title

Titel

Subtitle

Simon Heinke Julius Lerm Felix Griesau Marco Klamke
Lars Debor Petros Simidyan Blerta Hamzallari Sugandha Sachdeva

last Change: April 25, 2017

1 Introduction

Medicine today is highly advanced and is able to treat an immense amount of diseases and disorders. This results in high standards and a lot of pressure on people working in this branch. In order to further improve the standards those amount of errors has to be minimized, since they can have fatal consequences.
In pursuance of achieving, maintaining as well as improving these abilities, technological assistance is of utmost importance.

Since the brain is one of the main organs, caution and accuracy is essential while treating its conditions. Techniques such as the MEG (Magnetoencephalography) or EEG (Electroencephalography) observe the brains activity by measuring and monitoring magnetic fields or electrical deviations.
These procedures help diagnosing epilepsy, brain death and other severe brain diseases.
Furthermore they are used for research in fields like psychology. A proper visualization aids the usability of the generated data.

The MNE-CPP project builds tools for the purpose of making analyzing EEG and MEG data easier. Thus the whole project is open-source and everyone can contribute. As language only C++ is used, although the project simultaneously exists in other languages, e.g. Python. MNE-Scan, -Analyze and -Browse are some of the features of the existing framework.

The new extension of the current project and focuses on real-time 3D-visualization of EEG/MEG sensor date, while being as exact and fast as possible. It is integrated into the existing code and accessible through the MNE-Client as well as compatible with operating systems Linux and Windows.

Contents

2 Requirements

The product receives EEG/MEG sensor data and constructs a real-time 3D visualization of the brains current activity. Users can choose between further options, changing the output immediately to their personal preferences.

2.1 Mandatory criteria

The following functions have to be implemented correctly and must fulfill given requirements.

2.1.1 Surface constrained distance calculation (Geodesic problem on meshes)

Because the brain has an uneven surface, a function for calculating the distance between two separate points is needed.
As the euclidian distance would not respect the structure of the surface, a different approach for determining the exact distances has to be implemented. The function receives input data in form of a preprocessed triangulated surface mesh and calculates the distance between the vertices.

C111 Based on that data, the function calculates a matrix that holds values describing the distances between all vertices using double precision.

C112 The function must be able to process up to 200,000 vertices.

C113 The user can limit the calculation to a subset of vertices.

2.1.2 Point to plane mapping

Since the sensors do not directly touch the head, therefore float slightly above it, an accurate projection is needed to exactly localize their positions regarding the brain.
Hence a function must be implemented to solve this problem. The function receives a set of sensor locations in 3D-Space and maps them onto the underlying surface mesh. Thus every sensor gets assigned to a vertex of the mesh.

C121 The function must be able to handle data from MEG-sensors which have a known orientation.

C122 The function must be able to handle data from EEG-sensors which are non-orientated.

2.1.3 Interpolation algorithm

The algorithm receives a mesh and a subset of vertices

C131 Based on the said subset the algorithm must calculate the values for every vertex of the mesh.

C132 For this, the algorithm creates a matrix storing weights for the later interpolation. The interpolation process can be summarized by the following equation:

$y_{full} = W \cdot y_{sub}$, where W is the mentioned matrix and y_{sub} is the current dataset for the known sensors, i.e. vertices.

C133 The calculation of the weight matrix must be based on the result of the SCDC (2.1.1).

2.1.4 Integration in Disp3D

In order to ensure usability within the given framework MNE-CPP, the final visualization must be integrated into the preexisting GUI, namely Disp3D.

C141 A new function must be added to the Disp3D tree model. Internally this function must create a new handler.

2.2 Non-Functional Requirements

C211 The software has to run on the latest versions of 2 operating systems, namely Windows and Linux.

2.3 Optional criteria

Besides of the mandatory features there is some functionality that would enrich the software but are not set as main goals.

2.3.1 SCDC

C311 The computation time should not exceed 1 second.

2.3.2 Interpolation

C321 One interpolation cycle should take less than 17ms.

C322 Multiple methods for calculating the weight matrix can be implemented. The user can select one.

2.4 Differentiating criteria

C411 The program receives preprocessed data and does get in touch with hardware sensors.

C412 The program does not evaluate the data but processes the data for further visualisation.