting are used for discretization, the inner product for vector spaces is employed for the transformation from spatial into spatial frequency domain (analysis). The transformation from spatial into spatial frequency domain is computed by

$$\vec{c}^{\,\top} = \vec{f}^{\,\top} X \,, \tag{3}$$

where the columns of the matrix X contain the SPHARA BF (eigenvectors $\vec{x}_i$ in Eqs. (1) and (2)), $\vec{f}$ represents the data sampled in the spatial domain, and $\vec{c}$ are the SPHARA coefficients, the representation in the domain of spatial frequencies. For an analysis using eigenvectors computed by the FEM approach, the inner product incorporating the mass matrix B that assures the B-orthogonality needs to be applied [2,13]. The transformation from the spatial into the spatial frequency domain is then computed by

$$\vec{c}^{\,\top} = \vec{f}^{\,\top} B X \,, \tag{4}$$

with the mass matrix B. Discrete data $\vec{f}$ in the spatial domain are synthesized using the linear combination of the SPHARA coefficients $\vec{c}$ and the corresponding BF X

$$\vec{f}^{\,\top} = \vec{c}^{\,\top} X^{\top} \,. \tag{5}$$

The class `SpharaFilter` can be used to design different types of filters and to apply these filters to spatially irregularly sampled data. A filter matrix F can be determined by

$$F = R \cdot X \cdot (R \cdot X)^{\top} \,. \tag{6}$$

The matrix X contains the SPHARA BF, and the matrix R is a selection matrix, which contains a 1 on the main diagonal if the corresponding SPHARA BF from X is chosen. All other elements of this matrix are 0. If the Laplace–Beltrami operator with FEM discretization is used to calculate the SPHARA BF, the mass matrix B must be included in the equation to compute the filter matrix

$$F_{\mathrm{FEM}} = B \cdot R \cdot X \cdot (R \cdot X)^{\top} \,. \tag{7}$$

The spatial SPHARA filter is applied by multiplying the data matrix D with the filter matrix F

$$\tilde{D} = D \cdot F \,. \tag{8}$$

In the data matrix D, the rows represent the time samples and the columns the spatial samples (sensor positions). The matrix $\tilde{D}$ represents the filtered data.

As for filters in the temporal Fourier domain, an inappropriate filter design in the spatial Fourier domain can cause artifacts and disturbances such as Gibbs and Ringing phenomena. These disturbances can be reduced by choosing appropriate filter coefficients. The API of the SpharaPy package offers parameters to design adapted filters.

The computational complexity of the eigenanalysis of the discrete Laplace–Beltrami operator for determining the BF is $O(n^3)$, where n is the number of spatial sample points. The SPHARA transformation and SPHARA filtering are performed by the multiplication of a vector containing the data and a matrix, where the columns are the SPHARA BF. The complexity of the transformation and the filtering is $O(n^2)$. The amount of memory required to store the BF is $O(n^2)$.

3. Illustrative examples

Our example demonstrates the application of the SpharaPy package for the processing of EEG signals. We show three features of the SpharaPy package: computation of the SPHARA basis for the sensor setup of a waveguard 256-channel EEG system (ANT Neuro BV, Enschede, The Netherlands), SPHARA analysis of EEG data, and spatial low-pass filtering using a SPHARA-based filter. The data used in this example originate from a previously performed EEG experiment (see [2]) addressing the cortical activation related to somatosensory-evoked potentials (SEP). The median nerve of the right forearm of a volunteer was stimulated by bipolar electrodes. Data were sampled at 2048 Hz and software high-pass (24 dB/oct, cutoff-frequency 2 Hz) and notch (50 Hz and two harmonics) filtered. All recorded trials were manually checked for artifacts, and the remaining trials were averaged. The data include 256 channels and a time interval of 50 ms before to 130 ms after stimulation (369 time samples). A butterfly plot of all channels of the averaged EEG data is shown in Fig. 2(a). The components P14, N20 and N30, which are particularly important for the medical assessment of the SEP data, were selected manually for the volunteer [15,16]. In Fig. 2(a), these components are marked by red, blue and green vertical lines. The study was approved by the Ethics Committee of the Unitexversity Hospital Jena, and written informed consent was obtained.

In this article, only parts of the source code are presented. The complete source code of this example, which can also be used to generate the images of the manuscript and furthermore for the SPHARA analysis of the EEG data, is included in the SpharaPy package (Python and IPython notebook format, files `SpharaPy_example.[py|ipynb]`).

First, we import the required SpharaPy and Python modules for numerical calculations and visualization.

```
# import SpharaPy modules
import spharapy.trimesh as tm
import spharapy.spharatransform as st
import spharapy.spharafilter as sf
import spharapy.datasets as sd

# import further Python modules and functions
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
```

In the next step, the dataset of the EEG experiment is loaded. This dataset is included in the SpharaPy toolbox and contains vertex lists, triangle lists, and the averaged EEG data.

```
data_in = sd.load_eeg_256_channel_study()

vertlist = np.array(data_in['vertlist'])
trilist = np.array(data_in['trilist'])
eegdata = np.array(data_in['eegdata'])
```

Subsequently, we create an instance of the class `TriMesh` from the list of vertices and triangles. In this instance, we specify the spatial arrangement of the sensor positions.

```
mesh_eeg = tm.TriMesh(trilist, vertlist)
```

Then, we create an instance of the class `SpharaTransform`, which is used for the spatial SPHARA analysis of the EEG data. The SPHARA basis is determined using an FEM discretization of the Laplace–Beltrami operator. The power contribution of the 60 low-frequency SPHARA BF to the EEG data is illustrated in Fig. 2(b).

```
sphara_transform_fem = st.SpharaTransform(mesh_eeg
    , 'fem')
sphara_trans_eegdata =
    sphara_transform_fem.analysis(eegdata.
        transpose())
```

We determine an instance of the `spharapy.SpharaFilter`, which is used to design and apply the spatial filter. The SPHARA basis in this example is determined by means of an FEM discretized Laplace–Beltrami operator. We implement a spatial low-pass filter using the twenty lowest frequency SPHARA BF. Nine of the twenty lowest frequency BF used for spatial filtering are shown in Fig. 3.

```
sphara_filter_fem = sf.SpharaFilter(mesh_eeg,
    mode='fem', specification=20)
# extract the BF used in the filter design,
# for visualization
basis_functions_fem, natural_frequencies_fem =
    sphara_filter_fem.basis()
```

We apply this filter to the EEG data. The result of the filtering in comparison to the unfiltered data is shown in the first row of Fig. 4.

```
sphara_filt_eegdata =
sphara_filter_fem.filter(eegdata.transpose()).
    transpose()
```

In the last step of our example, the effectiveness of the spatial SPHARA-based filter is demonstrated. We increase the noise in the EEG data by adding artificially created white noise at different signal-to-noise ratios (3dB, 0dB, and -3dB). The spatial low-pass SPHARA filter employing 20 BF is applied to these data.

```
# vector with SNR in dB
db_val_vec = [3, 0, -3]

# compute the power of the SEP data
power_sep = np.sum(np.square(np.absolute(eegdata)))
    / eegdata.size

# vector with standard deviations of the noise
# relative to signal power for given SNR
noise_sd_vec = list(map(lambda db_val:
np.sqrt(power_sep / (10 ** (db_val / 10))),
    db_val_vec))

# add the noise to the EEG data
eegdata_noise = list(map(lambda noise_sd:
eegdata + np.random.normal(0, noise_sd, [256, 369]),
    noise_sd_vec))

# apply the spatial SPHARA lowpass filter the
# EEG data containing the artificial noise
```

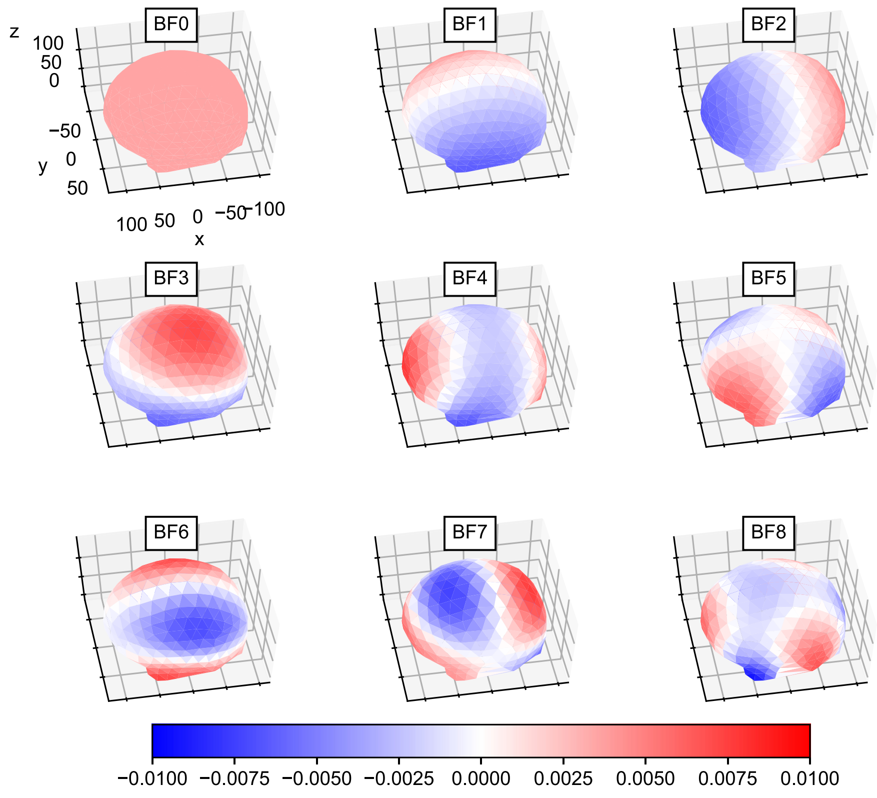

Fig. 3. The nine low-frequency SPHARA BF for the 256-channel EEG system; the spatial DC component subfigure is indicated with BF0. The individual BF are plotted on the triangular grid, which specifies the EEG sensor setup.

```
eegdata_noise_filt = list(map(lambda eeg_noise:
(sphara_filter_fem.filter(eeg_noise.transpose()).
    transpose()), eegdata_noise))
```

The results of the SPHARA filtering are shown in the Figs. 4 and 5. In Fig. 4 the EEG time series are plotted. The left column contains the unfiltered and the right column the filtered time series. Three time points (P14, N20 and N30) are marked by vertical lines (red, blue and green) in the individual plots. For these points in time, the distributions of the electrical potential on the scalp are shown in Fig. 5. In correspondence to Figs. 4 and 5 shows unfiltered data in the left three columns and SPHARA filtered data in the right three columns.

4. Impact

Using the presented SpharaPy Python package, it is possible to determine customized spatial Fourier bases for multi-sensor systems with spatially arbitrarily arranged sensors. The BF can be explicitly determined for the spatial domain of the measured data. For this reason, the mapping of the spatial domain of the data to a domain (e.g., rectangular plane with Cartesian grid or unit spherical surface), for which a Fourier basis can be implicitly defined in the transformation rule, can be omitted. As the BF are adapted to the spatial domain of the data, window functions for handling the boundaries of the domain can also be omitted, which is another advantage of SPHARA.

These properties result in a number of potential applications of the SPHARA transformation, such as spatial analysis and filtering as well as dimension reduction and data compression.

In many other multivariate data decomposition methods, such as PCA, ICA, and PARAFAC, recorded data sets are used to generate the components for spatial data decomposition. In SPHARA, however, the BF are only determined from topological information about the sensor setup and positions. Thus, SPHARA BF can be computed before data recording.

SPHARA-based spatial filters are applied separately on the data of individual time samples. For this reason, the phase properties of recorded time series are not affected, and phase analysis methods can be applied to noise-reduced data sets.

The SPHARA BF can be efficiently computed, and the two introduced SAHARA-based applications, transformation and filtering, can also be applied very rapidly. This high processing speed enables the online application of SPHARA-based methods.

5. Conclusions

In this paper we have introduced SpharaPy, a Python implementation of SPHARA, which is a new method for spatial harmonic analysis of multisensor data. SPHARA can be considered as a generalization of the discrete spatial Fourier transform. We have discussed some theoretical basics of SPHARA in the paper. In an application example of the new software toolbox from the field of biosignal processing the calculation of the SPHARA basis, the design of a spatial low-pass filter and the filtering of multichannel data were shown.

Recently, spatially distributed multisensor networks have become increasingly relevant. This opens up further application areas for SPHARA-based methods. Examples from the fields of meteorology and geosciences include multi-sensor systems for monitoring weather, glacier, earthquake, and volcanic activities as well as technical applications, such as traffic control and vibration monitoring.

Declaration of competing interest

The authors wish to confirm that there are no known conflicts of interest associated with this publication and there has been

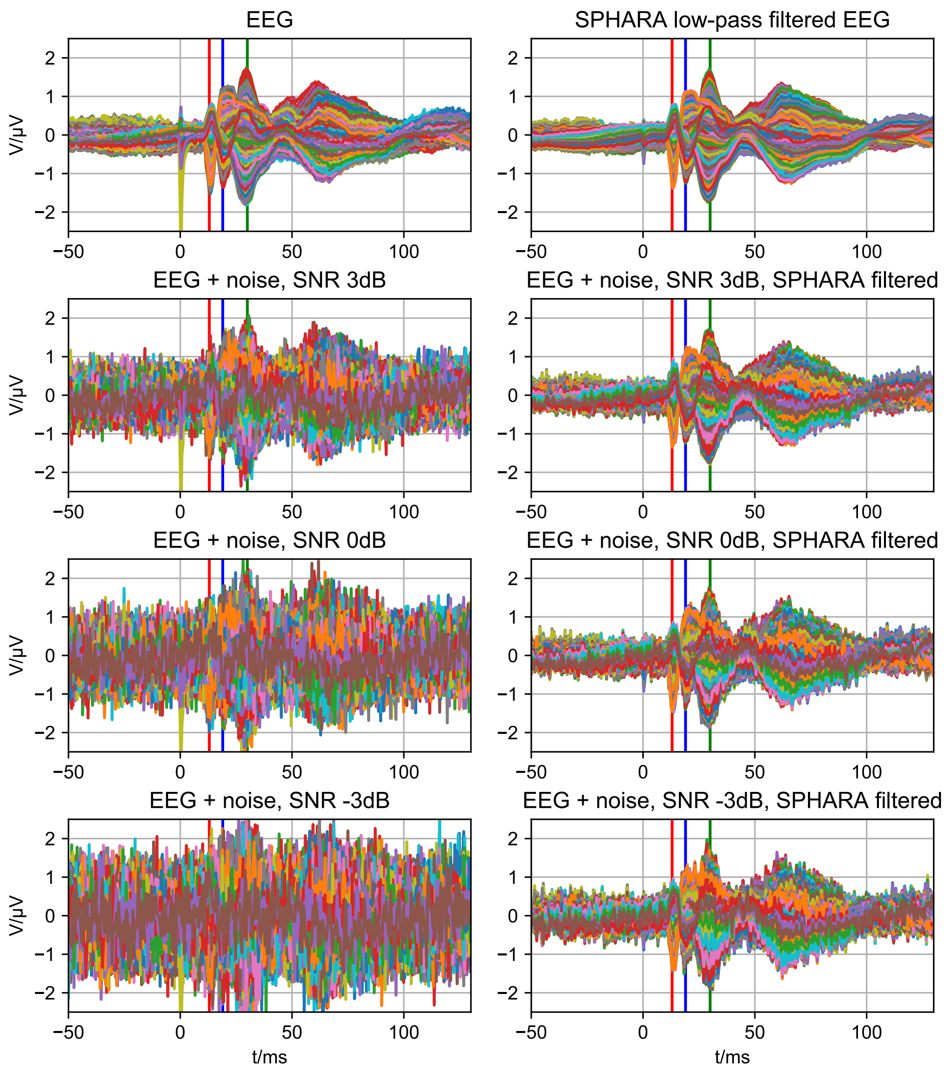

Fig. 4. Application of the SPHARA-based spatial filter to EEG data, represented in the time domain; left column shows unfiltered data, while the right column shows SPHARA low-pass-filtered data. Each EEG channel is indicated in a different color. In the first row, the data contain no additional artificial noise. In the second to fourth rows, the data were disturbed with spatially non-correlated artificial Gaussian noise with signal-to-noise ratios 3, 0, and −3 dB. In all subfigures the components P14, N20 and N30 are marked by red, blue and green vertical lines, respectively. For these selected time samples, the topographical maps are shown in Fig. 5. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

no significant financial support for this work that could have influenced its outcome.

Acknowledgments

This project has received funding from the German Research Foundation under grand number Gr 5251/1-1, from the European Union Horizon 2020 research and innovation programme under grant agreement No 686865 (project BREAKBEN), and from Marie Skłodowska-Curie, Germany Innovative Training Network under grant agreement No 813483 (project INFANS). We acknowledge support for the Article Processing Charge from the German Research Foundation and the Open Access Publication Fund of the Technische Universität Ilmenau.

Appendix. Theoretical aspects

The properties of the SPHARA basis are essentially determined by the method used to discretize the Laplace–Beltrami operator. The eigenvectors $\vec{x}_i$ and eigenvalues λ_i are computed according to Eqs. (1) and (2). Advantageous properties of the Laplace matrix L, containing the discrete Laplace–Beltrami operator, are symmetry, positive weights, positive semi-definiteness, locality and convergence, cf. [2,14].

The symmetry $L_{ij} = L_{ji}$ leads to real eigenvalues and orthogonal eigenvectors regarding the scalar product of vector spaces. Positive edge weights for the determination of the discrete Laplace–Beltrami operator and matrix symmetry, lead to the positive semi-definiteness of L. The locality property enables to determine the Laplace–Beltrami operator from a vertex and its neighboring vertices, adjacent via an edge of the triangular mesh. The convergence property signifies the convergence from the discrete to the continuous Laplace–Beltrami operator for a sufficient refinement of the mesh.

The Laplacian matrices L discretized by means of inverse Euclidean or unit weighting are positive semi-definite, symmetric and use positive weights. Due to these properties, the eigenvalues for these two discretizations are non-negative $\lambda_i \in \mathbb{R}$ with $\lambda_i \geq 0$ and can be considered as discrete spatial frequencies. The eigenvectors $\vec{x}_i$ are real-valued and form an orthonormal basis regarding the scalar product of vector spaces.

The FEM approach does not fulfill the positive weight property, if the mesh contains triangles with interior angles in the

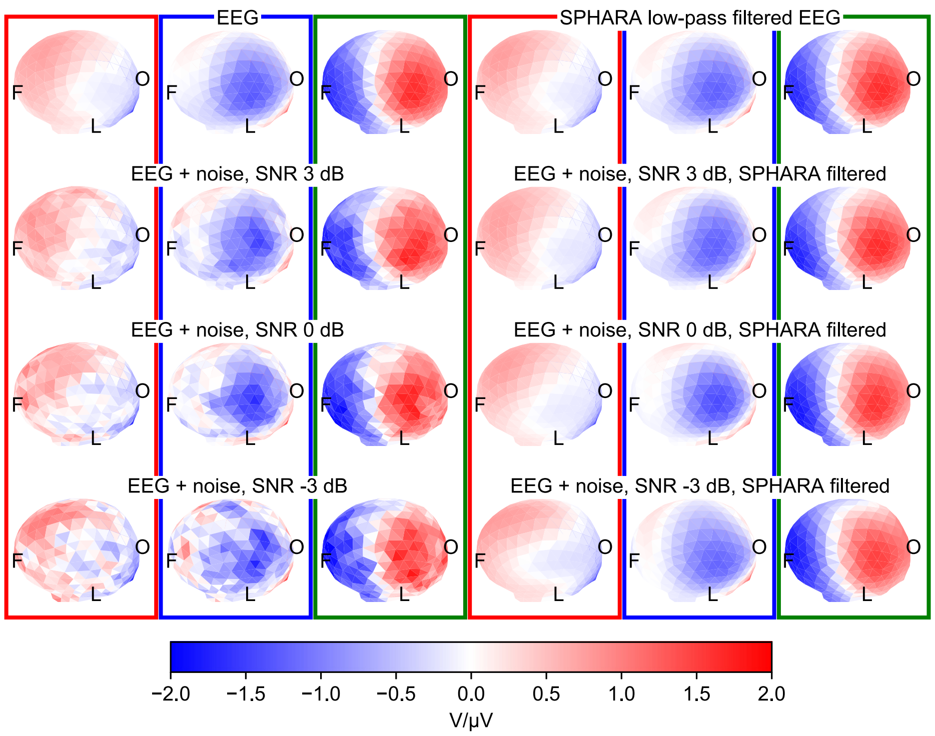

Fig. 5. Application of the SPHARA-based spatial filter to EEG data, represented in the spatial domain; the potential distributions on the scalp are shown from the top left. The frontal (F), occipital (O) and left (L) positions of the head are indicated. The three left columns show unfiltered data, while the three right columns show SPHARA low-pass-filtered data. In the first row, the data contain no additional artificial noise. In the second to fourth rows, the data were disturbed with spatially non-correlated artificial Gaussian noise with signal-to-noise ratios 3, 0, and -3 dB. Columns that correspond to components P14, N20 and N30 are marked by red, blue and green frames respectively, cf. also Fig. 4. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

interval $(\pi/2, \pi)$, for which the cotangent is negative. For the FEM formulation, the basis is computed by solving the generalized symmetric definite eigenproblem, cf. Eq. (2). Thus, the inversion of the mass matrix B is avoided. Because $B^{-1}S$ is not symmetric, the eigenvectors $\vec{x}_i$ are real-valued, but not orthonormal with respect to the scalar product for vector spaces. To use these eigenvectors as basis, the inner product defined in Eq. (4) has to be used, which assures the B-orthogonality. For more details see also [2,13].

If the triangular mesh representing the spatial multisensor arrangement possesses a boundary, several boundary conditions (BC) can be applied to compute the basis functions by using the Eqs. (1) or (2). Assuming that the values measured on the boundary are not 0 or constant, the Neumann BC is more appropriate. But also other BC such as Dirichlet are feasible. The basis functions are determined adaptively for the domain specified by the triangulation of the sensor positions. Thus, a special processing of the boundary of the domain, e.g. by means of a window function, is not required.

References

[1] Rao KR, Kim DN, Hwang JJ. Fast Fourier transform: algorithms and applications. Signals and communication technology, Springer; 2010.
[2] Graichen U, Eichardt R, Fiedler P, Strohmeier D, Zanow F, Haueisen J. SPHARA - A generalized spatial Fourier analysis for multi-sensor systems with non-uniformly arranged sensors: Application to EEG. PLoS One 2015;10:1–22.
[3] Chung FRK. Spectral graph theory, vol. 92. CBMS regional conference series in mathematics, American Mathematical Society; 1997.
[4] Taubin G. Signal processing approach to fair surface design. In: Proceedings of the ACM SIGGRAPH conference on computer graphics; 1995. p. 351–8.
[5] Vallet B, Levy B. Spectral geometry processing with manifold harmonics. Comput Graph Forum 2008;27(2):251–60.
[6] Lévy B. Laplace-Beltrami eigenfunctions towards an algorithm that "understands" geometry. In: IEEE international conference on shape modeling and applications 2006; 2006, p. 1–8.
[7] Shi Y, Lai R, Wang DJJ, Pelletier D, Mohr D, Sicotte N, Toga AW. Metric optimization for surface analysis in the Laplace-Beltrami embedding space. IEEE Trans Med Imaging 2014;33(7):1447–63.
[8] Zhu F, Li S, Wang G. Example-based materials in Laplace-Beltrami shape space. Comput Graph Forum 2015;34(1):36–46.
[9] Melzi S, Rodolà E, Castellani U, Bronstein MM. Localized manifold harmonics for spectral shape analysis. Comput Graph Forum 2018;37(6):20–34.
[10] Zhang H, van Kaick O, Dyer R. Spectral methods for mesh processing and analysis. In: Schmalstieg D, Bittner J, editors. STAR proceedings of eurographics, vol. 92(RC-20404). 2007, p. 1–22.
[11] Fujiwara K. Eigenvalues of Laplacians on a closed Riemannian manifold and its nets. Proc Amer Math Soc 1995;123(8):2585–94.
[12] Meyer M, Desbrun M, Schröder P, Barr A. Discrete differential geometry operators for triangulated 2-manifolds. In: Hege HC, Polthier K, editors. Visualization and mathematics III. Springer; 2003, p. 35–57.
[13] Vallet B, Levy B. Spectral geometry processing with manifold harmonics. Technical report inria-00186931, Université Nancy, Institut National Polytechnique de Lorraine; 2007.
[14] Wardetzky M, Mathur S, Kälberer F, Grinspun E. Discrete laplace operators: No free lunch. In: Belyaev A, Garland M, editors. SGP07: Eurographics Symposium on Geometry Processing. Eurographics Association; 2007, p. 33–7.
[15] Mauguiere F, Allison T, Babiloni C, Buchner H, Eisen AA, Goodin DS, Jones SJ, Kakigi R, Matsuoka S, Nuwer MR, Rossini PM, Shibasaki H. Somatosensory evoked potentials. In: Deuschl G, Eisen A, editors. Recommendations for the practice of clinical neurophysiology: guidelines of the international federation of clinical neurophysiology. Elsevier Science B.V.; 1999, p. 79–90.
[16] Cruccu G, Aminoff MJ, Curio G, Guerit JM, Kakigi R, Mauguiere F, Rossini PM, Treede R-D, Garcia-Larrea L. Recommendations for the clinical use of somatosensory-evoked potentials. Clinical Neurophysiol 2008;119(8):1705–19.