Software-Project 2017

Functional Specification

Real-Time Mesh Utilities

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

last Change: April 28, 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 this 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, migraine variants and other brain diseases. In addition, assistance in identifying brain death is one 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 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)

Note: From this point on **S**urface **C**onstrained **D**istance **C**alculation will be referred to as **SCDC**.

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 core feature is the ongoing interpolation, visualizing a particular set of sensor data, representing the brains 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 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).

C134 Bad channels, a part of the given sensor 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 GUI, 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 has to run on the latest versions of 2 operating systems, namely Windows and Linux.

C152 Features 2.1.1 and 2.1.2 are implemented in the class *GeometryInfo*, while 2.1.3 is facilitated in the class *Interpolation*.

C153 The function to integrate the product into Disp3D (2.1.4) 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 integratable into MNE-Scan.

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 (2.1.1)

C211 The computation time should be as fast as possible.

C212 A threshold is used for reduction of processing time.

2.2.2 Interpolation (2.1.3)

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 application

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 framework is already used to do research on certain brain-functions and diseases (e.g. epilepsy). The program will contribute to the functionality of the whole framework. The library will be used with the front-end-applications already existing, mainly MNE-Scan.
A second application is the usage in new programs based on the library. Users can rely on the functionality and build new software for his needs.

3.2 Potential target audience

The main target group of the project consists of scientists and physicians that want to research and analyze the behavior of the human brain. It 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 do the SCDC (2.1.1) "offline" (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 to visualize MEG and EEG data live. The signal derives from actual activity in the brain which the software should reconstruct.

4.2 Software

4.2.1 The MNE-CPP framework

The MNE-CPP framework provides several applications dividable by front-end and back-end software. MNE Scan, MNE Browse and MNE Analyze are front-end applications which perform well in terms of 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 able to run on both, Windows and Linux to maximize the possible cases of usage. 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^{TM}) 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 part of MNE-Scan, a subprogram of MNE-CPP.

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 Online functions

These functions are performed continuously and in real-time.

F32 Based off prior calculations, the interpolation assigns every vertex a value.

F31 A visual output is generated.

6 Product Data

The following points depict the data used in the program.

D11 *Meshs:* A mesh consists out of vertices which represent a 3D model, most likely of a brain. 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 and potentially a subset of it.

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 comes as a matrix. Every row represents one signal of a sensor. Every column defines a certain point of time.

D15 *Intensity vector:* The intensity vector is an amount of values that specify the color 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 *Low latency:* In order to provide the real-time functionality, the latency has to be kept as low as possible.

S13 *Efficiency:* The program should work on normal hardware and be efficient enough to deliver its performance on most setups.

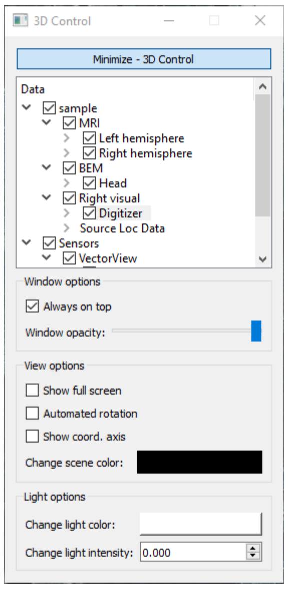
S14 *Sturdiness:* According to the variety of hardware setups the program also should be flexible on the software side.

S15 *GPU-Usage:* For maximum efficiency some operations should use the GPU instead of the CPU for more parallelism.

8 Graphical User Interface (GUI)

Due to being an extension of the existing project MNE-CPP, 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. (2.1.4)



Data
- ✓ sample
  - ✓ MRI
    - ✓ Left hemisphere
    - ✓ Right hemisphere
  - ✓ BEM
    - ✓ Head
  - ✓ Right visual
    - ✓ Digitizer
    - Source Loc Data
- ✓ Sensors
  - ✓ VectorView

Window options
- ✓ Always on top
- Window opacity:

View options
- ☐ Show full screen
- ☐ Automated rotation
- ☐ Show coord. axis
- Change scene color: ■

Light options
- Change light color: □
- Change light intensity: 0.000

Figure 1: Disp3D

The complete MNE-CPP project consists of 3 big subprograms, namely MNE-Scan, MNE-Analyze and MNE-Browse, which can be selected by a main launcher. Alongside Disp3D, the new features are executable through the MNE-Scan application.

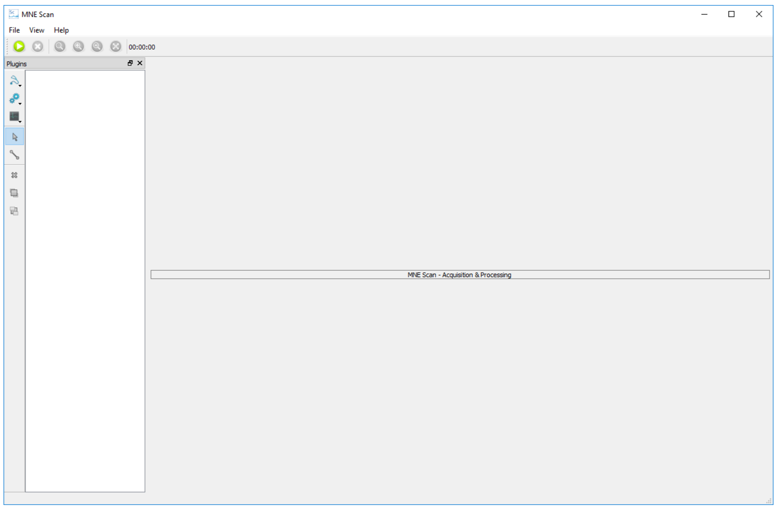


Figure 2: MNE-Scan

9 Quality Target Acquisition

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

10 Test Scenarios

11 Glossar