Software-Project 2017

Functional Specification

Real-Time Mesh Utilities

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

last change: July 3, 2017

1 Introduction

Techniques such as the MEG (Magnetoencephalography) or EEG (Electroencephalography) observe the brains activity by measuring and monitoring magnetic fields or electrical deviations produced by groups of neurons.
These procedures help diagnosing migraine variants and other brain diseases. In addition, assistance in research and localization of epilepsy is another possible usage of EEG/MEG.
Furthermore they are used for research in fields like psychology. A proper visualization aids the usability of generated data and provides a superficial graphic overview of the brains activity, thus enabling first interpretations or even diagnosis.

The MNE-CPP project builds tools for the purpose of making the analysis of EEG and MEG data easier. Thus, the whole project is open-source and everyone can contribute. As programming 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 standalone features of the existing framework.

The new extension of the current project focuses on real-time 3D-visualization of EEG/MEG sensor data, while being as exact and fast as possible. It should both be integrated into the existing code and accessible through MNE Scan, which functions as the real-time acquisition and processing software in the MNE-CPP project. MNE-CPP and thus MNE Scan are cross-platform capable (Windows, Linux, Mac).

Contents

2 Requirements

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

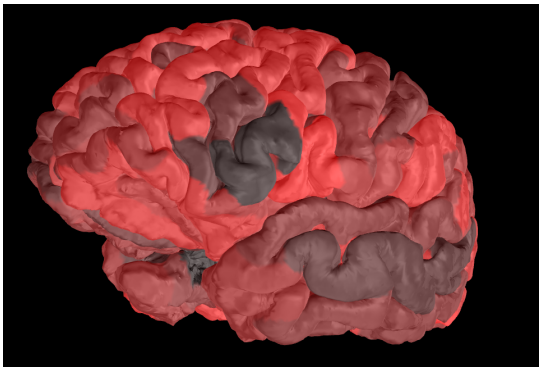

Figure 1: Interpolated brain activity

2.1 Mandatory Criteria

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

2.1.1 Surface Constrained Distance Calculation (SCDC)

Because the algorithm is supposed to work on folded surfaces, e.g. the brain, a function for calculating the distance between two 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 a given mesh, 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 Sensor-to-Mesh Mapping

Since the sensors do not directly touch the head and therefore float slightly above it, an accurate projection is needed to exactly localize their positions on the surface beneath (e.g. head/brain/MEG helmet).
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 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 main task is the ongoing interpolation, processing a particular set of sensor data, representing the brain's activity.
Because the number of vertices is bigger than the quantity of sensor points, most vertex-values must be interpolated. Thus the algorithm receives a mesh and a subset of vertices with their respective sensor data.

C131 Based on the said subset the algorithm must calculate the values of the neurophysiological activity 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 (**??**).

C134 Bad channels, a part of the given sensor meta data, must be considered during processing.

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 3D visualization library, namely Disp3D.

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

2.1.5 Non-Functional Requirements

C151 The software is cross-platform compatible.

C152 Features **??** and **??** are implemented in the class *GeometryInfo*, while **??** is facilitated in the class *Interpolation*.

C153 The function to integrate the product into Disp3D (**??**) is named *addSensorData()*.

C154 For introduction purposes, a product video is to be created and published on the MNE-CPP website.

C155 The software must be integratable into MNE Scan.

C156 Only Qt libraries and the Eigen framework are used during development.

2.2 Optional Criteria

As long as the mandatory requirements are fulfilled, it is desirable to further match the following criteria.

2.2.1 SCDC (??)

C211 The computation time should be as low as possible.

C212 A threshold for the maximum distance between two vertices is used for reduction of processing time.

2.2.2 Interpolation (??)

C221 One interpolation cycle should take less than 17ms.

C222 Multiple methods for calculating the weight matrix can be implemented. The user can select one.

C223 The computation is executed on GPU-level using compute shaders.

2.3 Delimiting Criteria

The following criteria limit the functionality of the system.

C311 The program receives preprocessed data and does not get in touch with hardware sensors.

C312 The program does not evaluate the data medically and solely processes the data for further visualization.

3 Product Usage

To describe what possible usages the software has, the aspects of scope, potential target audience and operating conditions will be outlined.

3.1 Scope

The application will be used for medical purposes especially in diagnostic scenarios. The MNE-CPP framework is already used to research on certain brain-functions and diseases (e.g. epilepsy). The real-time mesh utilities will be used by the front-end-applications already existing, mainly MNE Scan.
Furthermore, the software can be reused by other developers for additional enhancements or modifications.

3.2 Potential Target Audience

The main target group of the project consists of scientists and physicians who want to research and analyze the behavior of the human brain. E.g. MNE-CPP is already being used in the babyMEG lab at the Boston Children's Hospital to acquire and process MEG/EEG data of infants.

3.3 Operating Conditions

Because of the essential real-time-functionality the software has to run the SCDC (**??**) "offline" (i.e. not during the displaying process).
On the contrary, the interpolation has to be "online" (during the displaying process) due to the changing input and the live output.

4 Product Environment

4.1 Scenario

As mentioned before, the aim of the product is live visualization of MEG and EEG data. The software should reconstruct actual brain activity based on given sensor signals.
As mentioned before, the aim of the product is to visualize MEG and EEG data in real-time. The software should reconstruct actual brain activity based on given sensor signals.
The two possible sources of data are:

- The MEG, short for Magnetoencephalography, is a measurement of the neurophysiological activity of the brain. The changes of the magnetic field are recorded by sensors which are non-invasive (i.e. on the outside of the head). Besides research, the MEG is used for planning complex brain-surgeries.
- The EEG, short for Electroencephalography, is a measurement method for recording the electrical activity on the surface of the head. Electrodes that directly touch the skin measure potential fluctuations (brain waves) and amplify them. It is a standard method in neurology.

4.2 Software

4.2.1 The MNE-CPP Framework

The MNE-CPP framework provides several applications, dividable into front-end and back-end software. MNE Scan, MNE Browse and MNE Analyze are front-end applications which are designed for high usability. Beneath the front-end is the library layer which contains the core libraries provided by the framework.
The application must be integrated into the library layer.

4.2.2 System

The software must be usable across platforms in order to be accessible for more users. It requires OpenGL compatibility and depends on the Qt libraries (e.g. Qt3D).

4.3 Hardware

The program should run on most modern computers with enough performance to handle it.
A MEG-system (e.g. Elekta Neuromag® Vector View™) is connected to the computer and sends data which is handled by the software.

5 Product Functions

5.1 User Functions

F11 The user can access the implemented features as a part of MNE Scan (see section **??**).

F12 Either MEG or EEG data can be used as input.

F13 Any desired surface mesh and set of sensor data, selected by the user, can be used as input data for further calculations.

F14 A preferred subset of vertices can be selected, to perform distance calculations on.

F15 The user is able to select a threshold for identifying all relevant vertices while interpolating.

F16 As part of MNE Scan/Disp3D all graphical options are available to the user, meaning e.g. different coloring and rotation of the object.

5.2 Offline Functions

These functions are performed one time and are executed prior to the interpolation process.

F21 The distance between two vertices on a given mesh, is calculated following the structure of the surface. Therefore the euclidian distance is not used.

F22 All floating sensor points are projected onto the nearest vertices on the input mesh.

F23 A weight matrix, containing values for all vertices, is generated.

5.3 Real-Time Functions

These functions are performed continuously and in real-time.

F31 Based off prior calculations, the interpolation assigns every vertex a value.

6 Product Data

The following points depict the data used in the program.

D11 *Meshes:* A mesh consists of vertices which represent a 3D model, most likely that of a brain, scalp (head) or sensor helmet. The mesh is included in the input.

D12 *Distance matrix:* The matrix consists of values which describe the distances between the vertices. The distance matrix is calculated from the mesh.

D13 *Sensor array:* The sensor array contains the sensors positions which are represented by vectors. The sensor array is included in the input.

D14 *Measurement data:* The measurement data is represented as a matrix. Every row holds one signal curve of a sensor. Thus, every column defines a certain point of time.

D15 *Intensity vector:* The intensity vector is a set of values that specify the neurophysiological activity of the vertices.

7 Product Services

S11 *Real-time processing:* Once the software is initialized, it should process the input data in real-time.

S12 *Efficiency:* The program should work on normal hardware and be efficient enough to deliver its performance on most setups.

S13 *Sturdiness:* The software should be independent of the executing hardware.

S14 *GPU-Usage:* For maximum efficiency some operations should use the GPU instead of the CPU for more parallelism, whenever possible.

8 Graphical User Interface (GUI)

Due to being an extension of the existing MNE-CPP project, no new GUI is needed. Nevertheless the new features must be integrated into Disp3D and MNE Scan.

Disp3D acts as a test environment with a basic interface. It loads sample data and features various graphical options.
The new functions are executable through the GUI of Disp3D. For that, a new item is created to expand the existing product. (??)

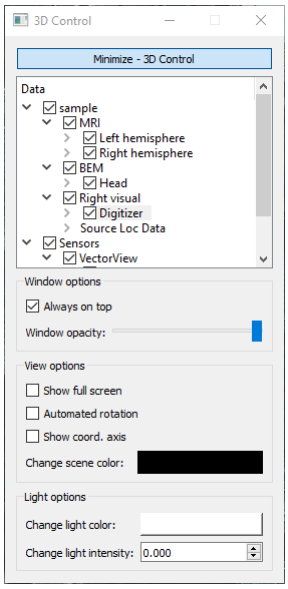

Figure 2: Disp3D

The three standalone applications, namely MNE Scan, MNE Analyze and MNE Browse, can be accessed by a main launcher. Alongside Disp3D, the new features are executable through the MNE Scan application.

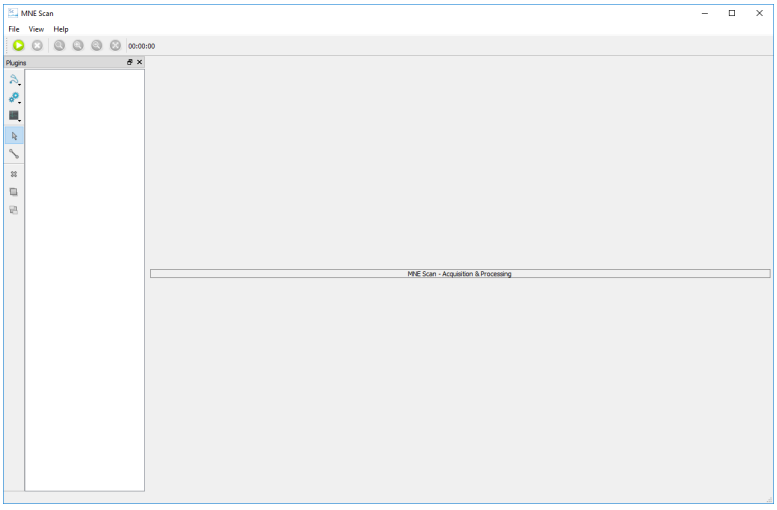

Figure 3: MNE Scan

9 Quality Target Acquisition

	Importance			
	very important	important	less important	not important
Sturdiness			•	
Reliability		•		
Correctness		•		
Usability			•	
Efficiency	•			
Portability		•		
Compatibility		•		
Security				•

10 Test Scenarios

10.1 Functional Testing

10.1.1 Projection Testing

To verify the sensor-mesh-mapping algorithm in case of EEG-sensors, the closest vertex to a sensor can be determined by calculating the euclidean distance between said sensor and each vertex of the mesh and comparing the minimal result of this calculation with the outcome of the projecting algorithm. In case of MEG-sensors, the algorithm can be tested by comparing both euclidean distance and solid angle between each vertex and the sensor, i.e. the sensors location combined with its orientation.

10.2 Nonfunctional Testing

10.2.1 Computation Time Testing

In order to identify time consuming sections of code, the passed time is recorded for every major step within the respective features of the program. This helps to ensure a high level of efficiency.

10.2.2 Memory Allocation Testing

Since the program is to be run alongside an ongoing MEG/EEG-scan, it must not take up too much memory. To assure this, the program should be profiled with suitable tools during runtime.

10.2.3 Pattern Testing

As there already are more or less accurate implementations of the interpolation, the general pattern of brain activity during one example can be recorded using these preexisting tools. Later, this can be compared to the output of the real-time mesh utilities.

11 Glossary

MNE-CPP is a cross-platform C++ framework that provides MEG/EEG tools and applications for fast non-invasive brain monitoring.

MEG stands for Magnetoencephalography.

EEG stands for Electroencephalography.

Vertex is a point where two or more lines or edges meet.

Mesh contains a number of vertices and edges, forming a 3D shape. Often these vertices and edges are structured as triangles.

Euclidian Distance is the shortest distance between two points in a 3D space.

SCDC stands for surface constrained distance calculation. A shortest path between two points on a tessellated surface is determined.

Interpolation is the mathematical process of determining the value of a function between two points at which it has prescribed values.

Bad Channels are sensor inputs that do not deliver working signals or are not set up correctly.

CPU (central processing unit) is alternately referred to as processor and is the computer component that is responsible for interpreting and executing most of the commands from other hardware and software.

GPU (grapics processing unit) is a programmable logic chip specialized for display functions. It renders images, animations and videos for the computer's screen.

GPU-Level is a term for using the GPU (possibly instead of the CPU) to do calculations.

Shaders are used in 3D computer graphic calculations. A shader is a small program or set of algorithms that determines how 3D surface properties of objects are rendered, and how light interacts with the object.

Compute Shaders are a shader type in OpenGL and Direct3D. They can execute calculations outside the typical rendering pipeline, thus making parallelization possible.

Qt is a cross-platform framework, mainly used for software applications and graphical user interfaces (GUIs).

Eigen is a library that provides mathematical functions for programming.

Efficiency describes an economical usage of resources. Low storage consumption and small GPU workload are optimal for the system.

Sturdiness is the ability of the system to function properly and without failures even under non-optimal circumstances.

Reliability measures the possibility of wrong output or total failure of the system.

Security describes how easily a software can be manipulated.

Portability is the ability of software to be ported from one system to another.