Thomas Eixelberger, BSc.

realDTI
A software package to visualize diffusion tensor imaging data in the Unreal Engine 4 using the Oculus Rift and Leap Motion

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Biomedical Engineering

submitted to

Graz University of Technology

Supervisor

Assoc.Prof. Dipl.-Ing. Dr.techn Reinhold Scherer

Institute of Neural Engineering

Graz, April 2017

AFFIDAVIT

Abstract

Alzheimer's diseases, stroke and multiple sclerosis are a few examples for nervous diseases. They can be investigated with the help of diffusion tensor imaging (DTI). This technique represents the diffusion process of molecules which can be visualized via fiber objects which stand for the direction of the diffusion. The doctor can investigate the fibers as broken or missing fibers are an evidence for a nervous disease.

Within the framework functional magnetic resonance imaging (fMRI) data are used and visualized. The calculation of the fibers is done by existing software packages. The main focus is on the realization within the gaming engine. This work is a proof of concept and represents the first step into a new research area in which the scientific data is connected to a gaming application. The purpose of this is accomplish evaluation of scientific data via easy tools.

The main idea of this project is to develop a software environment which visualizes DTI fibers using the Unreal Engine 4. The Unreal Engine symbols one of the high performance open source gaming engines which can operate with a virtual reality headset. It is also the case that the fibers should be visible on a normal PC monitor and on an Oculus Rift virtual reality headset. Using the Oculus Rift stands for working with the state of the art headset at the start date of this thesis. The control of the program should be done by the Leap Motion gesture control but as well using mouse and keyboard. One of the best ways to a use a widely-used consumer motion controller represents the Leap Motion controller.

Moreover the pros and cons of a visualization using a VR headset for an user are tested. It is the same case for using the Leap Motion as a control.

Keywords: Unreal Engine 4, diffusion tensor imaging, Leap Motion, Oculus Rift, fiber tracking

Kurzfassung

Alzheimer, Schlaganfälle und Multiple Sklerose sind nur einige Vertreter von vielen Nervenkrankheiten. Sie können mittels Diffusions-Tensor-Bildgebung (DTI) untersucht werden, Diese Technik stellt den Diffufsionsprozess von Molekülen dar, der nun mittels Traktografie als Fasern, welche die Diffusionsrichtung repräsentieren, dargestellt werden kann. Ärzte können diese Fasern untersuchen und durchgebrochene oder fehlende können als ein Indiz für eine Nervenerkrankung ansehen.

Im Rahmen dieser Arbeiten werden Funktionelle Magnetresonanztomographie (fMRI) Daten verwendet und grafisch dargestellt. Die Berechnung der Fasern wird durch bestehende Softwarelösungen realisiert. Dadurch soll der Hauptfokus auf die Realisierung innerhalb der Spieleumgebung liegen. Diese Arbeit stellt eine Machbarkeitsstudie und den ersten Schritt in ein neues Forschungsgebiet, welches Forschungsdaten mit Spieleanwendungen verbindet. Somit wird erhofft dass man mittels einfacher Werkzeuge die Auswertung von wissenschaftlichen Daten erreicht werden kann.

Die Idee hinter diesem Projekt ist die Erstellung einer Softwareumgebung welche DTI Fasern mittels der Unreal Engine 4 darstellen kann. Die Unreal Engine 4 repräsentiert eine Hochleistungs-Spiele-Engine welche Open Source ist und mit Virtual Reality umgehen kann. Weiters sollen die Fasern neben dem normalen PC Monitor auch mittels einer Oculus Rift Virtual Reality Headset betrachtet werden können.Die Oculus Rift stellte zu Beginn dieser Arbeit das Stand der Technik im Bereich von Virtual Reality Headset dar. Die Steuerung des Programms soll mittels einer Gestensteuerung geschehen oder wahlweise per Maus und Tastatur. Eine der besten Wege um eine weitverbreitete Endverbraucherlösung für Gestensteuerung zu benutzen stellt die Leap Motion dar und wird deshalb auch hier verwendet.

Außerdem wird getestet ob die Visualisierung mittels eines VR headset für die Benutzer von Vor- oder Nachteil ist. Gleiches gilt auch für Steuerung mittels der Leap Motion Gestensteuerung.

Schlüßelwörter: Unreal Engine 4, Diffusions-Tensor-Bildgebung, Leap Motion, Oculus Rift, Traktografie

Contents

1. Introduction and problem

1.1. Motivation

Many nervous system diseases like Alzheimer's disease, stroke or multiple sclerosis can be investigated with the help of diffusion tensor images (DTI). It allows representing the diffusion process of molecules and it shows the results as fiber objects. For an example the detection of a stroke is described in [21] and the investigation of a rehabilitations technique via DTI is done by [17]. The desired brain areas have minutes after a the event a lower diffusion coefficient comparing to healthy areas. This can be explained through the canceled sodium-potassium pumps that allows extracellular liquid to pour into the cell. DTI can also be used for an early discovery of multiple sklerosis [4] and for the evaluation of brain damage [14]. Finally, the last big research area of DTI is the analysis and detection of Alzheimer's disease. The paper [12] describes the detection and [1] used the technology to understand the disease better. In both cases the fractional anisotropy is decreasing and the diffusivity is increasing. This mechanism yields to a change of the measured signals.

All these articles are just few examples for using the DTI technology at everyday clinical practice and it can be done with nearly every new magnetic resonance tomography. To sum up it is a practicable non-invasive diagnostic system with an enormous research potential. This thesis is a small part of it and can make the handling with these diseases easier.

The main purpose of this work is to develop a framework which allows analyzing DTI fibers using an Oculus Rift VR headset. It should evaluate the usage of the Unreal Engine and the Oculus for functional magnetic resonance imaging (fMRI) data. It ought also to give a statement about the usability of a Leap Motion gesture controller at this particular problem.

The virtual reality headset could help doctors with the interpretation of the fibers. People get a better feeling for the structure of the brain and the 3D elements maximizes the immersion. Using the Oculus Rift and the Leap Motion is a cheap method to analyze fMRI data via a standard PC without any special software and hardware.

1.2. Background

Diffusion tensor imaging (DTI) [7] is the most used variant of diffusion-weighted magnetic resonance imaging (DW-MRI). It measures the diffusion movement of water molecules with the help of magnetic resonance tomography (MRI) and displays them graphically [23]. The contrast of the image is generated by only using a pulsed-field-gradient, like the classical MRI, and it is not necessary to inject a contrast medium. DTI also determines the anisotropy of the diffusion and assigns it as a tensor to every voxel of the images.

1.2.1. Basic principle

The basic principle is based on the first Fick's law. It describes the relationship between the flux J and the concentration gradient ∇c.

$$J = -D\nabla c \tag{1}$$

$$D = \begin{bmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{yx} & D_{yy} & D_{yz} \\ D_{zx} & D_{zy} & D_{zz} \end{bmatrix} \tag{2}$$

In this formula D is the diffusion tensor in an anisotropic medium. It is the case that the MRI measures the self diffusion of water also called Brownian motion. The movement is initialized by thermal energy. Technically spoken the diffusion is not applicably because there is no free diffusion of water molecules. So the tensor is called apparent diffusion coefficient (ADC) and it depends not only from the direction but also from the diffusion length.

If there is no diffusion the change in nuclear magnetization over time will be given by the classical Bloch equation. M represents the magnetization and T_1 and T_2 the relaxation times.

$$\frac{dM}{dt} = \gamma\left(M \times B\right) - \frac{M_x\vec{i} + M_y\vec{j}}{T_2} - \frac{\left(M_z - M_0\right)\vec{k}}{T_1} \tag{3}$$

The result of this equation by given diffusion in the medium yields to the Stejskal-Tanner-equation [18] and for a detailed explanation derivation look at the PHD thesis of Ferran Prados Carrasco [2].

$$A(g) = A_0 e^{-bg^T Dg} \tag{4}$$

Here symbols $A(g)$ the signal strength of a gradient field in the desired direction g. A_0 describes the strength at an unweighted measurement and b summarizes different measurement parameters. The positive semi-defined quadratic diffusion tensor D assigns every direction g an ADC. D can be estimated after a measurement using the least squares method.

1.2.2. Fiber tracking

DTI gives information about connectivity and organization of white matter fibers and the 3D pictures are developed based on tensors from voxels. It is an method which describes the nature of diffusion in a given volume and fiber direction is the main eigenvector. Furthermore the fiber goes in direction of the largest diffusion coefficient.

This method allows distinction between normal and damaged areas because water molecules move parallel to normal fibers and the movement is perpendicular around damaged areas. This means a find of damaged areas after a stroke can be done easily because the fibers are broken at the damaged areas.

1.3. Goal

The purpose of this thesis is to develop a software package with a gaming engine to visualize DTI data by a virtual reality headset and control it using a motion controller. Based on previous research which was done during the master practice before this thesis a stereoscopic visualization could improve the clarity of the calculated fibers. The purpose of using a gaming engine in this case the Unreal Engine 4 is that the software package can run on a standard PC without any special software. The chosen VR headset is the Oculus Rift because it is a common consumer headset and the Leap Motion represents a cheap widely used motion controller. The next reason for a decision for it is that the Leap Motion provides a decent amount of pre-configured gestures. It is just a proof of concept and the first step into a new research area.

There are three main questions to answer:

1. Is it possible to visualize DTI data in the Unreal Engine?
2. How is the performance and the usability of the Oculus Rift?
3. Is a control using the Leap Motion profitable and intuitive?

2. Methods

The following paragraphs will explain the software packages and the needed steps. All pros and cons of these are described in the discussion part in detail.

2.1. Data acquisition

Figure 1 shows the pre-processing sequence of the 3D visualization. The DICOM (Digital Imaging and Communications in Medicine) files [10] are created via a diffusion magnetic resonance tomography. This was done by a MRI using an EPI (echo planar imaging) sequence.

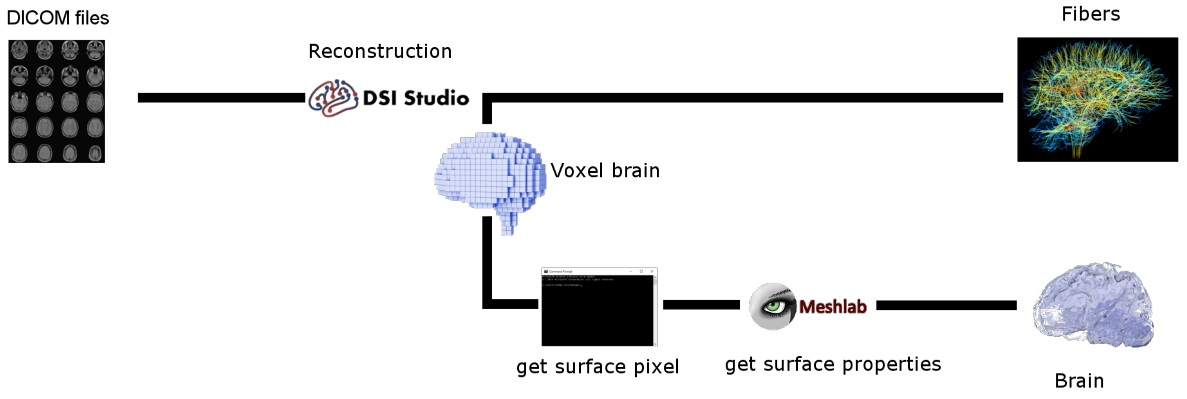


Figure 1: Using DICOM files DSI Studion reconstructs the fibers and the voxels of the brain. The fibers are ready to use in the Unreal Engine but the voxel data has to be converted to surface data. A self programmed CUDA package calculates the surface pixel and MeshLab generates the normals of the surface. Now the brain is also ready to use.

2.2. Reconstruction software

The fibres are generated via DSI Studio [24] (Fang-Cheng, University of Pittsburgh, U.S.) which calculates the fibres with the deterministic diffusion fiber tracking algorithm [23] from the DICOM files. There are many parameters values like the number of the extracted fibers or an addable atlas to render the cortex regions. All standard parameters are described in the manual (section A.1).

After the reconstruction of the fibres and generating the brain volume, it is possible to save the data in different file formats. For this project the data are saved as textfiles and the fiber data is ready to render in the Unreal engine 4, but the brain volume has to be processed further more.

2.2.1. Brain

Get surface pixel A CUDA(Compute Unified Device Architecture) [11] (Nvidia Corporation, Santa Clara, California, U.S.) program was written to convert the voxel volume data to mesh data. All functions are taken from [22]. It deletes all of the voxel which are not on the surface of the volume and saves the remaining data in a structured textfile.

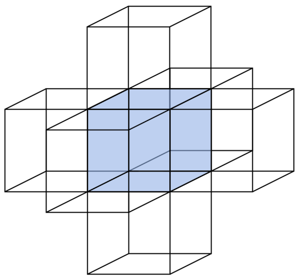


Figure 2: 6-neighborhood of a voxel in the 3D space. Every pixel shares a surface with the blue pixel in the center is a neighbor of the blue one.

For converting the volume data to mesh data, the algorithm looks at the 6-neighborhood of each voxel like shown in figure 2. If the actual voxel has all neighbors the voxel will be inside of the volume and it is deleted. If one or more neighbors are missing the voxel will be on the surface and it will not be deleted.

Get surface properties (Triangulation) After calculating the surface pixel, the textfile is loaded into Meshlab [3] (Istituto di Scienza e Tecnologie dell'Informazione & National Research Council, Italy) where the mesh data is converted into an obj.-file. The conversion is done with the poisson surface reconstruction algorithm [8]. An obj.-file [13] is used because it contains besides the pixel values also information about the triangulation of the surface which is needed to render it in the game engine.

2.3. Unreal 4.10.2

The Unreal Engine 4 is a game engine developed by Epic Games (Cary, North Carolina, U.S.). It can be used for free and includes a wide spectrum of plugins like Oculus Rift and Leap Motion plugin. It is possible to write code in C++ or in a graphical programming language, called blueprint. The blueprint mode is just a frontend which uses C++ code and to program an application the developer has to connect different blocks with each other. Every block is a graphical object for a special C++ code. The main concept of programming C++ in the engine was taken from [16] and of blueprints from [15]. Pros and cons are discussed in the discussion part.

The main procedure of the developed program can be seen in figure 3. It is not necessary to check if the Leap Motion is connected using a program function because the desired plugin detects the controller at runtime and starts the hardware at any time automatically.

After starting the Unreal Engine a function checks if the Oculus Rift is connected correctly. If so the program will be rendered on the VR headset and otherwise it will be rendered on the normal PC display. The whole procedure is described in chapter 2.7. A splash screen (2.6) is be shown after a health warning on the screen for 12s and in the background the properties from the configuration files are loaded into variables. It is also the case that the program loads the data of the fibers and the brain. After that the objects will be rendered on the screen which is described in the next paragraph and the software is waiting for an user input via the Leap Motion or the keyboard and mouse. There are different user inputs like opening a menu, taking a screenshot, translation and rotation of the objects. The input methods can be found in 2.8.

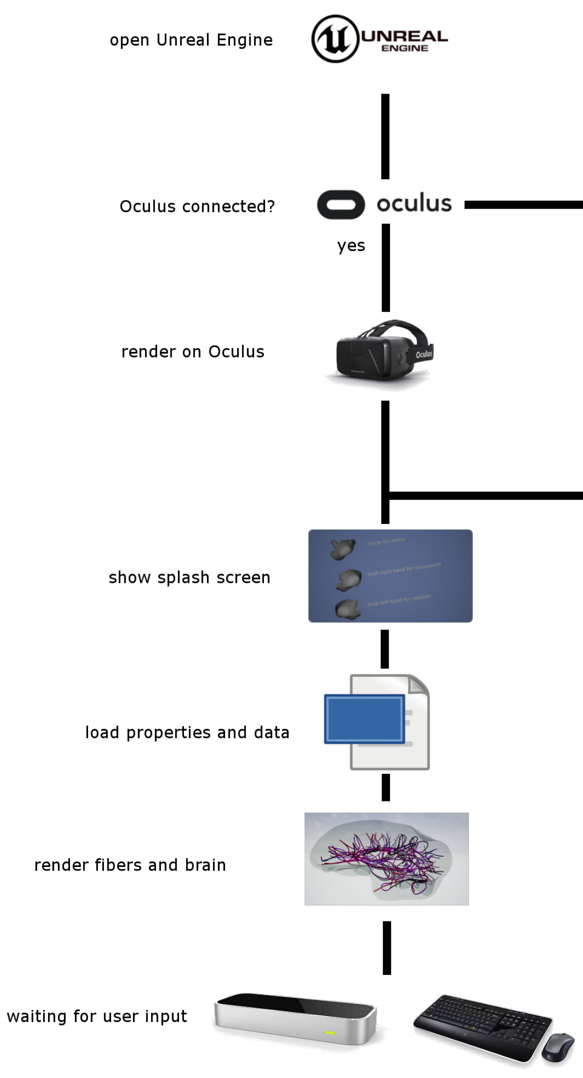


Figure 3: Main procedure of the Unreal program.

2.4. Rendering objects

Every object has to have a point of center, a shape and a shader. The center specifies the location of the object and the shape is a wire frame of the element. Moreover the shader describes a surface over the wire frame with different properties like color and roughness. The next two paragraphs describe the algorithms to render the brain and the fibers in the Unreal Engine.

Brain The first step of the render algorithm for the brain surface is to load the data from the `.obj` file [20]. The file includes the triangles of the surface and the vertexes of these geometrical shapes. Unreal Engine 4 does not support direct input of data at runtime via blueprints and so it is useful to write a little C++ function for doing that. This is important because otherwise the data had to be loaded before the program is compiled and the user had to compile the Unreal program whenever new datasets should to be loaded. For further information look at the following paragraph in 2.6.

After loading the triangles they are added to a new mesh and having done that the properties (shader, location,...) of the mesh are set to the desired values. For later manipulations (translation, rotation) of the brain it is important to calculate the point of mass. For doing that the coordinates of the vertexes are summed up and divided by their count.

Fibers The algorithm to render the fibers is more complicated and needs more steps as seen in figure 5. First of all the data is loaded from the text file. Secondly a new mesh is created and the first 2 points are added to the mesh. These points are the start and the end point of the mesh. Then the direction of the tangent in the start and end point are calculated using the precursor respectively successor point. This means that direction vector from the point before the start and the direction from the end to the point after this one is figured out. The precursor and the successor are displayed as red dots in figure 5. If there is no data point before the start or after the end the direction vector from start to end point or vice versa is used as the tangent. So the spline mesh gets a curve form. After that a cylindrical shape is pulled over the mesh to get the appearance of a fiber. Every element get its shader and certain color assigned. These steps are repeated for all points. At the end all meshes are combined to one big array of meshes so that the second mesh is attached to the first and so on.

Figure 4 shows the cylindrical shape with the standard Unreal Engine shader. This can be stretched and follow the path of the mesh.

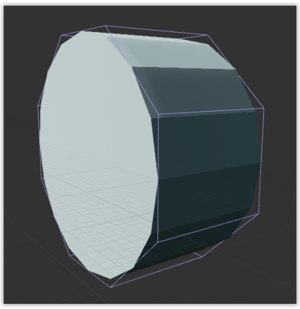


Figure 4: Cylindrical shape of a fiber mesh.

The colors of the different meshes describe a gradient from the start to the end color. These steps are repeated for every fiber, so every filament is a single object and can be manipulated later.

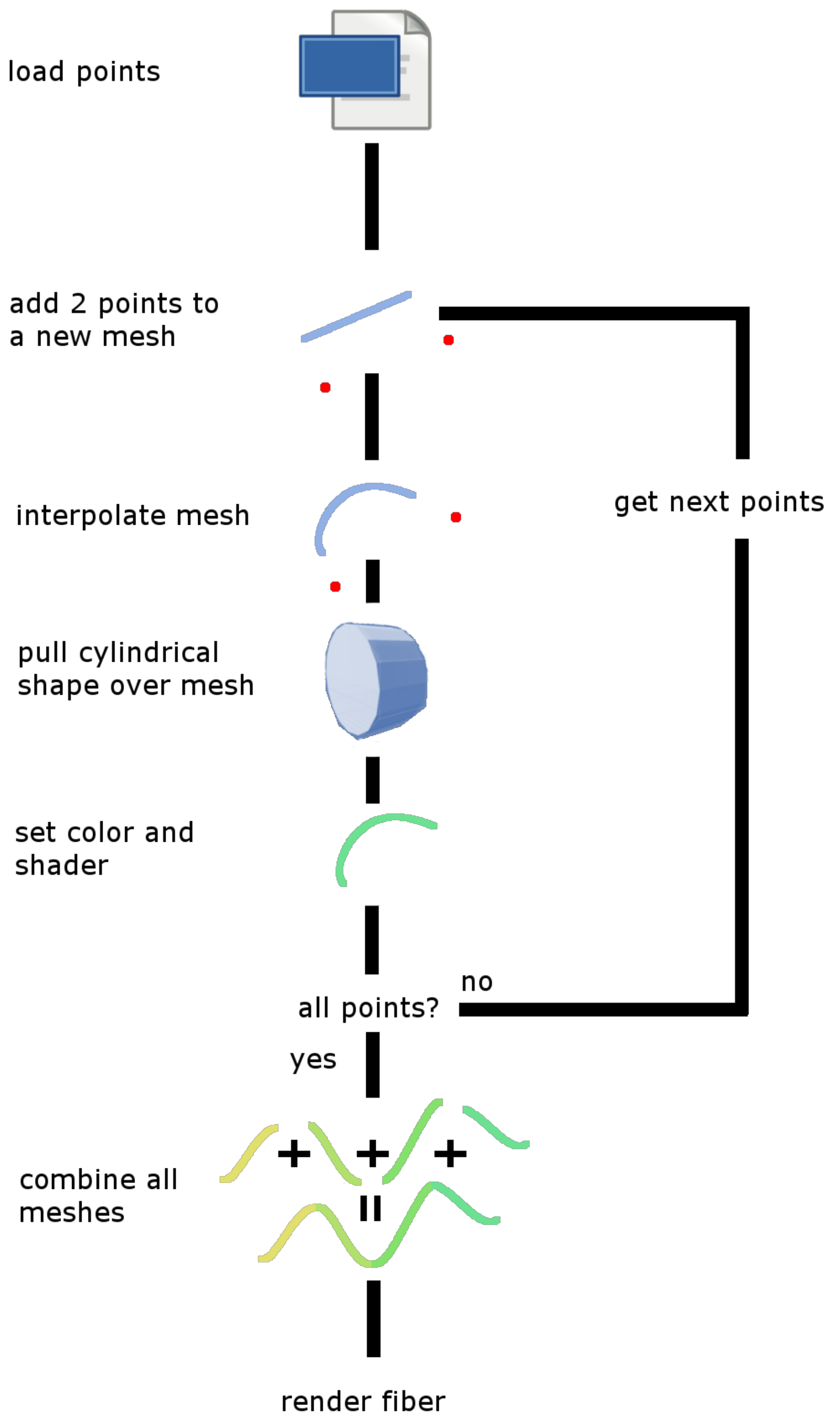


Figure 5: Render algorithm for one fiber. After loading the data the points are added in pairs to new meshes. Then the meshes are interpolated and a cylindrical shape with a certain shader is pulled over. After setting the color the meshes are combined to one. The last step is to render the fiber.

2.5. Main program

The main program consists of two parts. First the start up program, written in C#, and secondly the Unreal Engine program.

2.5.1. StartUp program

The purpose of the software is to read the configuration files and to manipulate them before using the VR headset. After that the program starts the Unreal program.

Configuration files There are two different files, `ColorProp.ini` and `MenuConfig.ini` to load and save different properties. The first includes the RGB color code in a normalized way of the fiber start, fiber end, brain and the background (figure 6b). `MenuConfig.ini` saves the absolute file paths of the fiber data and the `.obj` file of the brain. It is shown in figure 6a and the value `true` represents if the fibers respectively the brain is rendered.

```
true
F:/Data/hundred.txt
true
F:/Data/brain.obj
```

(a) `MenuConfig.ini`

```
1.0 0.0 0.0
1.0 0.0 1.0
1.0 1.0 0.0
0.706284 0.823368 1.0
```

(b) `ColorProp.ini`

Figure 6: Example configuration files.

2.6. Menus

It is possible to implement menu widgets in the Unreal Engine but the 2D menu has to be contained in a 3D object. This leads to a three dimensional look and it is necessary to display it on the Oculus Rift. First of all a normal 2D menu widget is created and all functions like a click event are implemented in the background code of the widget. Secondly the widget is attached to an empty actor and thirdly the actor is spawned in the desired level in front of the camera. The empty actor with the attached menu widget is shown in figure 7. It is important that the normal of the object is parallel to the viewing direction to face the menu in front of the user.

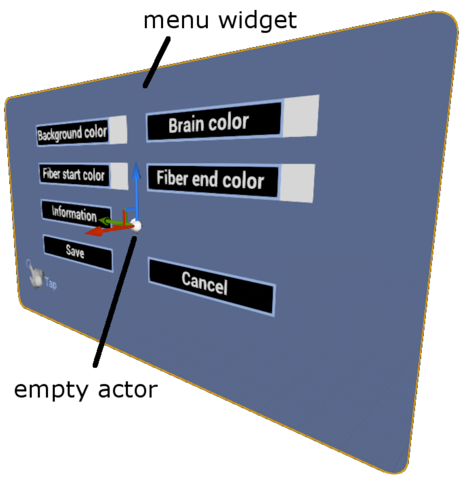


Figure 7: Menu widget with the empty actor to spawn it as a 3D object.

Pointer A menu also needs a pointer like a mouse pointer on the desktop. A 3D pointer is required for the output on the Oculus Rift. For realizing that a sphere is spawned in front of the camera between the surface of the menu and the camera as displayed if figure 8. A ray is generated from the middle of the camera through the sphere and to the menu. After that the hit point on the menu widget is read out and an algorithm calculates which object on the menu was hit. If a button was hit the button color would change and if the tap gesture happens the click event of the desired button will be triggered.

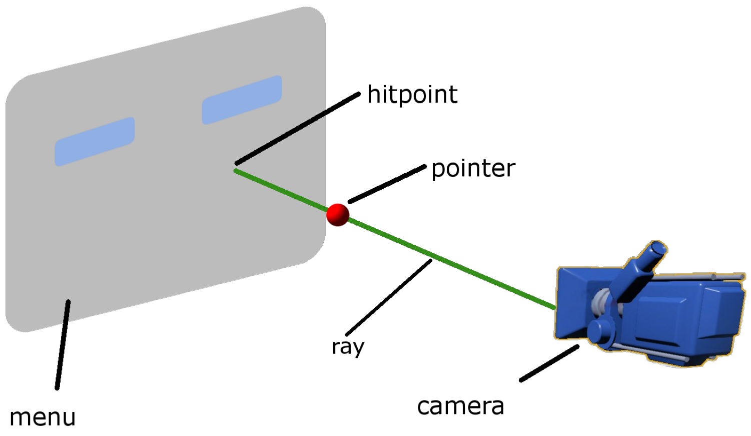


Figure 8: The ray runs from the camera through the pointer and hits the menu.

Loading and saving of properties At the start of the Unreal Engine program the color properties are loaded from the configuration file and if the user changes the values they will be saved. There is no blueprint object which realizes this and so the loading and saving procedure has to be written in C++. It is possible to develop own blueprint object by the following code line in the header file of a `BlueprintFunctionLibrary`.

```
UFUNCTION(BlueprintCallable, Category = "MyUtilities")
    static bool FileSaveString(FString SaveText, FString FileName);
```

The standard C++ header files of Unreal does not include any IO functionality and so it is necessary to include the `DesktopPlatformModule.h` and `IDesktopPlatform.h` from the Unreal C++ libraries. These two header files include the IO functionality. The following code snippet shows the implementation of a file save object.

```
#include "Utilities.h"
#include "Developer/DesktopPlatform/Public/DesktopPlatformModule.h"
#include "Developer/DesktopPlatform/Public/IDesktopPlatform.h"


bool UUtilities::FileSaveString(FString SaveText, FString FileName)
{
    return FFileHelper::SaveStringToFile(SaveText, *(FPaths::GameDir() +
        FileName));
}
```

Figure 9 shows the compiled save file object as a blueprint element. It is possible to insert the text which should be saved at the `Save Text` input parameter and `File Name` defines the filename where the text is saved to. The `Return Value` delivers a boolean value of achievement.

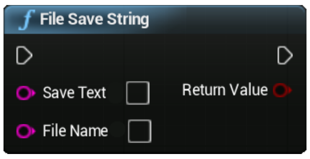


Figure 9: A self programmed blueprint object which saves a text to a chosen file.

Be aware of different data types in the Unreal Engine. The data types are not the standard C++ data types and for more information look at [1].

Splash screen This menu should be rendered when the software starts. It is painted for 12s and shows the instruction manual of the Leap Motion gestures to the user. The Unreal Engine also loads the properties from the configuration files. All gestures are animated and this is made by the animation editor. The gaming engine also tries to load the Oculus Rift camera. If the VR headset is not connected or the stereo output does not work the output device will be changed automatically to the pc display.

[1]`https://docs.unrealengine.com/latest/INT/Programming/Introduction/`

Main menu All parameters like background color, color of the brain and so on can be changed at runtime. The main menu provides this ability to the user. Moreover the user can operate using the 3D pointer which was mentioned previously and all parameters are loaded or saved like described in the above paragraph. To change the values of the colors an object to manipulate is necessary.

Color picker It is a dynamic menu which gets desired color from the main menu and the color can be manipulated by the different RGB channels. It is necessary to operate with it through the Leap Motion and so the interface has to be as simple as possible. After closing the menu it sends the adjusted color back to the main menu.

2.6.1. Shaders

In Unreal Engine 4 it is possible to create shaders in a graphical interface like the blueprint interface or code like with the HLSL (High Level Shading Language) compiler. All shaders are developed in the visual environment and they also should be adjustable at runtime. This is made by transfer parameters. So it is possible to create an instance of the material and change the value of the variable at runtime.

Brain The brain shader should look like glass with refraction and reflections. All parameters like refraction coefficient, base color and so on are changeable during the runtime. So the main menu can adjust the color of the brain. If the shader is used within another instance the refraction and reflection to create different looks can be changed in the code without changing the main shader. To create a glass shader it is necessary to change the blend mode to translucent so that the Unreal Engine creates a transparent material.

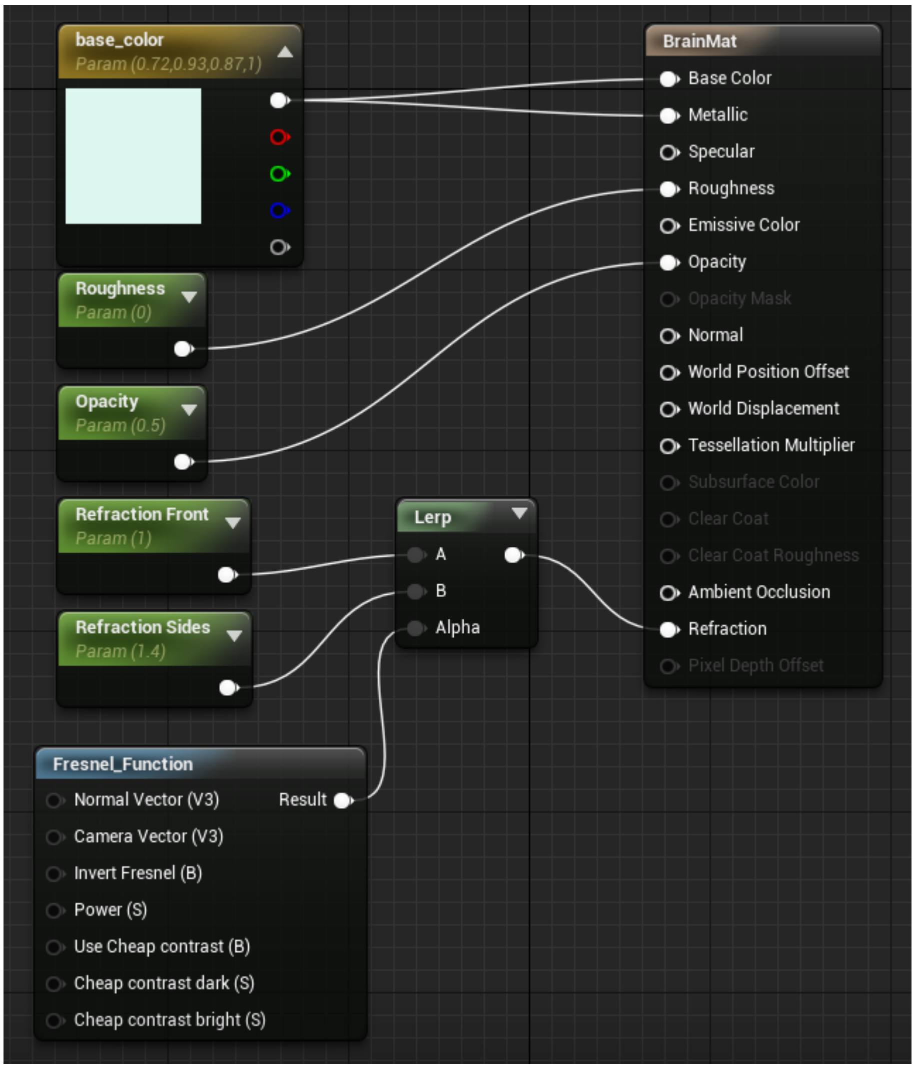


Figure 10: The parameter `base_color` changes the main color of the glass. Roughness represents the roughness of the glass. Is the roughness low the material is smooth. `Opacity` gives information about the opacity. Is the opacity 1 the material is intransparent and if it is 0 the material is invisible. `Refraction` changes the level of refraction of the material. `Refraction Front` represents the refraction coefficent of the environment and `Refraction Sides` the coefficent of the material.

Figure 10 shows the glass shader which is used for the brain material. `base_color` is used to change the main color of the material and can be changed by the user within the main menu. `Roughness` is set to 0 and `Opacity` to 0.5. The `Refraction` of glass equals 1.4 to 1.5 according to physics. It is set to 1.4 at this project.

Fibers The fiber shader should be a smooth material like plastic. For doing this it is necessary to use the opaque blend mode to create an intransparent material and the shading model is changed to clear coat so that the material is shiny.

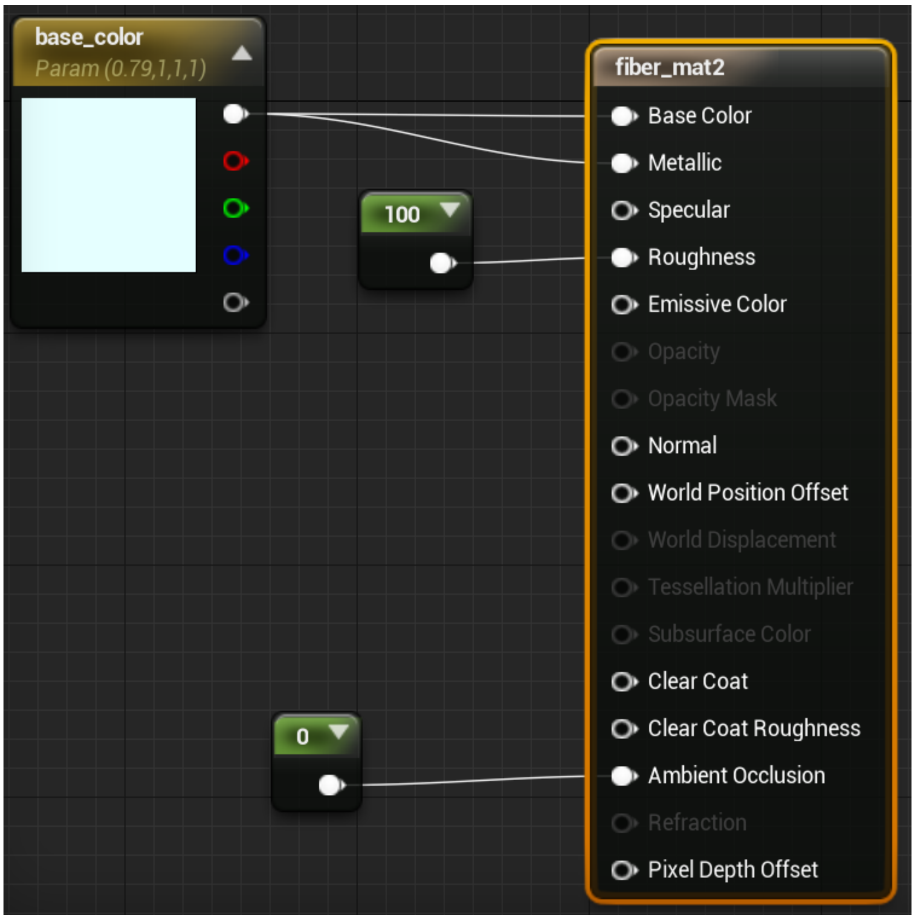


Figure 11: `base_color` changes the main color and `Metallic` gives information about the color of the reflected color. Roughness represents the roughness of the material. Is the roughness low the material is smooth. `Ambient Occlusion` represents the ability of light the environment around the material via the reflected light.

For the fiber material the parameters in figure 11 are set to the following values. The `base_color` could also be adjusted at runtime and the `Roughness` is set to the constant value of 100. Through this the material gets a natural touch and `Ambient Occlusion` is set to 0 because if the value is higher the fibers will light the environment and the inner surface of the brain will also be lighted with the same color. It would not be possible to see through the brain surface if the surface had lighted up too strong.

Instances Figure 12 shows the creation of material of fibers. This dynamic instance can be adjusted at any time.

2.7. Output devices

The Unreal Engine tries to start the stereo mode during showing the loading screen but if the start of this mode does not work the program will automatically change to the 2D

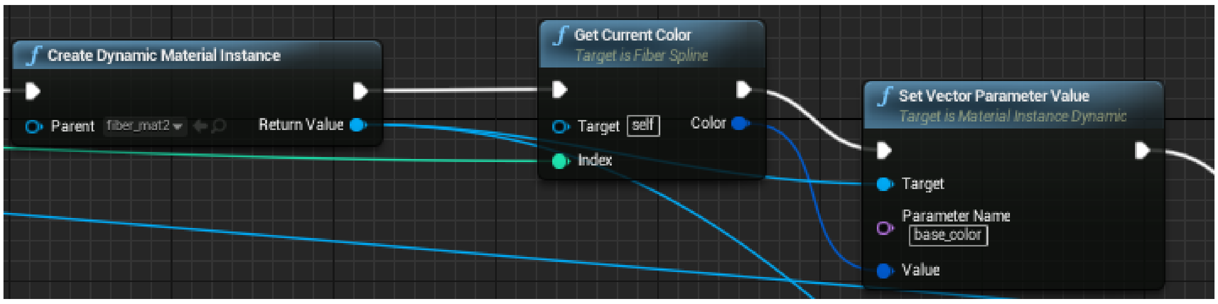


Figure 12: To create an instance of a material it is necessary to create it using the block `Create Dynamic Material Instance` and after that the parameters of the shaders can be changed by `Set Parameter Value`. The block `Get Current Color` is just to get the color which is set at the main menu.

pc display mode.

2.7.1. Oculus Rift

The Oculus Rift [19] (Oculus VR, Menlo Park, California, U.S.) is a VR headset and there are 3 different versions available at the moment. For this project just the Development Kit 1 and 2 were used and tested with the project. All uses of Oculus Rift in this thesis include both versions.

The minimum system requirements are displayed in table 1. For the Oculus DK1 it is just possible to use the SDK 0.8 and for the DK2 both SDK are capable.

Table 1: Minimum system requirements

SDK	CPU	GPU	RAM
Oculus SDK 0.8 Beta	2.5+ GHz processor	GTX 600 / AMD HD 7000	4GB
Oculus SDK 1.0+	Intel i5-4590	GTX 970 / AMD R9 290	8GB

The use of the Oculus Rift is really simple. The only important thing is to activate the Oculus plug in in the Unreal Engine as shown in section 13.

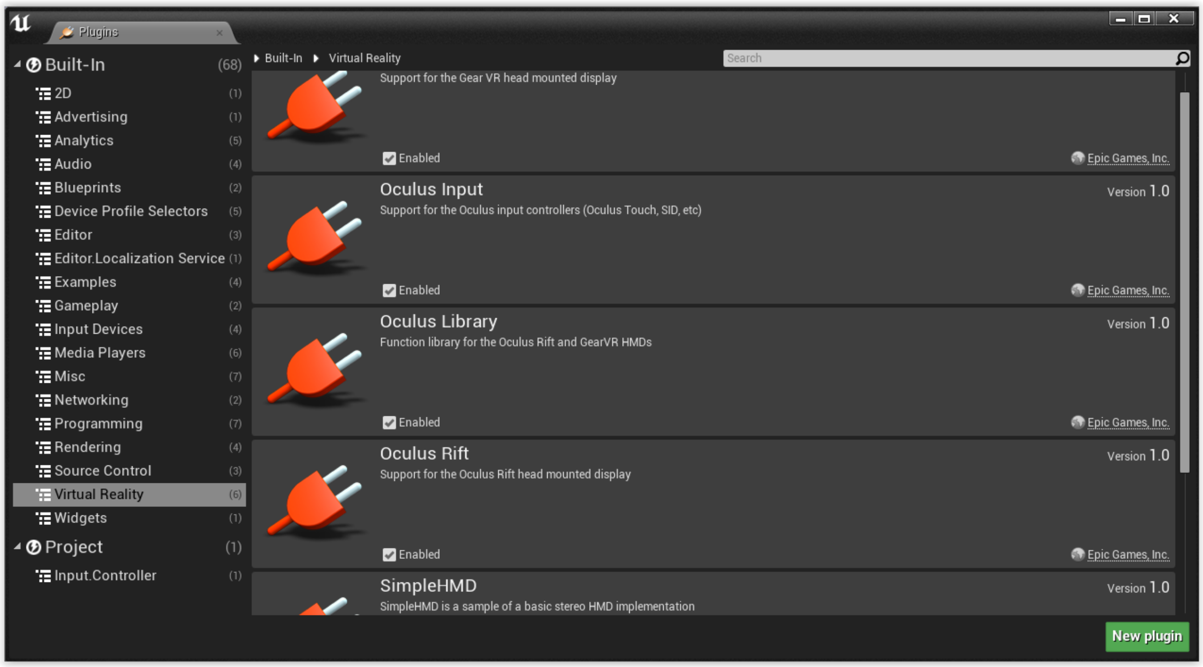


Figure 13: Pugins Editor in Unreal Engine 4. It can be reached by `Edit` → `Plugins`.

To start the stereo mode it is necessary to add the `Execute Console Command` for example in the level blueprint and execute the command `stereo on`. To prevent jitter effects it is necessary to add a post process volume over the viewed objects and set the anti aliasing method to none. Setting the motion blur to 0 can also help.

2.7.2. PC display

If the Oculus Rift is not or not correctly connected the software will load the 2D camera and will project it on the pc display. All menus and properties are the same and work as displayed with the VR headset.

2.8. Input devices

2.8.1. Leap Motion

The Leap Motion [9] (Leap Motion, Inc, San Francisco, California, U.S.) is a motion sensor which recognizes hand and finger gestures. Unreal Engine 4 includes by default a plugin for the leap motion and it does not matter which SDK of the Leap Motion is installed. There are few gestures learned by default and the device differs between right and left hand. All gestures are explained in table 2.

Table 2: All learned gestures of a leap motion.

Circle	A single finger tracing a circle
Swipe	A long, linear movement of a finger
Key tap	A tapping movement by a finger as if tapping a keyboard key
Screen Tap	A tapping movement by the finger as if tapping a vertical computer screen
Grab	Grab indicates how close a hand is to being a fist
Pinch	Pinch indicates whether one finger is touching another, on a scale of zero to one.

It is also possible to get single information about every finger and bone. The values in table 2 are taken from the official documentation [2] [3].

For a better usability the movement of the hand has to be relative to the camera so that for an example a forward movement of the hand in positive z direction does not lead to a movement on the screen in x direction.

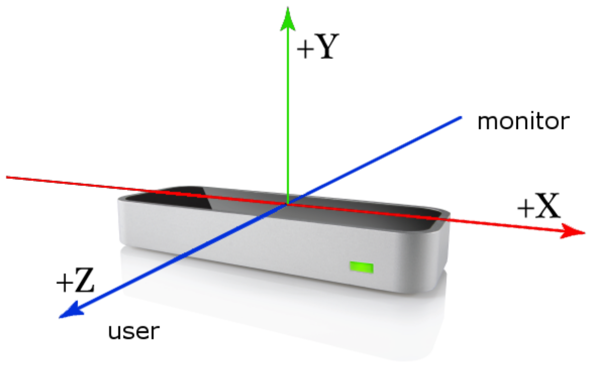


Figure 14: The 3D coordinate system used by the Leap motion and the positive z-axis points from the monitor to the user. [Taken from `https://developer.leapmotion.com/documentation/csharp/devguide/Leap_Overview.html`]

First the software gets the current position of the hand. Secondly it gets the position of the camera and transforms the position of (0,0,0) to the camera space. The position of the hand is also converted to the camera space and after that the transformed position of (0,0,0) is subtracted from the position of the hand. At the end the vector is rotated with the rotation of the camera, so the resulting vector describes the movement of the hand relative to the camera.

Gestures If a gesture is recognized by the leap motion an event will be triggered and the program will execute the code in the event. Figure 15 shows the right hand movement and it will be triggered if any movement of the right is recognized. It does not matter if

[2] `https://developer.leapmotion.com/features`
[3] `https://developer.leapmotion.com/documentation/csharp/devguide/Leap_Gestures.html`

the hand is grabbed, pinched or opened the event will trigger. The following if condition checks if the hand is grabbed and so the event can be used for a grabbed right hand movement.

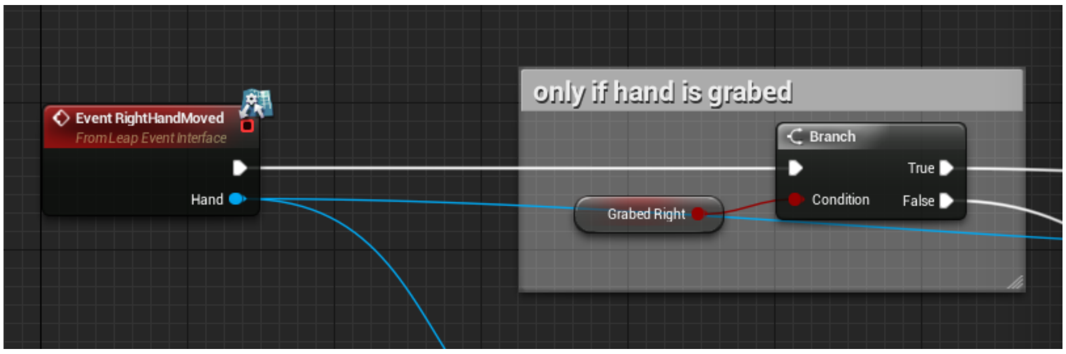


Figure 15: Event of the right hand movement with following if condition.

For different interaction with the software different gestures are used. In figure 16 all used gesture are listed.

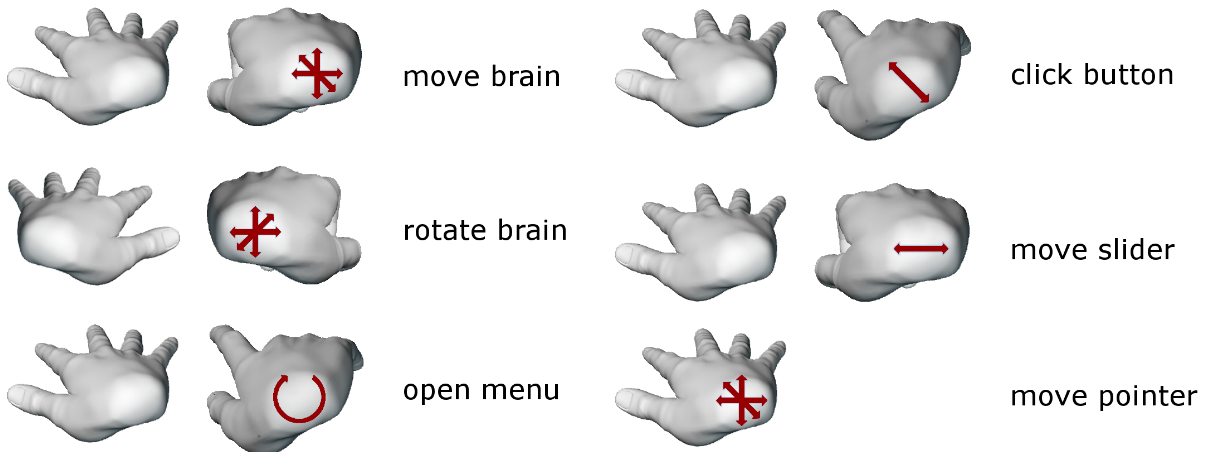


Figure 16: Grab the right hand and move it in x, y and z direction to move the brain relative to the camera. Grab the left hand and move it in x, y and z direction to rotate the brain along the desired axis. Make a circle gesture with a minimum diameter of 5 cm to open the main menu. Tap the right trigger finger to the monitor or the keyboard to click a button. Grab the right hand over a slider and move along the x-axis to change value of the slider. Move the open right hand to affect the menu pointer.

2.8.2. Mouse and keyboard

The whole software can also be controlled by mouse and keyboard like described in figure 17. It does not matter if the Leap Motion is connected or not because the mouse and keyboard will also work.

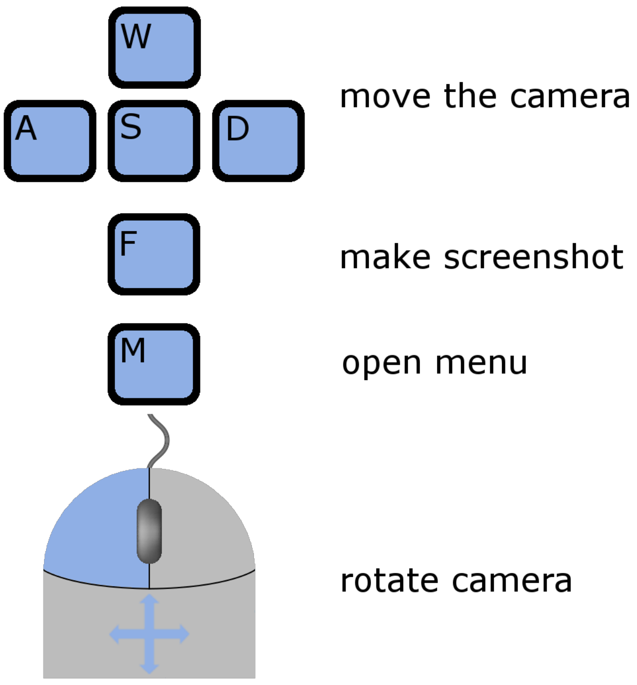


Figure 17: W, A, S, and D move the camera to left and right respectively in or against the viewing direction. Press F to make a screenshot and M to open the menu. By pressing the left mouse button and moving the mouse it is possible to rotate the camera.

3. Results

The result part discusses the main software package. DSI Studio and MeshLab were not developed exclusively for this thesis and so these programs are not a part of the result section. The CUDA program is developed for this but it only got a command line output. How the programs will work is described in the manual as appendix.

3.1. Start up program

The C# program in figure 18 chooses the files and starts the Unreal Engine program after pressing the `Start` button.

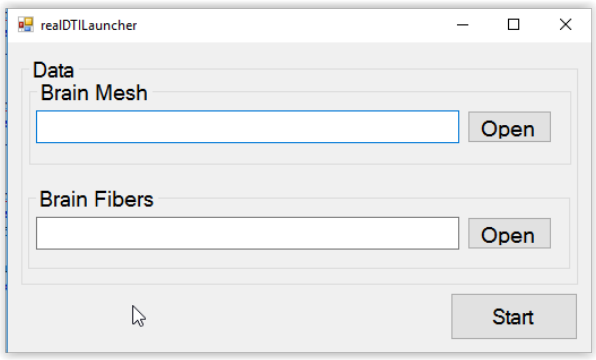


Figure 18: A little C# program to choose the brain surface `.obj` file and the `.txt` file of the fiber data.

3.2. Menus

All screenshots are made with the implemented screenshot hotkey `F`.

3.2.1. Splash Screen

The loading screen gives information about the control using the Leap Motion. All hand objects of figure 19 are animated and they represent the gestures.

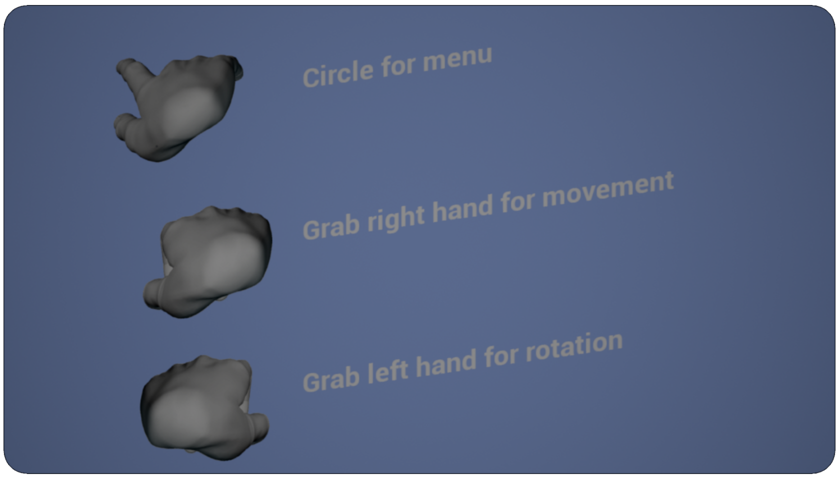


Figure 19: The loading screen shows the gestures and during this some actions are performed.

3.2.2. Main menu

If the 3D pointer is over a button it will be colored in red to symbol that a tap gesture is available. After a tap on the desired button from figure 20 the color picker is opened or the changed colors are saved in the configuration file. The information button opens the loading screen to help the user with the control gestures. Right of the color buttons rectangles are painted which show the current color of the object. The tap gesture at the bottom left corner is also animated.

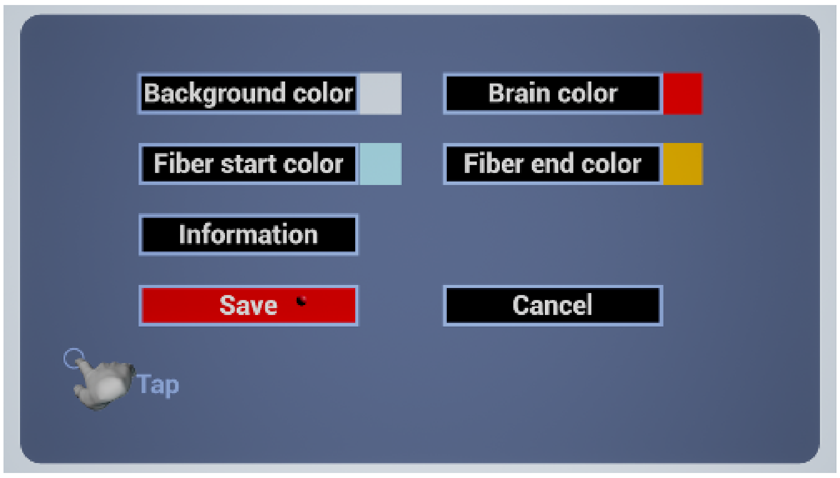


Figure 20: The main menu is opened via a circle gesture or the hotkey `M`.

3.2.3. Color Picker

The color picker in figure 21 changes the RGB values of the desired color. After tapping the OK button the values are send back to the main menu.

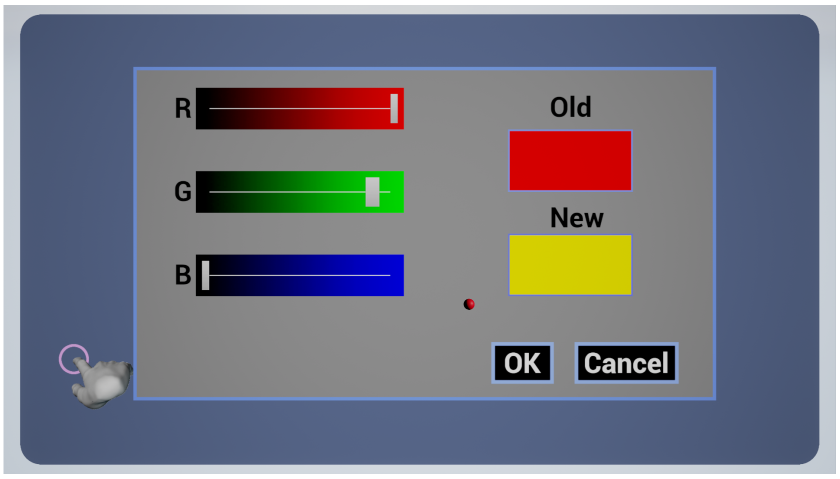
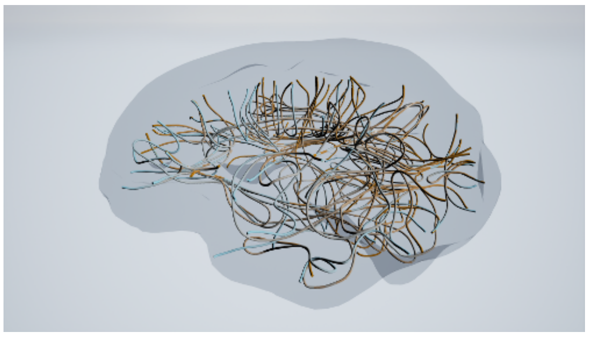


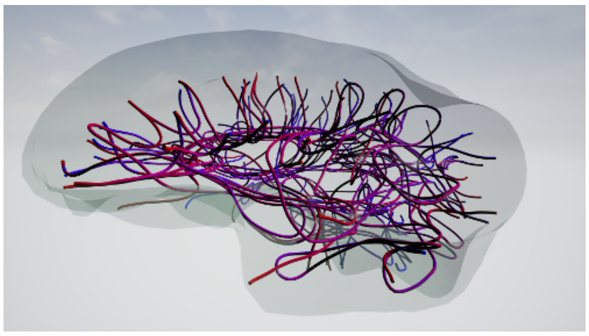
Figure 21: The color picker is opened when an user wants to change a color from the main menu. The sliders can be adjusted when a grab gesture of the right hand is performed.

3.3. Brain and fibres

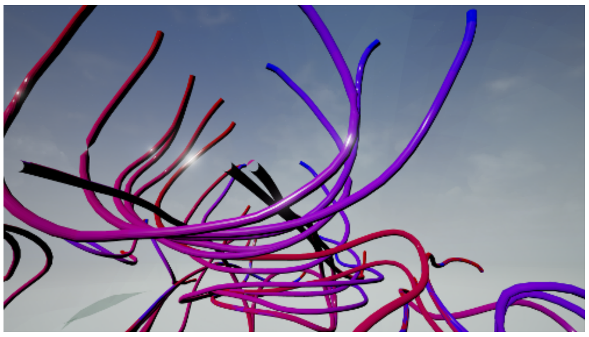
It is the same dataset of fMRI data at any subfigure of 22 but with different colors, amount of fibers and viewing positions. Figure 22a shows the brain with 100 green and ocher fibers from behind the brain. The second picture (figure 22b) shows the same configuration but with red and blue fibers. Figure 22c shows the same configuarion as in figure 22b but from the inside. It is clear to see that the start color of the fibers is red and the end color blue. The last screenshot (figure 22d) represent the configuration with 1000 fibers.



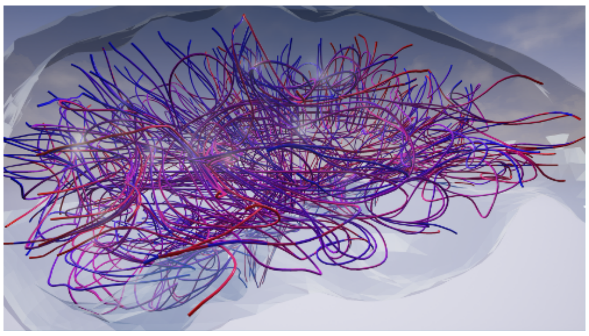
(a) 100 fibers



(b) 100 fibers



(c) 100 fibers from inside the brain.



(d) 1000 fibers

Figure 22: Some pictures of different colors, viewing positions and amount of fibers.

3.4. Performance

All following diagrams were measured on an Intel i7-5820k, 16GB DDR4 RAM and a Nvidia GTX 980 Ti. The used Oculus Rift was a Development Kit 2 and the differences of the headsets are described in the chapter 4.3.1.

Figure 23 symbols the relationship between amount of fibers and frames per second. The interpolation curve describes a rational function.

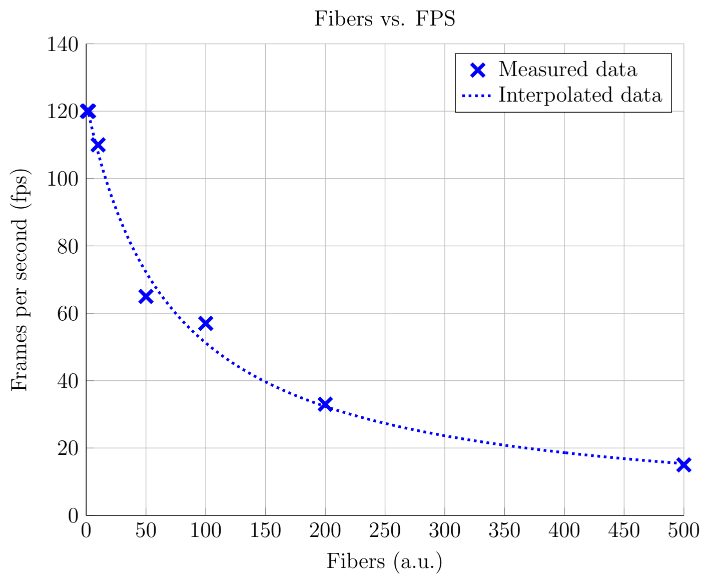


Figure 23: Relation between amount of fibers and frames per second.

Figure 24 describes the usage of the CPU in % at different amounts of fibers. All measured data characterize a logistic function.

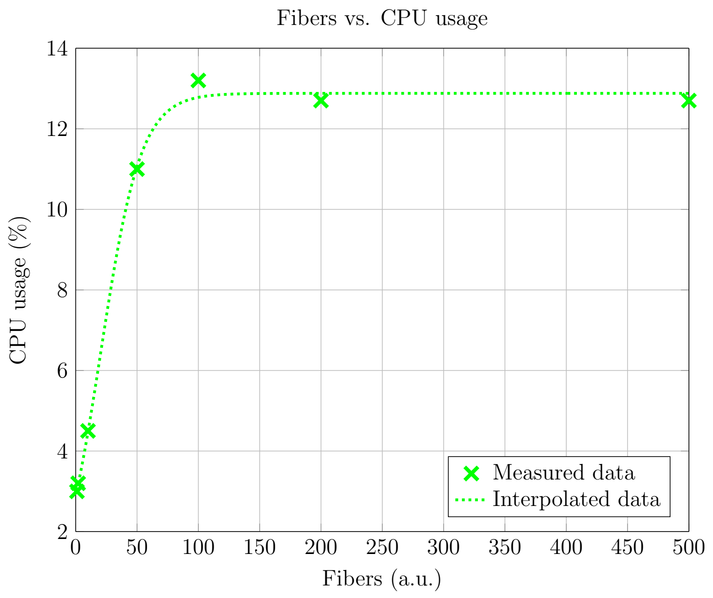


Figure 24: Relation between amount of fibers and usage of the CPU.

At last the dependency of the RAM usage is painted in figure 25. The best fitting curve is a function to the power of three.

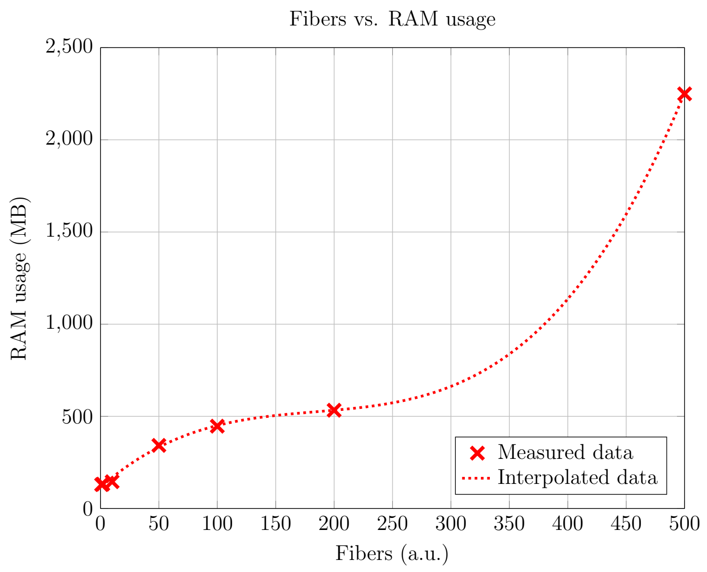


Figure 25: Relation between amount of fibers and usage of the RAM.

4. Discussion

4.1. DSI Studio

To create fibers out of DICOM files there are different software packages like Freesurfer [5] available on the market but only DSI Studio could read out the b values correctly. A b value which is used in equation 4 is a combination of gradient strength and pulse duration.

The next advantages are the graphical user interface and the good usability. It is also the case that DSI Studio is a free software and it is compatible with Windows. The manual [4] explains the analyzing steps really simple. The only disadvantage is that the calculated brain volume is only exportable as volume data. For triangulation which is necessary for rendering the brain in the Unreal Engine the data have to be surface data.

4.1.1. Surface data

Because of the reason from above the volume data have to be converted to surface data. This conversion takes about 15 minutes using 8 parallel threads on the CPU. So it is a good idea to run the program on the graphics card. Via CUDA (Compute Unified Device Architecture) it is possible to reduce the calculation time to under a minute with same results.

4.2. MeshLab

To do the triangulation it is necessary to use MeshLab. The reasons to use MeshLab are on the one hand it delivers tested and robust results. The next is that the software has got many different adjustable parameters to improve the outcome and it saves the data to `.obj` files. These files include the pixel data of the surface as well as the normals and the indexes of the triangles. All of these parameters are required to render 3D objects in the Unreal Engine.

[4] `http://dsi-studio.labsolver.org/Manual`

4.3. Unreal Engine

The Unreal Engine delivers many adjustable parameters and free plugins. Furthermore the documentation is very detailed and helps a beginner to learn the characteristics of a gaming engine.

By comparison to Unity (Unity Technologies, San Francisco, California, U.S.) which is also a popular free gaming engine Unreal Engine features a better integration of the Oculus Rift and the Leap Motion. It is also the case that the graphical programming interface Blueprint lightens the beginning of programming. Moreover Unity is only program in C# or JavaScript thus code based development. The product of Epic games is more powerful and by the use of C++ it is more adjustable.

The Unity engine does not support the integration of VR headsets and the Leap Motion at the beginning of the project and so it was obviously to use the Unreal Engine 4. Unfortunately the Unreal Engine has some special characteristics and these lead to different problems during implementation. The next paragraphs will discuss the problems.

4.3.1. Oculus Rift integration

It is very easy to include the Oculus Rift into Unreal Engine 4 and all versions of the VR headset are compatible. On the other side all versions are slightly different. After developing with the DK2 and testing with it. It turned out that not all functions work the same on DK1 and DK2.

The first difference affects the center of the camera. The 2 VR headsets got 2 different middle points and so there is no general solution to find this point. It is also the case that the firmware 1.0 and 0.8 beta are different. The firmwares calculate the normal on the camera differently and they also have various minimum system requirements.

Through the miss of the motion tracking at the DK1 the control and the game design is a disparate to the DK2. For an example the movement via the motion tracking in the 3D space is more intuitive and has to be adjusted for the DK1.

The last difference is the resolution and the framerate of the two headsets. DK1 only uses a resolution of 1280x720 and the DK2 1920x1080. This means it is necessary to optimize the program to 720p resolution and not to 1080p. On the one hand it raises the performance but on the other hand it lowers the field of view. The bottom line is that the reduction of the resolution reduces the usability.

At the end it is important to mention that the project runs with both versions and it is a matter of taste which headset delivers better results. The DK1 has a poorer resolution and the movement is less intuitive but on the other hand the jitter effect (chapter 4.4.1) does not occur.

4.3.2. Leap Motion integration

The integration of the Leap Motion is more complicated than the implementation of the Oculus Rift. There is a whole development kit of the functions for the Leap Motion but only for C++ or C#. The only way to use it in the graphical development environment is to write own blueprint functions and events.

The next big problem is the recognizability of the gesture. All in all the false positive and false negative rate is really high and make a handling exhausting. It is also the case that the reaction time is to high to use it in real time and so the feedback is very laggy.

Finally the amount of gestures is manageable and so it is hard to find different gestures for different actions. False recognized gestures make the control also worse. For an example the software could hardly differ between a grab and a pinch or between a screen tap and a key tap. It is not recommended to use a screen tap and a key tap gesture for different actions. The only solution would be to learn new gestures and overwrite the firmware. This was too much work for this project and so it was necessary to use the standard gestures.

4.3.3. Menus and mouse pointer

There are two reasons why every 2D menu has to be included into a 3D object. The first one is that if the menu is a 2D object the Oculus has problems to render it. The object smears over the display and the user could not see anything. The second reason is the that the 3D mouse pointer needs a 3D object to interact with it.

If the mouse pointer is just the normal 2D mouse pointer it will also be rendered only on one of the two displays of the Oculus Rift. The pointer interacts with the Oculus Rift like with 2 normal PC monitors which are extended. So the user would move the pointer from one to the other display but without any connection between the location of the pointer which is rendered on the headset and the location at the virtual space of the program.

It is also the case that the depth information is not available using a 2D pointer. If the menu is not in the same depth as the pointer the usability of the pointer will not be given. To create a ray cast like described in section 2.6 the object has to be a 3D object.

4.3.4. Color Picker and file handling

Another challenge is to implement a color picker because the picker has to be controllable using the Leap Motion. So the design has to be as simple as possible. This is also the reason why the project has a start up program. The Unreal Engine project is a standalone software but it has no plugin for a file opener. It is also not possible to create a file opener which is usable via gestures because the interface would be to filigree to use it with the Leap Motion.

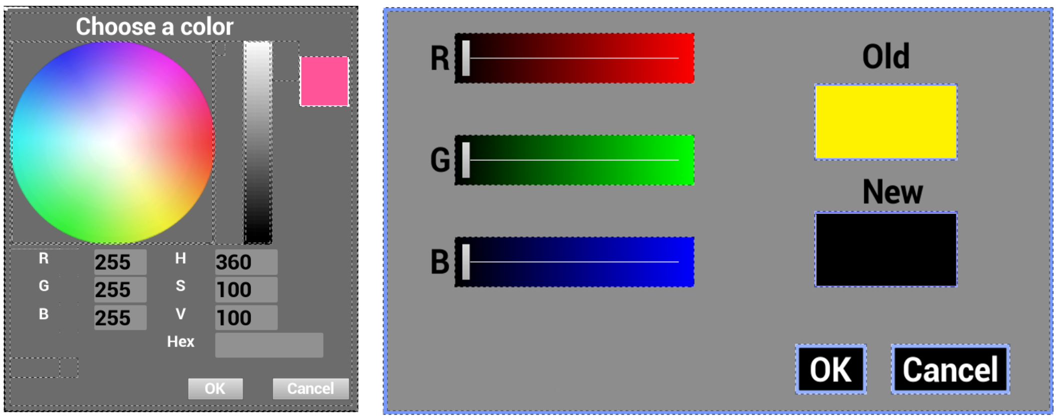


Figure 26: The left picture shows the first design and the right one the final design of the color picker.

Figure 26 shows two different designs of the color picker. At the left one all numeric values are changeable and the user could choose a color by clicking into the circle or changing the values. This design is very easy if the user has a mouse and a keyboard but it will be impossible to use via the Leap Motion. This is one example how the input device modifies the design of the software package.

4.4. Artifacts

4.4.1. Jitter

If the amount of fibers is to high the framerate will drop and jitters will appear. The reason is that the firmware of the Oculus Rift tries to interpolate the position of the objects during a change of the viewing angle. It is the case that the Oculus runs on a static value of FPS for an example at the DK2 it is 75Hz to reduce flicker artifacts. If the change appears but the program delivers a framerate which is lower than the framerate of the headset it will calculate the new position of the object and will render it. This would be nice if the end position of the object is the same as the calculated but if the positions are not the same the object will jump from the calculation location to the real location.

Figure 27 symbols the jitter effect at 3 frames. The unwanted movement to the right and then to the left is the jitter effect. This phenomena appears more frequent when the amount of fibers is too high for the performance of the PC and the native framerate of the headset is higher. So the jitter effect appears more likely on the DK2 as on the DK1.

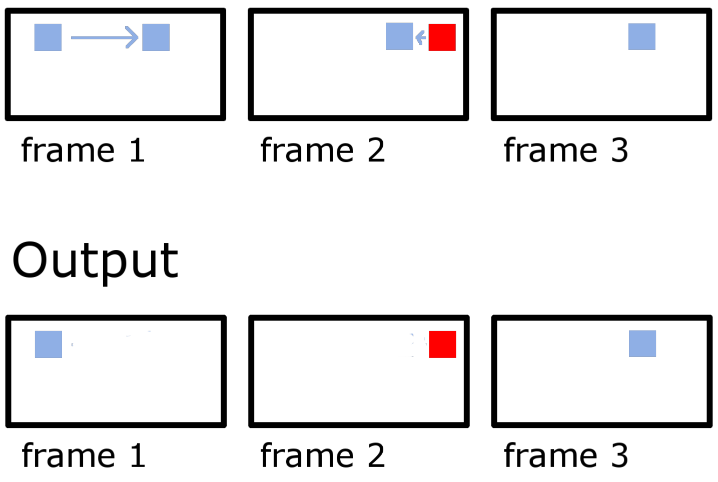


Figure 27: The upper series of frames describes the calculation frames and the bottom frames are the output frames. In the first frame the blue object will move to the right. It is rendered at the old position. Frame 2 is the calculated frame of the Oculus. The framerate is lower than the native framerate of the Oculus. The headset calculates the position of the object (red square) and this object is rendered. The original location is the blue square and the object has to move to the left in the next frame. At frame 3 the original position is rendered on the headset.

4.4.2. Overlaying fibers

The fiber in the red circle of figure 28 should be yellow but it has violet segments. At the same position as the yellow there is a violet fiber but this one is little bit closer to the camera and so the engine renders the violet fiber. For the interpretation of the fibers these artifacts do not matter and so it is not so important to solve this problem at this time.

A solution would be an algorithm which looks at the neighbor segment and color it the same way. The algorithm would use the neighbor and would decide which segment of the two is more likely the same color as the neighbor.

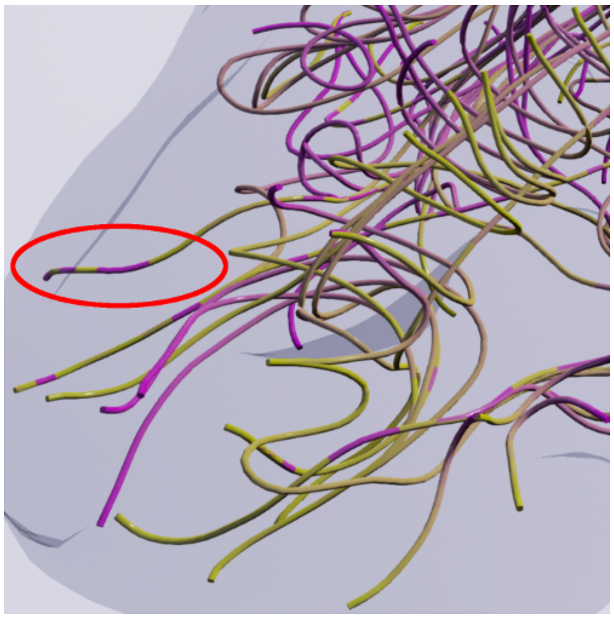


Figure 28: The whole fiber under the red circle should be yellow and not include violet segments.

4.5. Performance

The performance is indirect proportional to the count of fibers more fibers lead to poorer performance every fiber is a single object with single segments and every segment symbols a mesh which is rendered separately at runtime. It is also the case that all objects are spawned at runtime and the are dynamic. This means the objects are not pre rendered and so the light, colors and so on are calculated at runtime.

The reason behind this solution is that the program can be extended easily. For an example the program can be extended so that the user can select single fibers to manipulate them or to choose a typically area at runtime to render just this area.

4.5.1. FPS

The frames per seconds can drop under 30 and the user feels it laggy. The development system was a high end PC but the framerate dropped to about 30fps by 500 fibers and to 50-60fps by 100 fibers as it can be seen in figure 23. Normally 50-60fps would be no problem for the user but through the jitter effects the frame drop is noticeable.

The reduction of quality or the resolution does not affect the FPS. A solution would be the limitation of the field of view or the render distance. Next it would be a good idea to not paint objects which are hidden behind other objects. The cylindrical shape which is used to render the fibers could also be optimized. The count of facets could be decremented without a quality lost. Finally a big optimization of the quality settings could also improve the count of the FPS.

4.5.2. CPU and RAM

The usage of the CPU as seen in picture 24 describes a logistic function and runs to a maximum value of about 13%. This limit is not the maximum usage of one kernel of the PC and this means that the CPU does not limit the program. The whole program uses all kernels and distribute it over all equally. That means a more powerful CPU does not increase the FPS count.

Figure 25 shows the trend of the RAM usage over the amount of fibers. The fitting curve describes a function to the power of 3. This result satisfies the calculated memory requirements of $\mathcal{O}(n^3)$.

4.6. Conclusion

The software solution delivers robust 3D data of the DTI fibers and is simple to use. All parameters are intuitive to change and it gives the user a good view inside the brain. The user gets a overview and can detect failures in the fibers.

Figure 22 shows four adjustments and its obvious to see that the user can get a feeling for the fibers. This means the user get an overview where a single fiber starts and where its ends because of the different colors. The next point is that the software gives an operator the feeling of being inside of the brain and it also delivers a good overview of the brain when zoomed out. All color properties are saved and this contributes a lot to the usability.

The disadvantages are the artifacts and that the program is very computationally intensive. So it is just usable on a highend PC and this means the hardware is expensive. Another problem is the loss of orientation of the user and the missing connectivity between the fibers and the different parts of the cortex.

Concluding the thesis contributes an easily extendable platform for visualizing diffusion tensor images and can be executed on standard PCs without any special software packages.

4.7. Further research

The first point is to improve the performance of the software, so the jitter and stutter effect will not occur. This could be done by different quality settings of the engine or outsourcing of calculations on a second graphics card for example.

The second step could be implement an atlas to get the connections between the fibers and the cortex areas. This should be simple because every fiber is a single object and so it would be possible to allocate these to the atlas. Having done that the user could decide which area is important and turn off the others.

Thirdly an improvement of the gesture control could be done. One option would be learning new gestures to the Leap Motion or using another gesture controller like the Microsoft Kinect.

The last idea would be implement an orientation system which guides the user through the brain. So the user would not loose the orientation.

References

[1] Julio Acosta-Cabronero and Peter J Nestor. Diffusion tensor imaging in alzheimer's disease: insights into the limbic-diencephalic network and methodological considerations. *Frontiers in aging neuroscience*, 6:266, 2014.

[2] Ferran Prados Carrasco. Visualization and processing of diffusion tensor mri. *Universitat de Girona*, 2012.

[3] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. Meshlab: an open-source mesh processing tool. 2008.

[4] Olivier Commowick, Pierre Fillard, Olivier Clatz, and Simon K Warfield. Detection of dti white matter abnormalities in multiple sclerosis patients. *Medical image computing and computer-assisted intervention : MICCAI ... International Conference on Medical Image Computing and Computer-Assisted Intervention*, 11(Pt 1):975–82, 2008.

[5] Bruce Fischl. Freesurfer. 2012.

[6] Epic Games. Unreal engine 4 official website, 2016. `https://www.unrealengine.com/`.

[7] Derek Jones. *Diffusion MRI : theory, methods, and application.* Oxford University Press, Oxford New York, 2010.

[8] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. 2008.

[9] Leap Motion. Leap motion official website, 2016. `https://www.leapmotion.com/`.

[10] National Electrical Manufacturers Association (NEMA). Dicom, digital imaging and communications in medicine, 2016. `http://medical.nema.org/standard.html`.

[11] Nvidia. Cuda project official website, 2016. `http://www.nvidia.com/object/cuda_home_new.html`.

[12] Kenichi Oishi, Michelle M Mielke, Marilyn Albert, Constantine G Lyketsos, and Susumu Mori. Dti analyses and clinical applications in alzheimer's disease. *Journal of Alzheimer's disease : JAD*, 26 Suppl 3:287–96, 2011.

[13] Keith Rule. *3D Graphics File Formats: A Programmer's Reference.* Addison-Wesley (C), 1996.

[14] Emilia Sbardella, Francesca Tona, Nikolaos Petsas, and Patrizia Pantano. Dti measurements in multiple sclerosis: Evaluation of brain damage and clinical implications. *Multiple sclerosis international*, 2013:671730, 2013.

[15] Brenden Sewell. *Blueprints Visual Scripting for Unreal Engine : build professional 3D games with Unreal Engine 4's Visual Scripting system.* Packt Publishing, Birmingham, UK, 2015.

[16] William Sherif. *Learning C++ by creating games with UE4 : learn C++ programming with a fun, real-world application that allows you to create your own games.* Packt Publishing, Birmingham, UK, 2015.

[17] Jie Song, Brittany M Young, Zack Nigogosyan, Leo M Walton, Veena A Nair, Scott W Grogan, Mitchell E Tyler, Dorothy Farrar-Edwards, Kristin E Caldera, Justin A Sattin, Justin C Williams, and Vivek Prabhakaran. Characterizing relationships of dti, fmri, and motor recovery in stroke rehabilitation utilizing brain-computer interface technology. *Frontiers in neuroengineering*, 7:31, 2014.

[18] E.O. Stejskal and J.E. Tanner. Spin diffusion measurements: Spin echoes in the presence of a time-dependent field gradient. *The Journal of Chemical Physics*, 42(1):288–292, 1965.

[19] Oculus VR. Oculus rift official website, 2016. `https://www.oculus.com/`.

[20] Wavefront. Obj definition, 2016. `http://www.martinreddy.net/gfx/3d/OBJ.spec`.

[21] D J Werring, A T Toosy, C A Clark, G J Parker, G J Barker, D H Miller, and A J Thompson. Diffusion tensor imaging can detect and quantify corticospinal tract degeneration after stroke. *Journal of neurology, neurosurgery, and psychiatry*, 69(2):269–72, Aug 2000.

[22] Nicholas Wilt. *A Comprehensive Guide to GPU programming.* Pearson Education, 2013.

[23] Fang-Cheng Yeh, Timothy D. Verstynen, Yibao Wang, Juan C. Fernández-Miranda, and Wen-Yih Isaac Tseng. Deterministic diffusion fiber tracking improved by quantitative anisotropy. 2013.

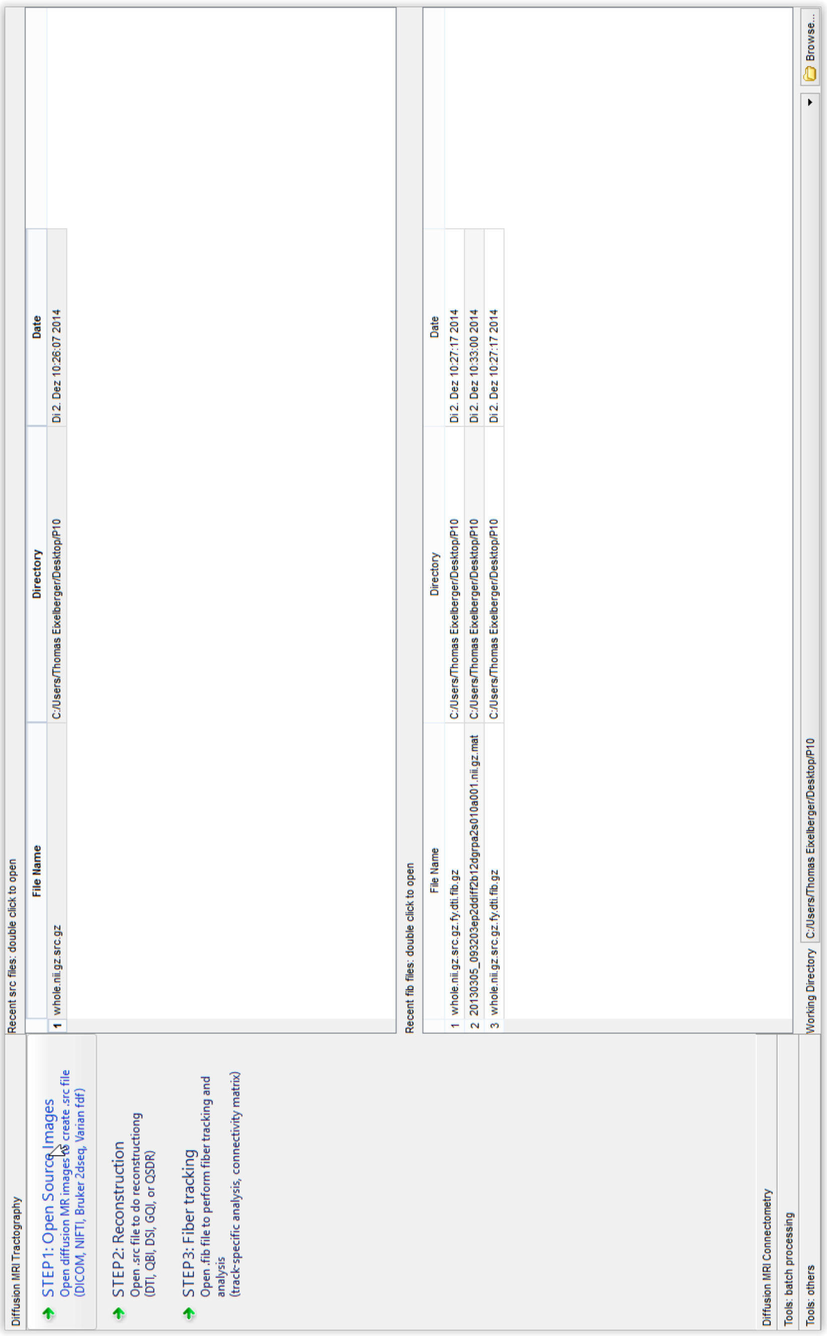
[24] Fang-Cheng Yeh, Van Jay Wedeen, and Wen-Yih Isaac Tseng. Generalized -sampling imaging. 2010.

A. Manual

A.1. DSI Studio

It can be downloaded from the official website [5].

1. Click on the `Open Source Images` button to open the desired DICOM files.



Diffusion MRI Tractography

STEP1: Open Source Images
Open diffusion MR images to create .src file
(DICOM, NIFTI, Bruker 2dseq, Varian fdf)

STEP2: Reconstruction
Open .src file to do reconstructiong
(DTI, QBI, DSI, GQI, or QSDR)

STEP3: Fiber tracking
Open .fib file to perform fiber tracking and analysis
(track-specific analysis, connectivity matrix)

Diffusion MRI Connectometry

Tools: batch processing

Tools: others

Recent src files: double click to open

	File Name	Directory	Date
1	whole.nii.gz.src.gz	C:/Users/Thomas Exelberger/Desktop/P10	Di 2. Dez 10:26:07 2014

Recent fib files: double click to open

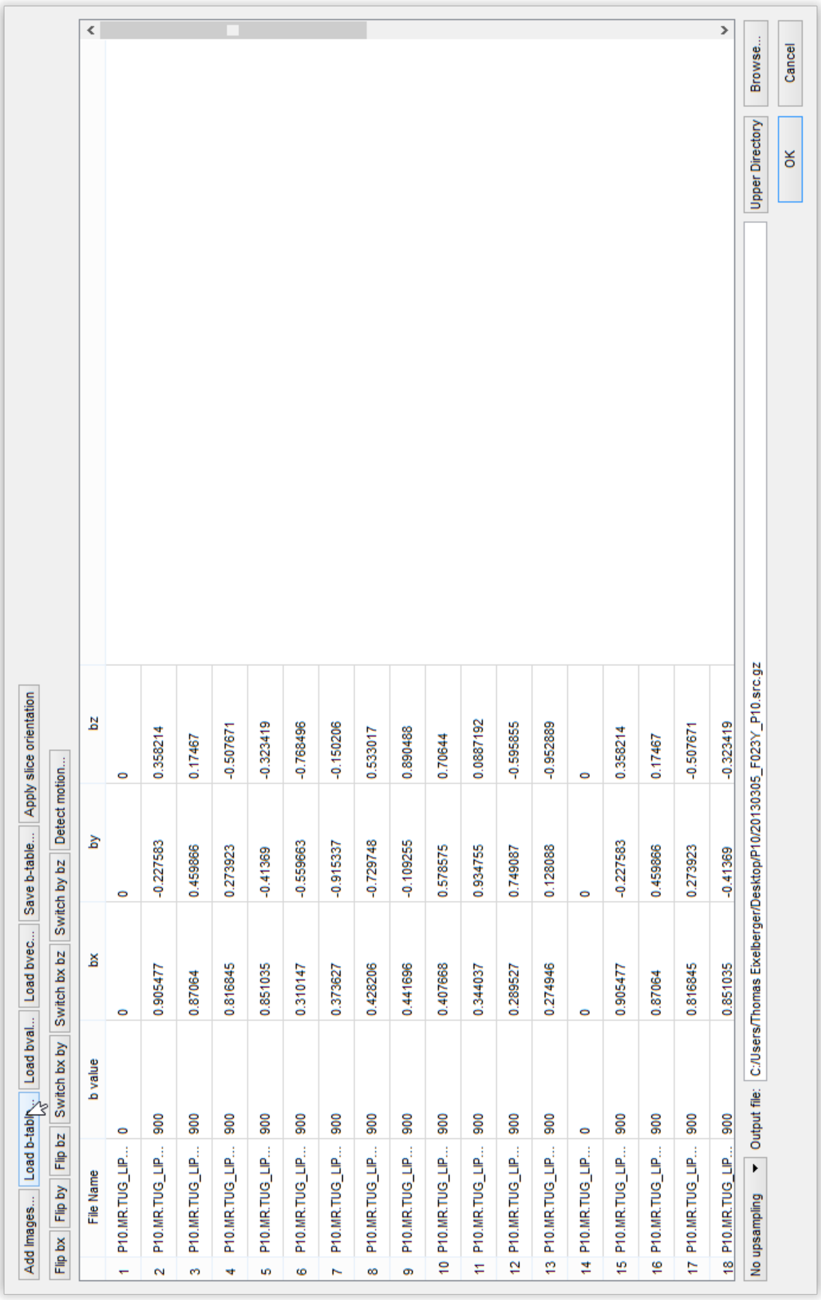
	File Name	Directory	Date
1	whole.nii.gz.src.gz.fy.dti.fib.gz	C:/Users/Thomas Exelberger/Desktop/P10	Di 2. Dez 10:27:17 2014
2	20130305_093203ep2d diff2b12dgrpa2s010a001.nii.gz.mat	C:/Users/Thomas Exelberger/Desktop/P10	Di 2. Dez 10:33:00 2014
3	whole.nii.gz.src.gz.fy.dti.fib.gz	C:/Users/Thomas Exelberger/Desktop/P10	Di 2. Dez 10:27:17 2014

Working Directory C:/Users/Thomas Exelberger/Desktop/P10

Figure 29: First step: open the DICOM files.

[5] `http://dsi-studio.labsolver.org/`

2. Check the btable data and if necessary load an external btable.

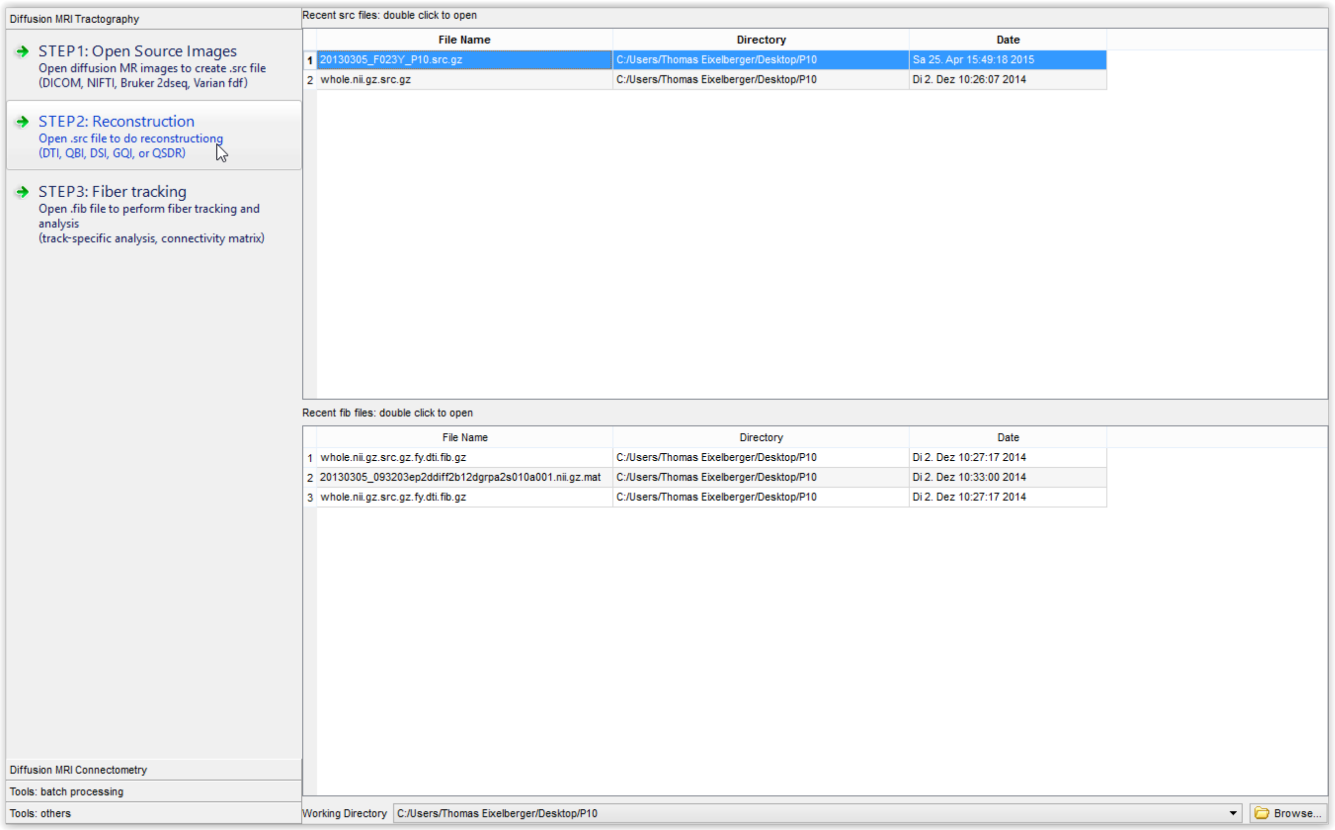


	File Name	b value	bx	by	bz
1	P10.MR.TUG_LIP...	0	0	0	0
2	P10.MR.TUG_LIP...	900	0.905477	-0.227583	0.358214
3	P10.MR.TUG_LIP...	900	0.87064	0.459866	0.17467
4	P10.MR.TUG_LIP...	900	0.816845	0.273923	-0.507671
5	P10.MR.TUG_LIP...	900	0.851035	-0.41369	-0.323419
6	P10.MR.TUG_LIP...	900	0.310147	-0.559663	-0.768496
7	P10.MR.TUG_LIP...	900	0.373627	-0.915337	-0.150206
8	P10.MR.TUG_LIP...	900	0.428206	-0.729748	0.533017
9	P10.MR.TUG_LIP...	900	0.441696	-0.109255	0.890488
10	P10.MR.TUG_LIP...	900	0.407668	0.578575	0.70644
11	P10.MR.TUG_LIP...	900	0.344037	0.934755	0.0887192
12	P10.MR.TUG_LIP...	900	0.289527	0.749087	-0.595855
13	P10.MR.TUG_LIP...	900	0.274946	0.128088	-0.952889
14	P10.MR.TUG_LIP...	0	0	0	0
15	P10.MR.TUG_LIP...	900	0.905477	-0.227583	0.358214
16	P10.MR.TUG_LIP...	900	0.87064	0.459866	0.17467
17	P10.MR.TUG_LIP...	900	0.816845	0.273923	-0.507671
18	P10.MR.TUG_LIP...	900	0.851035	-0.41369	-0.323419

Output file: C:/Users/Thomas Eixelberger/Desktop/P10/20130305_F023Y_P10.src.gz

Figure 30: Second step: Check the btable.

3. Press the `Reconstruction` button and select the earlier created src-file.



Diffusion MRI Tractography

STEP1: Open Source Images
Open diffusion MR images to create .src file
(DICOM, NIFTI, Bruker 2dseq, Varian fdf)

STEP2: Reconstruction
Open .src file to do reconstructiong
(DTI, QBI, DSI, GQI, or QSDR)

STEP3: Fiber tracking
Open .fib file to perform fiber tracking and analysis
(track-specific analysis, connectivity matrix)

Diffusion MRI Connectometry

Tools: batch processing

Tools: others

Recent src files: double click to open

	File Name	Directory	Date
1	20130305_F023Y_P10.src.gz	C:/Users/Thomas Eixelberger/Desktop/P10	Sa 25. Apr 15:49:18 2015
2	whole.nii.gz.src.gz	C:/Users/Thomas Eixelberger/Desktop/P10	Di 2. Dez 10:26:07 2014

Recent fib files: double click to open

	File Name	Directory	Date
1	whole.nii.gz.src.gz.fy.dti.fib.gz	C:/Users/Thomas Eixelberger/Desktop/P10	Di 2. Dez 10:27:17 2014
2	20130305_093203ep2ddiff2b12dgrpa2s010a001.nii.gz.mat	C:/Users/Thomas Eixelberger/Desktop/P10	Di 2. Dez 10:33:00 2014
3	whole.nii.gz.src.gz.fy.dti.fib.gz	C:/Users/Thomas Eixelberger/Desktop/P10	Di 2. Dez 10:27:17 2014

Working Directory C:/Users/Thomas Eixelberger/Desktop/P10

Figure 31: Third step: Check the btable.

4. Adjust the threshold, dilation and so on when necessary.

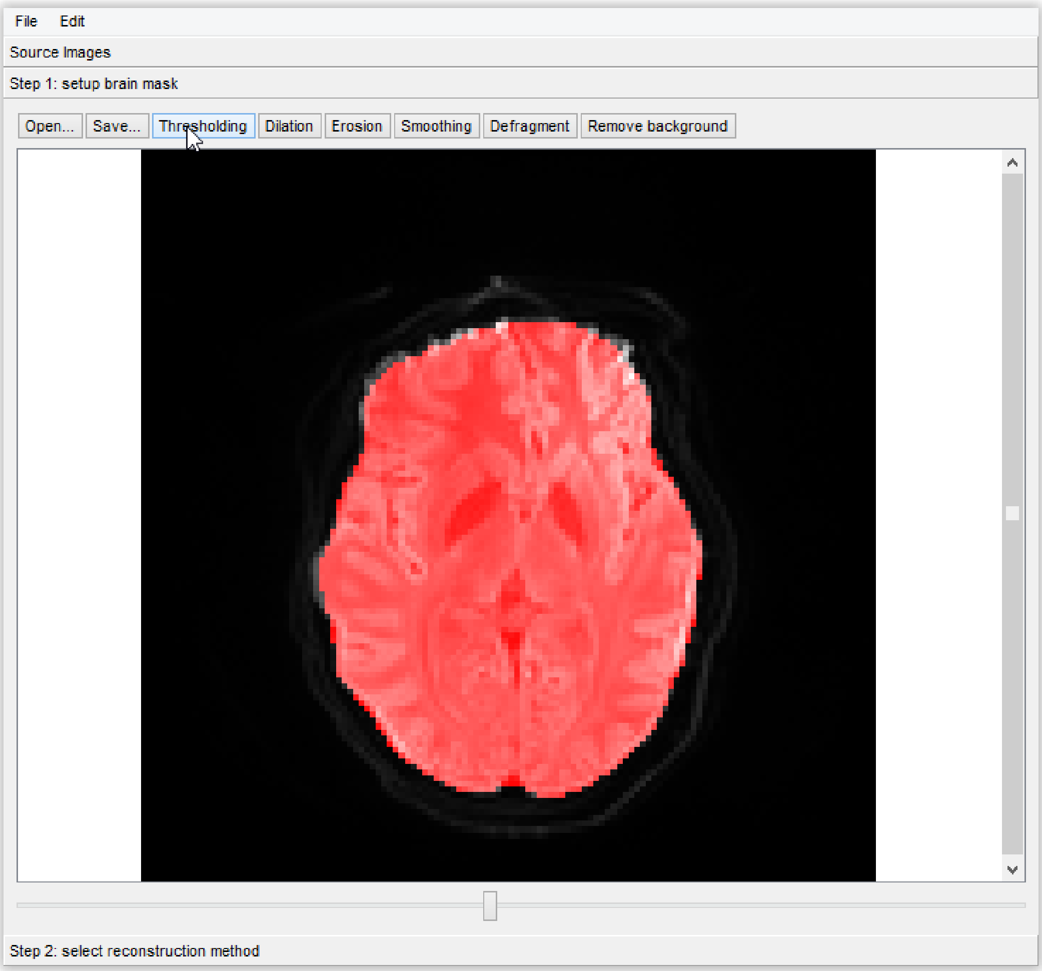
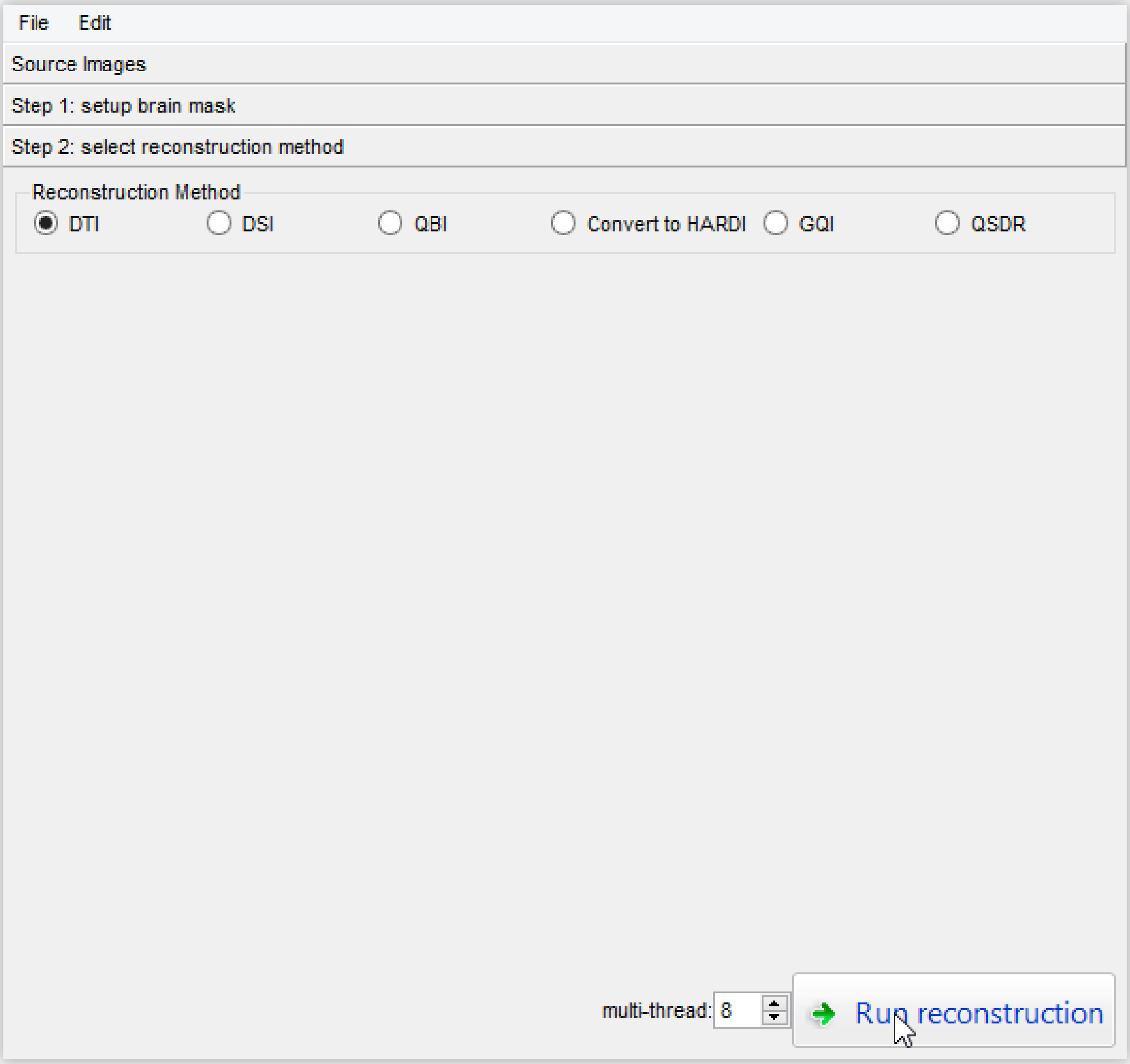


Figure 32: Fourth step: adjust different values.

5. Click on `select reconstruction method` and select `DTI`. After that press `Run reconstruction`.

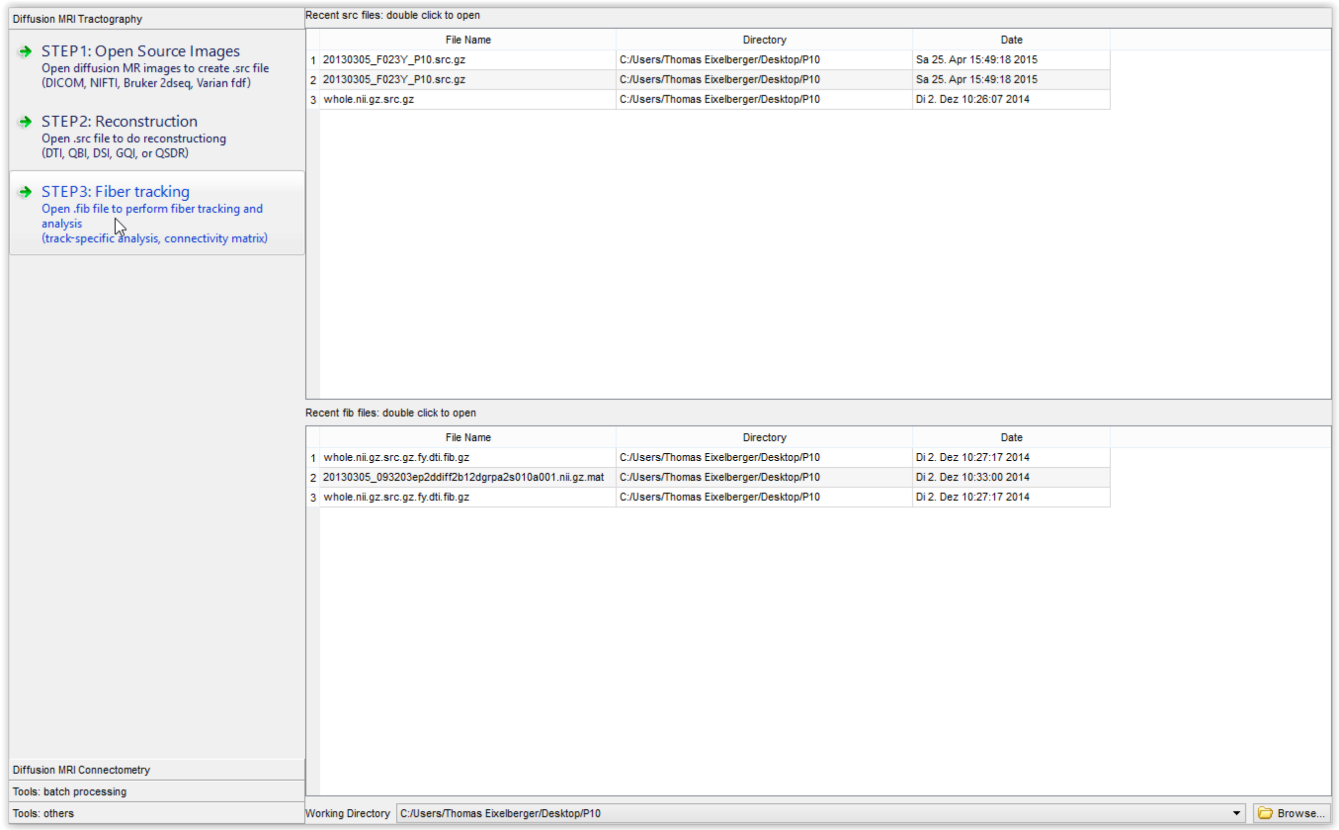


Reconstruction Method: [✓ DTI] DSI QBI Convert to HARDI GQI QSDR

multi-thread: 8

Figure 33: Fifth step: run reconstruction.

6. A fib file is created and now press on the `Fiber tracking` button. Select the created file.


Diffusion MRI Tractography
STEP1: Open Source Images
Open diffusion MR images to create .src file
(DICOM, NIFTI, Bruker 2dseq, Varian fdf)
STEP2: Reconstruction
Open .src file to do reconstructiong
(DTI, QBI, DSI, GQI, or QSDR)
STEP3: Fiber tracking
Open .fib file to perform fiber tracking and analysis
(track-specific analysis, connectivity matrix)
Diffusion MRI Connectometry
Tools: batch processing
Tools: others
Browse...

Recent src files: double click to open

	File Name	Directory	Date
1	20130305_F023Y_P10.src.gz	C:/Users/Thomas Eixelberger/Desktop/P10	Sa 25. Apr 15:49:18 2015
2	20130305_F023Y_P10.src.gz	C:/Users/Thomas Eixelberger/Desktop/P10	Sa 25. Apr 15:49:18 2015
3	whole.nii.gz.src.gz	C:/Users/Thomas Eixelberger/Desktop/P10	Di 2. Dez 10:26:07 2014

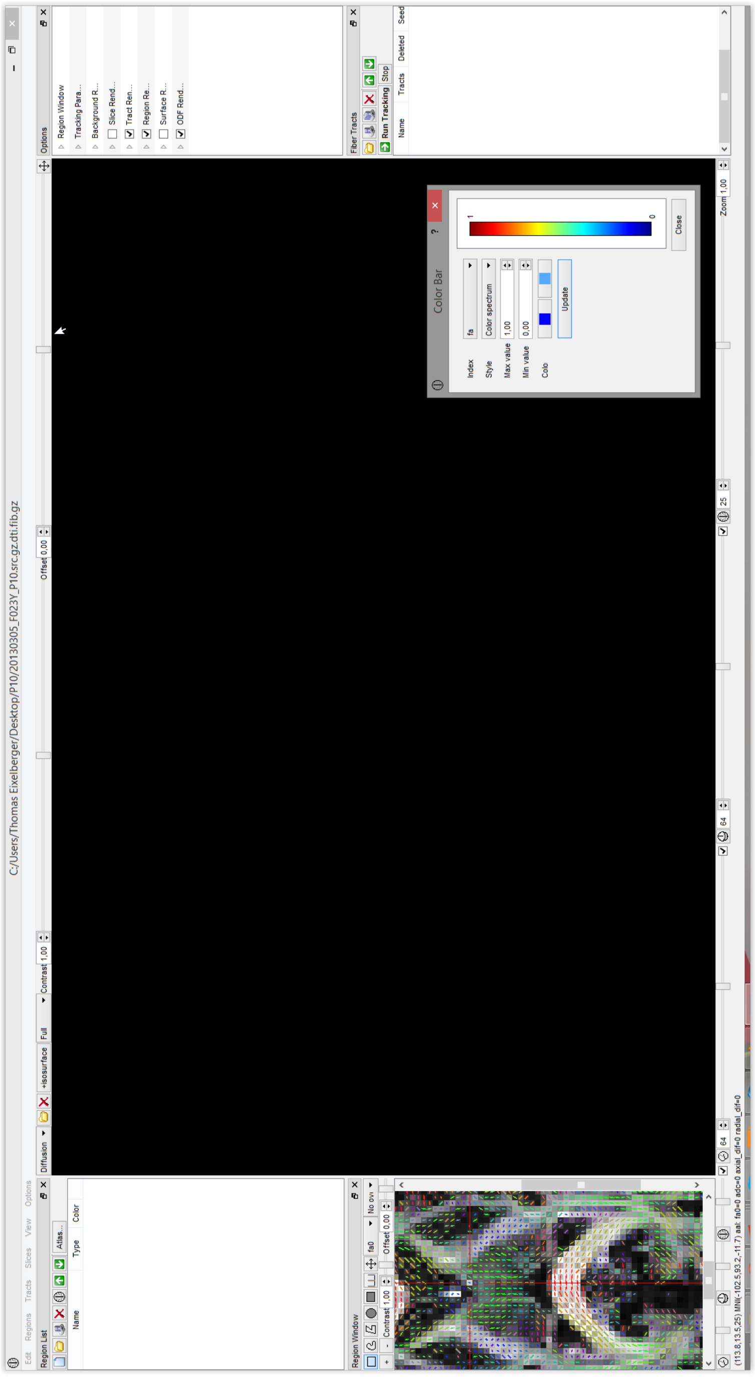
Recent fib files: double click to open

	File Name	Directory	Date
1	whole.nii.gz.src.gz.fy.dti.fib.gz	C:/Users/Thomas Eixelberger/Desktop/P10	Di 2. Dez 10:27:17 2014
2	20130305_093203ep2ddiff2b12dgrpa2s010a001.nii.gz.mat	C:/Users/Thomas Eixelberger/Desktop/P10	Di 2. Dez 10:33:00 2014
3	whole.nii.gz.src.gz.fy.dti.fib.gz	C:/Users/Thomas Eixelberger/Desktop/P10	Di 2. Dez 10:27:17 2014

Working Directory: C:/Users/Thomas Eixelberger/Desktop/P10

Figure 34: Sixth step: open the fib file.

7. Adjust the fiber tracking values on the right side of the window 35 like number of fibers.



C:/Users/Thomas Eixelberger/Desktop/P10/20130305_F023Y_P10.src.gz.dti.fib.gz

Index fa
Style Color spectrum
Max value 1,00
Min value 0,00

Figure 35: Seventh step: adjust parameters.

8. Run the tracking by clicking on `Run tracking` at the bottom right hand corner.

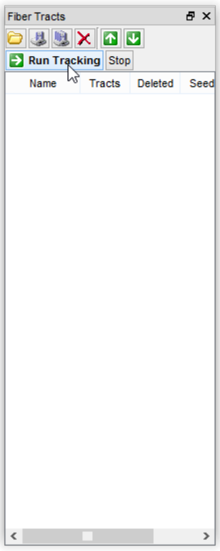


Figure 36: Eight step: run tracking.

9. Generate the brain volume by clicking at the top left hand corner on the brain icon.

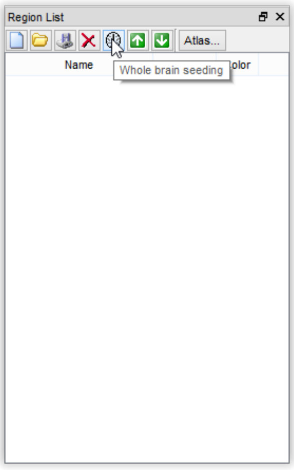
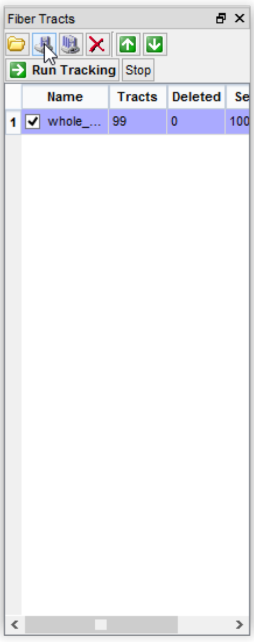
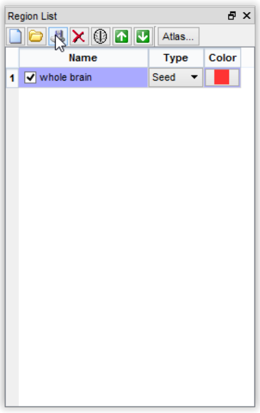


Figure 37: Ninth step: generate the brain volume.

10. Save the data if the reconstruction looks fine by clicking on the save buttons. Save the data always as txt-files.



(a) Save fibers



(b) Save brain

Figure 38: Tenth step: save the generated data.

A.2. Neighbourhood

To convert the brain voxel to surface data start the console program `CUDA_Neighbor.exe` with the following syntax.

CUDA_Neighbor.exe -i [input file] **-o** [output file]

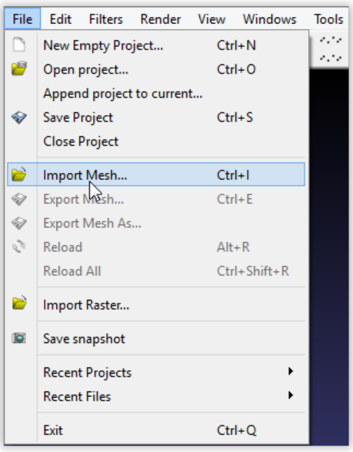
-i [input file] represents the absolute or relative path to the brain voxel file

-o [output file] represents the absolute or relative path to the surface file

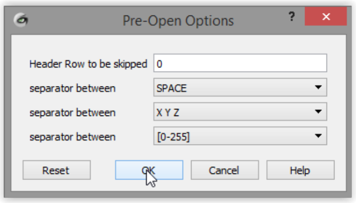
```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\CUDA_Neighbor.exe -i whole_brain.txt -o surface.txt
Device Number: 0
  Device name: GeForce GT 750M
  Memory Clock Rate (KHz): 2500000
  Memory Bus Width (bits): 128
  Peak Memory Bandwidth (GB/s): 80.000000

Reading data...
Reading done!
Calculating surface...
100 % [==================================================]
Calculating done!
Writing output file...
Writing output file done!
41447 points in 75.098 seconds
```

A.3. MeshLab

1. Load the surface data by clicking on `File->Import Mesh` and change the separator to `SPACE`.



(a) Import the data.



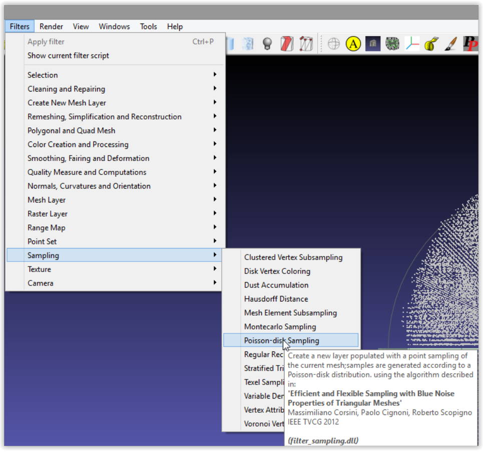
Pre-Open Options

Header Row to be skipped	0
separator between	SPACE
separator between	X Y Z
separator between	[0-255]

(b) Change to SPACE.

Figure 39: First step: import the surface data.

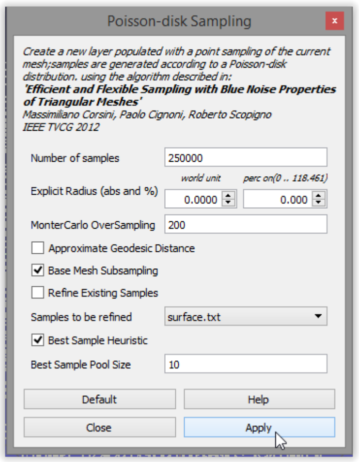
2. Click on `Filters->Sampling->Poisson-disk Sampling` to upsample the data. A good value for the `samples` is about **250000** and for `MonteCarlo OverSampling` **200**. The item `Base Mesh Subsampling` has to be checked.



Create a new layer populated with a point sampling of the current mesh;samples are generated according to a Poisson-disk distribution. using the algorithm described in:
'Efficient and Flexible Sampling with Blue Noise Properties of Triangular Meshes'
Massimiliano Corsini, Paolo Cignoni, Roberto Scopigno
IEEE TVCG 2012

(filter_sampling.dll)

(a) Choose disk sampling.



Poisson-disk Sampling

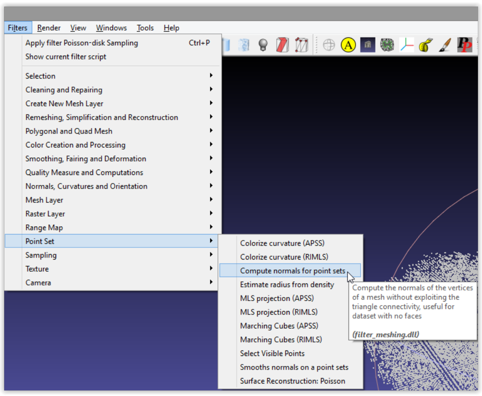
Create a new layer populated with a point sampling of the current mesh;samples are generated according to a Poisson-disk distribution. using the algorithm described in:
'Efficient and Flexible Sampling with Blue Noise Properties of Triangular Meshes'
Massimiliano Corsini, Paolo Cignoni, Roberto Scopigno
IEEE TVCG 2012

Number of samples	250000	
	world unit	*perc on(0 .. 118.461)*
Explicit Radius (abs and %)	0.0000	0.000
MonterCarlo OverSampling	200	
☐ Approximate Geodesic Distance		
☑ Base Mesh Subsampling		
☐ Refine Existing Samples		
Samples to be refined	surface.txt	
☑ Best Sample Heuristic		
Best Sample Pool Size	10	

(b) Change the parameters like above.

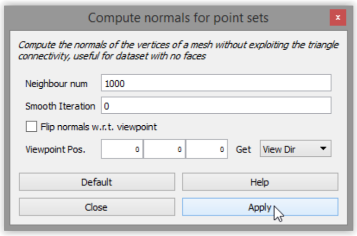
Figure 40: Second step: perform an upsampling.

3. Calculate the normals by clicking `Filters->Point Set->Compute normals for point sets`. Change `Neighbour num` to **1000**.


Filters
Render
View
Windows
Tools
Help
Apply filter Poisson-disk Sampling Ctrl+P
Show current filter script
Selection
Cleaning and Repairing
Create New Mesh Layer
Remeshing, Simplification and Reconstruction
Polygonal and Quad Mesh
Color Creation and Processing
Smoothing, Fairing and Deformation
Quality Measure and Computations
Normals, Curvatures and Orientation
Mesh Layer
Raster Layer
Range Map
Point Set
Sampling
Texture
Camera
Colorize curvature (APSS)
Colorize curvature (RIMLS)
Compute normals for point sets
Estimate radius from density
MLS projection (APSS)
MLS projection (RIMLS)
Marching Cubes (APSS)
Marching Cubes (RIMLS)
Select Visible Points
Smooths normals on a point sets
Surface Reconstruction: Poisson

Compute the normals of the vertices of a mesh without exploiting the triangle connectivity, useful for dataset with no faces

(filter_meshing.dll)



Compute normals for point sets

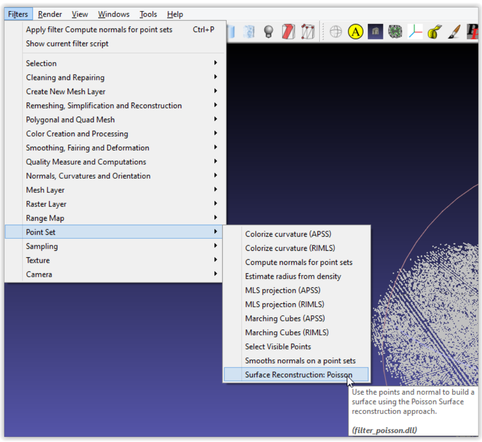
Compute the normals of the vertices of a mesh without exploiting the triangle connectivity, useful for dataset with no faces

Neighbour num	1000
Smooth Iteration	0
☐ Flip normals w.r.t. viewpoint	
Viewpoint Pos.	0 0 0 Get View Dir

(a) Choose the point compute normals. (b) Change the parameters like above.

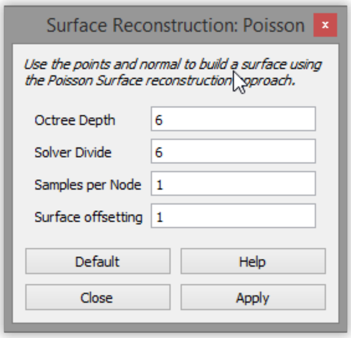
Figure 41: Third step: calculate the normals.

4. Reconstruct the surface by `Filters-> Point Set-> Surface Reconstruction: Poisson`. A gods approach is to change `Octree Depth` to 6, `Solver Divide` to 6, `Samples per Node` to 1 and `Surface offsetting` to 1.


Filters
Render
View
Windows
Tools
Help
Apply filter Compute normals for point sets Ctrl+P
Show current filter script
Selection
Cleaning and Repairing
Create New Mesh Layer
Remeshing, Simplification and Reconstruction
Polygonal and Quad Mesh
Color Creation and Processing
Smoothing, Fairing and Deformation
Quality Measure and Computations
Normals, Curvatures and Orientation
Mesh Layer
Raster Layer
Range Map
Point Set
Sampling
Texture
Camera
Colorize curvature (APSS)
Colorize curvature (RIMLS)
Compute normals for point sets
Estimate radius from density
MLS projection (APSS)
MLS projection (RIMLS)
Marching Cubes (APSS)
Marching Cubes (RIMLS)
Select Visible Points
Smooths normals on a point sets
Surface Reconstruction: Poisson

Use the points and normal to build a surface using the Poisson Surface reconstruction approach.

(filter_poisson.dll)



Surface Reconstruction: Poisson

Use the points and normal to build a surface using the Poisson Surface reconstruction approach.

Octree Depth	6
Solver Divide	6
Samples per Node	1
Surface offsetting	1

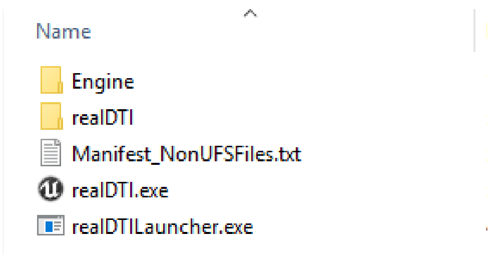
(a) Choose the point compute normals. (b) Change the parameters like above.

Figure 42: Fourth step: reconstruct the surface.

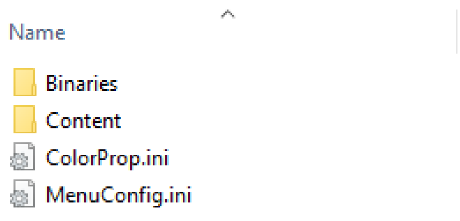
5. If the result is good enough, save the surface as a .obj-file by `File -> Export Mesh`.

A.4. Main program

1. Check the right folder structure. The main folder should include the `realDTI-Launcher.exe` which represents the launcher for the Unreal program. The folder `realDTI` ought contain the two property files `MenuConfig.ini` and `ColorProp.ini`.


Name
Engine
realDTI
Manifest_NonUFSFiles.txt
realDTI.exe
realDTILauncher.exe

(a) Content of the main folder.


Name
Binaries
Content
ColorProp.ini
MenuConfig.ini

(b) Content of the folder `realDTI`.

Figure 43: Check the correct folder structure.

2. Plug in the Oculus Rift and the Leap Motion. The utility tool of the Oculus has to find it and if not press `Service` → `Stop Runtime`. Having done that press `Service` → `Start Runtime`. Now it is possible to test the headset by pressing `Advanced` → `Show Performance HUD`. These steps are only effective within the 0.8 SDK. The problem should not be present at 1.0+.

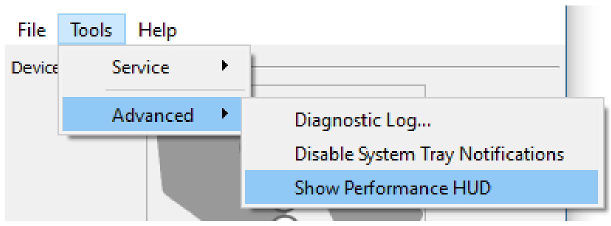


Figure 44: Test the Oculus if it works correctly under SDK 0.8.

3. Start the start up program `realDTILaucher.exe` and set the paths to the fibers text file and to the `.obj` brain file.

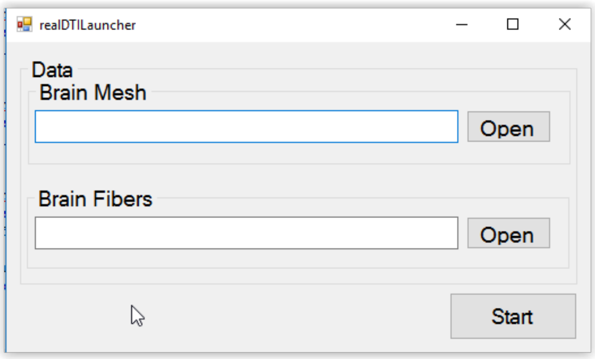


Figure 45: First step: set the paths to the property files.

4. After that click the `Start` button and the Unreal program will start automatically.

   Having done that the program can be controlled with the Leap Motion like described in figure 16 or with the mouse and the keyboard (figure 17).

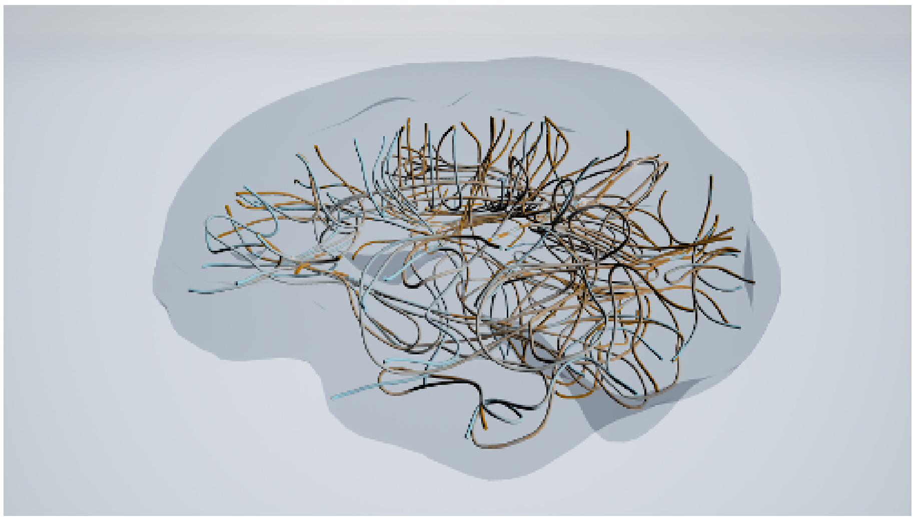


Figure 46: Second step: the brain and fibers are rendered and the Unreal program waits for user inputs