Clara Diederichsen - 27290735

30.05.2019

Exposee

This document contains the results of the final year project supervised by   
Professor Arthur Lowery, which is being undertaken by Clara Diederichsen in her   
final year of a Bachelor of Electrical and Computer Systems Engineering at the   
Monash University

Final Report

Final Year Project – Monash University

Mapping transmitter range onto   
patient-specific   
MRI scans of brain topology

# Significant Contributions

Designed and evaluated an algorithm to fit a robust model to the previously collected transmission range data. Generated a MATLAB script to apply the model to a STL-file and display the associated transmission quality score, given a fixed transmitter position. Implemented a visualisation in MeshLab which allowed a recalculation of results with the update of the transmitter position or orientation.

# Acknowledgements

I would like to take this opportunity to thank everyone who made this project possible.   
Professor Arthur Lowery for his consistent advice and unwavering enthusiasm throughout the year. Julian Szlawski for his advice and criticism as well as for providing me with an exemplary brain topology STL-file to be evaluated.   
Tim Feleppa for collecting the transmission range data that made this project possible and for his consistent input and advice.   
Friends and family for their unwavering support and encouragement.

# Executive Summary

The project was undertaken as a part of the Monash Vision Group Project to develop a piece of software that would reliably calculate the quality of transmission on a given patient-specific   
3D-model of a brain surface scan.  
As such a method of interpreting the collected transmission range data had to be implemented and tested for accuracy, as well as developing a method to visualize the results in a manner that is easy to understand. This was accomplished by generating MATLAB code to fit a robust polynomial model to the data and tested using simple geometry. The found polynomial models were then implemented as a MeshLab filter to improve the visualisation of the results as well as reduce the run time of the code execution.

The generated fitted models were tested on simple geometry, that allowed a separate calculation of the expected score values to test the execution of the code.

The developed software in MATLAB was able to estimate the data transmission quality of new faces with an adj. R-squared accuracy that explained the variety of more than 97% of the On-axis displacement with a Standard deviation of less than one millimetre.   
A secondary method was developed to reduce the processing time to less than 15s, with a reduced accuracy of 96% with a standard deviation of the residuals of less than 1.5mm of the on-axis displacement.

To visualise the data, the results were displayed in three different colours using MeshLab. 3D-printing the generated model was considered but decided against due to difficulty in visualising the complex results in a 3-dimensional layer by layer approach.

Suitable imaging technologies for generating the input STL-files were considered and evaluated and the decision was reached that MRI imaging provides the most suitable results.

To expand on this project an implementation of a true Real time updating system would be desirable that removes the need for running both MeshLab and MATLAB and may be accomplished by modifying the open source code provided for MeshLab.

# Table of contents

[Significant Contributions i](#_Toc10303010)

[Acknowledgements i](#_Toc10303011)

[Executive Summary ii](#_Toc10303012)

[Table of contents iii](#_Toc10303013)

[1. Introduction 1](#_Toc10303014)

[1.1. Project Aim 1](#_Toc10303015)

[1.2. Background 2](#_Toc10303016)

[2. Data Fitting 3](#_Toc10303017)

[2.1. Previously collected transmission range data 3](#_Toc10303018)

[2.2. Fitting a robust model to the transmission data 5](#_Toc10303019)

[3. STL model evaluation 10](#_Toc10303020)

[3.1. Preparation of faces for the application of the polynomials 10](#_Toc10303021)

[3.2. Testing on simple geometry 11](#_Toc10303022)

[3.3. Testing validity of the found polynomials on simple geometry 13](#_Toc10303023)

[3.4. Testing of polynomials on brain topology scan 14](#_Toc10303024)

[4. Visualisation of results using MeshLab 15](#_Toc10303025)

[4.1. Simplification of the STL input 16](#_Toc10303026)

[4.2. MeshLab filters 16](#_Toc10303027)

[4.3. Visualisation using 3D-printing/Colour Gradient 18](#_Toc10303028)

[5. Results 19](#_Toc10303029)

[6. Code execution 20](#_Toc10303030)

[6.1. Reading STL-files 20](#_Toc10303031)

[6.2. Calculation of the transmission scores 21](#_Toc10303032)

[6.3. Saving STL-files in colour 22](#_Toc10303033)

[6.4. Recalculation of the transmitter position 23](#_Toc10303034)

[7. Requirements for use 24](#_Toc10303035)

[7.1. Requirements for the imaging techniques 24](#_Toc10303036)

[7.2. Requirements for the STL-file input 25](#_Toc10303037)

[7.3. Hardware requirements 25](#_Toc10303038)

[8. Protocol for the replication of the results 26](#_Toc10303039)

[8.1. Pre-processing of a large resolution STL-file 26](#_Toc10303040)

[8.2. Calculation of transmission scores 27](#_Toc10303041)

[8.2.1. MATLAB Calculation of transmission scores 27](#_Toc10303042)

[8.2.2. MeshLab Calculation of transmission scores 28](#_Toc10303043)

[8.3. Recalculation of transmission scores 29](#_Toc10303044)

[8.4. Recalculation of polynomials 29](#_Toc10303045)

[9. Reflection and ideas for further improvement 30](#_Toc10303046)

[10. Conclusion 31](#_Toc10303047)

[11. References 32](#_Toc10303048)

[12. Appendix 34](#_Toc10303049)

[12.1. Appendix A - Poster Submission 34](#_Toc10303050)

[12.2. Appendix B – Data Fitting – Goodness of fit results 35](#_Toc10303051)

[12.3. Appendix C – Code 39](#_Toc10303052)

[Figure 1: Monash Vision Group Gennaris [13] 2](file:///C:\Monash\Study\FYP\Documents\Report\Report.docx#_Toc10303053)

[Figure 2: Potential tile arrangement on the visual cortex [1] 2](#_Toc10303054)

[Figure 3: Setup of the experimentally collected data points between receiver tile (green) and transmitter coil (blue) 3](file:///C:\Monash\Study\FYP\Documents\Report\Report.docx#_Toc10303055)

[Figure 4: Collected transmitter data at an off-axis distance of 5mm showing the variation dependent of the angle and on-axis distance 4](file:///C:\Monash\Study\FYP\Documents\Report\Report.docx#_Toc10303056)

[Figure 5: Point cloud showing the transmission data points that were collected, where lossless (lossless) indicates the transition between lossless-lossy and lossy (yellow) indicates the transition between lossy-no transmission 5](file:///C:\Monash\Study\FYP\Documents\Report\Report.docx#_Toc10303057)

[Figure 6: Both polynomials plotted with the raw data points, where yellow indicates the transition between lossy-no transmission and green indicates transition between lossless-lossy 6](file:///C:\Monash\Study\FYP\Documents\Report\Report.docx#_Toc10303058)

[Figure 7: Flat surface test to test the operation of the code and the different transitionary off-axis distances set to 40 and 60mm, where the transmitter (blue) is positioned at [15,50,38.5] 11](#_Toc10303059)

[Figure 8: Curved surface test to test the operation of the code and the different set transitionary angles set to 45 and 90degrees, where the transmitter (blue) is positioned at [0,0,0] with a norm of [1,1,1] 12](#_Toc10303060)

[Figure 9: Testing of the polynomials on both the flat surface and the sphere with the transmitter position remaining unchanged. 13](file:///C:\Monash\Study\FYP\Documents\Report\Report.docx#_Toc10303061)

[Figure 10: Left hand brain topology with the calculated quality scores displayed in green (lossless) , yellow (lossy) and red (no transmission) 14](file:///C:\Monash\Study\FYP\Documents\Report\Report.docx#_Toc10303062)

[Figure 11: Meshlab render of the MATLAB application of the polynomials on a STL-file of a brain surface 15](file:///C:\Monash\Study\FYP\Documents\Report\Report.docx#_Toc10303063)

[Figure 12: Output of the 'per face Function' filter after processing the left-hand side of the exemplary brain surface with r= 255\*x1/100 and g=255\*y1/75 17](file:///C:\Monash\Study\FYP\Documents\Report\Report.docx#_Toc10303064)

[Figure 13: Flowchart showing the correct reading of an STL-file into MATLAB 20](file:///C:\Monash\Study\FYP\Documents\Report\Report.docx#_Toc10303065)

[Figure 14: Flowchart showing the assignment of transmission scores 21](file:///C:\Monash\Study\FYP\Documents\Report\Report.docx#_Toc10303066)

[Figure 15: Flowchart showing the saving of different STL-files in their corresponding colour where lole = 'lossless', lo = 'lossy and notr = 'no transmission' 22](file:///C:\Monash\Study\FYP\Documents\Report\Report.docx#_Toc10303067)

[Figure 16: Flowchart showing the recalculation of the transmission scores based on an update from the Transmitter orientation or position where T\_pos = Transformer position, Tf = Transformation matrix, T\_no = Transmitter norm and \_o indicates the old position 23](file:///C:\Monash\Study\FYP\Documents\Report\Report.docx#_Toc10303068)

[Figure 17: Visible loss of detail through Simplification using the Quadric Edge Collapse Filter and reducing the number of faces from 286210 (left) by 50% to 143105 faces (right) 26](file:///C:\Monash\Study\FYP\Documents\Report\Report.docx#_Toc10303069)

[Figure 18: Comparison between the two different color schemes employed with [red/yellow/green] on the left and [white/yellow/green] on the right where both schemes correspond to [no transmission/lossy/lossless] 28](file:///C:\Monash\Study\FYP\Documents\Report\Report.docx#_Toc10303070)

# Introduction

This project was undertaken as part of the Monash Vision Group project that aims to develop a set of brain implants in combination with glasses worn by the user to restore vision.  
As the placement of the implants cannot be changed after the surgery, a form of guidance shall be constructed that allows the surgeon to identify suitable transmission quality regions for implantation, dependent on the position of the transmitter coil, on the visual cortex of the patient and hence allow them to decide on the optimal position for the implant within the suitable regions. As such the primary requirement for the software was to be usable by a person without extensive knowledge of software or electrical engineering.

## Project Aim

The aim of this project was to implement a software that allows the user to find the best Receiver Tile positioning on the visual cortex of the patient dependent on the transmitter range data collected previously.   
As such the project aimed to develop a reliable model that would be able to estimate the quality of transmission dependent on the positioning of a receiver tile to the position of the transmitter itself. This model was then to be tested on different geometric shapes to ensure that it was functioning correctly and then to be applied to an exemplary STL-file generated from a MRI scan.   
The software was then to be expanded to allow an easy to use case to be reapplied to different STL-files and make a reproduction of the results for new scans possible.

## Background

Monash Vision Group is a collaboration between Monash University and Industry partners, which aims to develop a clinically viable cortical vision prosthesis, referred to as the Gennaris bionic vision system or short ‘Gennaris’.

As part of this, the Final Year Project is to develop a method of interpretation and application of the recorded transmission performance of the wireless power and data link, between the transmitter module and the implant tile and apply this to a patient’s specific brain topology and visualise the latter. This shall allow for optimal tile placement on the patient’s visual cortex to ensure an acceptable data signal quality. In addition, the project shall include visualisation of the transmission results both through software. It shall allow the user to specify the position of the transmitter in relation to the STL-model and then start an automated calculation of the facets suitable for implantation based on the input.

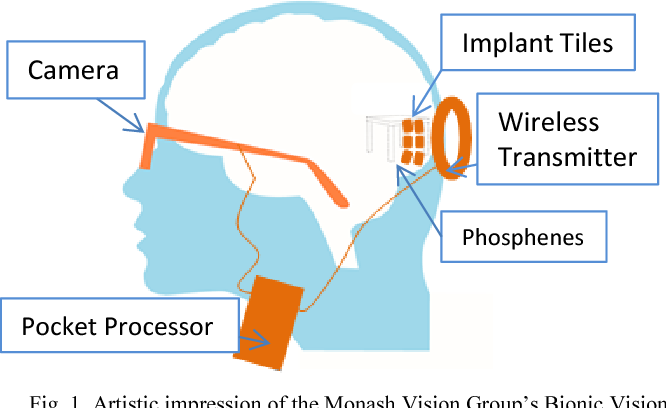


Figure 1: Monash Vision Group Gennaris [13]

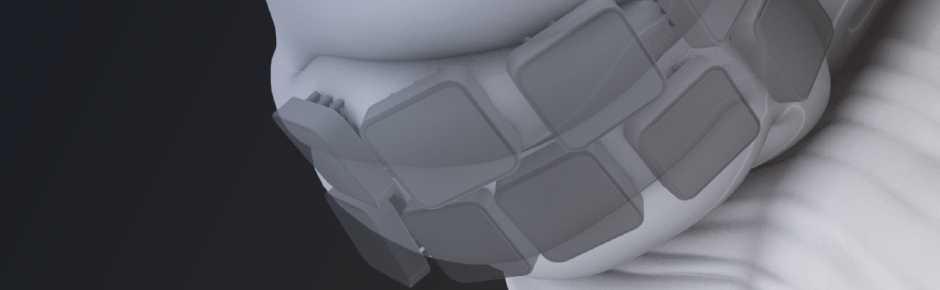


Figure 2: Potential tile arrangement on the visual cortex [1]

# Data Fitting

To be able to estimate the quality of transmission dependent on the position of the transmitter to the 3D-model of the brain surface the transmission range had to first be tested experimentally and then implemented in software. Different data points had previously been collected to give an estimate of the data transmission quality, which then was to be used to generate a robust model to identify the quality of transmission on new configurations.

## Previously collected transmission range data

The previously, by the Monash Vision group, collected data was gathered in an experimental setting in which different configurations of the transmitter to the receiver were evaluated. To gather the data points two different setups were considered, in which either the Receiver Tile was fixed, and the transmission coil was adjusted to different positions or the Coil was fixed, and the Tile position varied [2]. This resulted in two sets of readings that were collected [2].

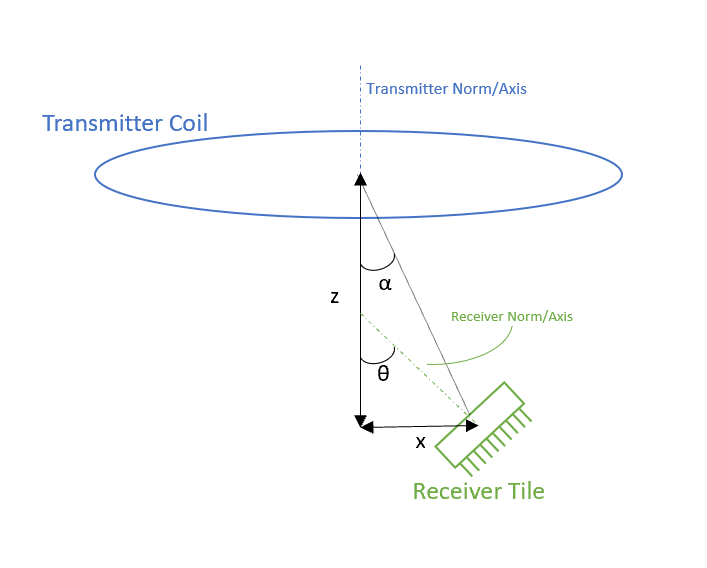


Figure 3: Setup of the experimentally collected data points between receiver tile (green) and transmitter coil (blue)

As shown in Figure 3 the transmission data was collected by varying three variables, namely the distance between the receiver and transmitter on the transmitter axis (z), the distance perpendicular from the normal of the transmitter (x) and the angle between the transmitter normal and the normal of the receiver (θ). These parameters were recorded in a file that took measurements of the transmission data quality at different angles and on-axis separations with 5mm increments of off-axis displacements [2].

The data, on the range of the transmitter, showed the characteristics of the transmission grouped into regions of transmission where the data was transmitted without any losses (lossless), with some errors (lossy) and a region of no transmission. As such colours were assigned to the different regions to easily identify the quality of transmission, with green corresponding to lossless, yellow to lossy and red to no transmission, as shown in Figure 4.

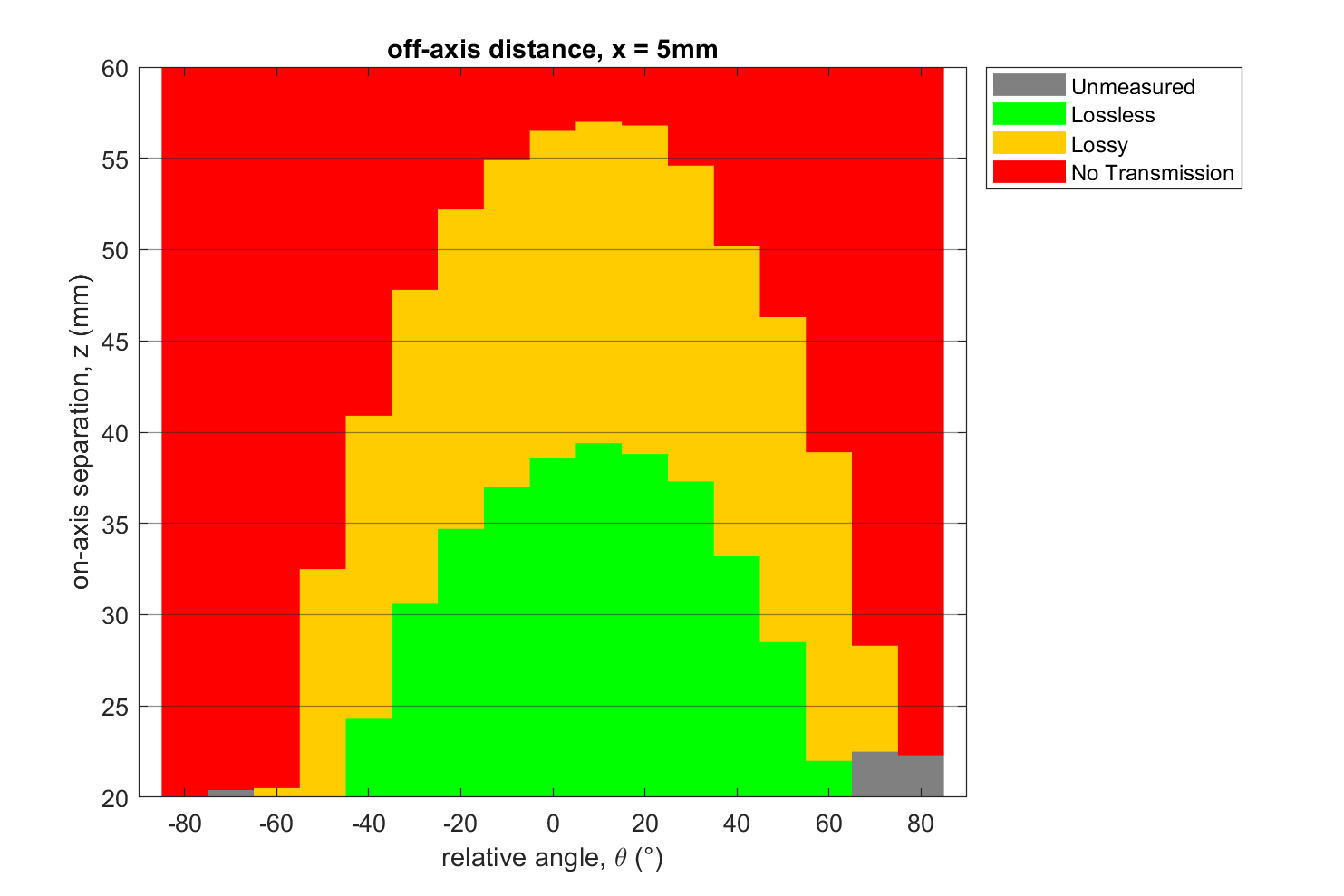


Figure 4: Collected transmitter data at an off-axis distance of 5mm showing the variation dependent of the angle and on-axis distance

As shown in Figure 4, if at a certain fixed angle and fixed off-axis displacement the on-axis distance was varied, the transmission always decreased with an increasing on-axis separation.   
Hence, as this is in line with electromagnetic theory behind a transmitter coil, in which the signal will be stronger the closer the receiver is to the coil, the on-axis distance can be used to identify the transmission quality, if the transitionary on-axis distances for a certain configuration are known [3].   
It can be inferred, that if the on-axis separation of the evaluated position is smaller than the transition between lossless and lossy, that the transmission at that point will be lossless. Similarly, if the separation is found to be larger than the transition separation between lossless to lossy, but smaller than the transitionary distance between lossy to no transmission, the quality of transmission will be lossy.   
To be able to evaluate a new configuration of the receiver to the transmitter, a model needed to be fit to the collected data to calculate the transitionary distances.

## Fitting a robust model to the transmission data

The collected data was imported into MATLAB and then converted into a 3-dimensional array, where the x-axis corresponds to the angle in degrees, the y-axis to the off-axis displacement and the z-axis to the on-axis distance. The model was required to correctly find the on-axis distance at which the transmission transitions between the different quality regions given a certain angle and off-axis displacement as input.   
The input to the model therefore would be the calculated angle and off-axis displacement and the output would be both the on-axis distance for transition between lossless-lossy and the on-axis distance for the transition between lossy and no transmission.   
Given the raw data array a point cloud was created shown in Figure 5. This point cloud indicates that a set of two polynomial functions would be able to estimate the surface that is created by the raw transitionary data points.



Figure 5: Point cloud showing the transmission data points that were collected, where lossless (lossless) indicates the transition between lossless-lossy and lossy (yellow) indicates the transition between lossy-no transmission

The two polynomials that most closely describe the data points are shown in Figure 6. These polynomials were found by iterating the model fitting to find the most inliers within a 95% confidence interval.

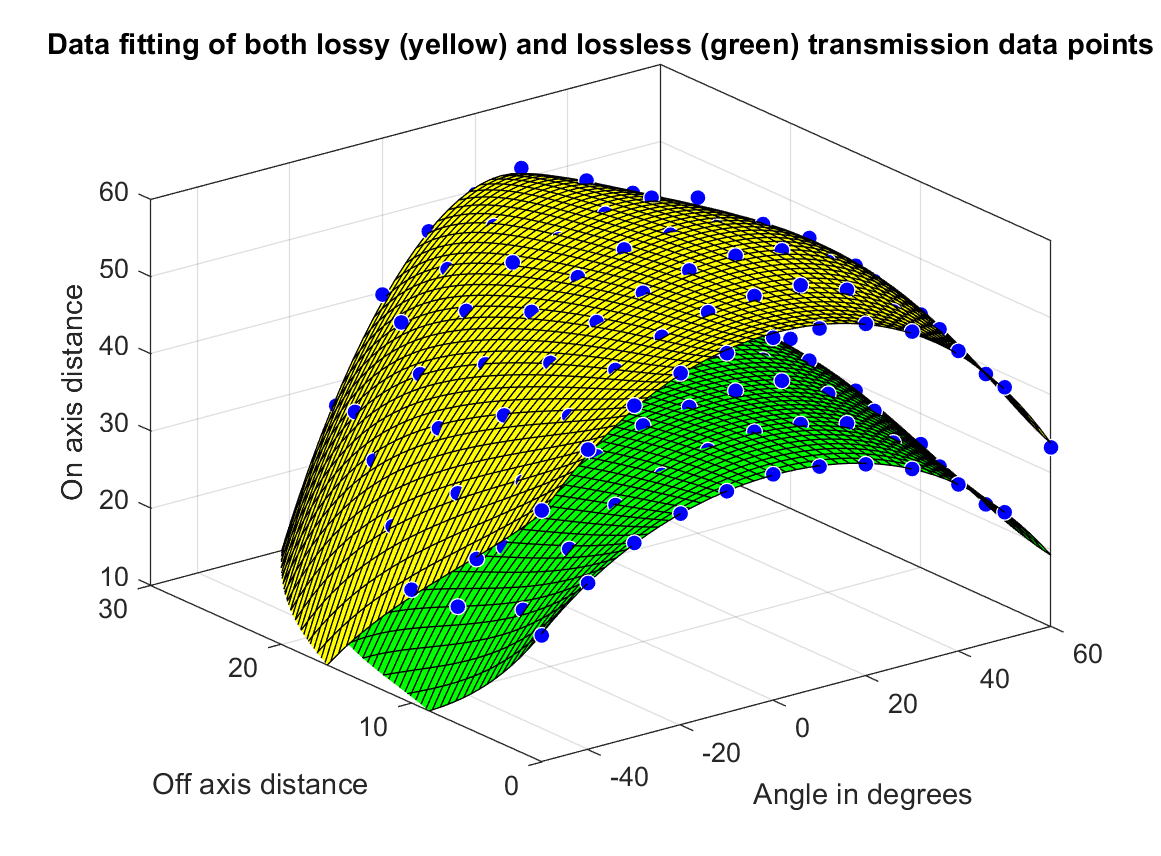


Figure 6: Both polynomials plotted with the raw data points, where yellow indicates the transition between lossy-no transmission and green indicates transition between lossless-lossy

The best degree of the polynomials fit was found through iteration of the different polynomial degree combinations and determining their goodness of fit. In addition, two different robust fitting methods were applied to the data and compared.   
The first, Least absolute residuals (LAR), was used to find the best fit using a model that weighs all the datapoints with the same weight, while the second, robust Bisquare fitting, adjusts the weight of the datapoints according to their distance from the fitted model. This therefore allowed for a comparison of two data fitting models under the assumption that either all data points carry the same importance for calculation of the polynomial or under Bisquare that the data contained a significant amount of outliers that needed to be accounted for by adjusting the weighting of the different datapoints. This was found to be a suitable approach as the two found polynomials which provided the best fit, were generated using either method, indicating that one of the transitionary distances, namely the Lossy-No Transmission transition, data contained outliers.

To evaluate the goodness of fit of a polynomial model, four different variables were considered [4].   
The first variable, was the sum squared error (SSE) in which the cumulative sum of the error between the datapoint and the polynomial expectation squared as represented by Equation 1, gives an indication of how close the fit model is to the datapoints. A lower SSE value indicates a better fit.

Equation 1: Formula to calculate the sum squared error of a data set and its expectation

The second variable taken into account was the R-squared score that was calculated by finding the Sum squares of the regression (SSR) divided by the sum squared about the mean (SST) which is equivalent to Equation 2:

Equation 2: Formula for calculation of R-squared of a data set and its fitted model

This value is a measure of the variation of the data around the average corresponding to the percentage of the variation of the data it can explain. Therefore, a higher R-squared score is associated with a better fit.   
The third and most significant variable was found to be the adjusted R-square. This value, based on the residual degrees of freedom (v), which is the number of fitted coefficients (m) subtracted from the number of response values (n), adjusts the previously calculated R-square value as shown in Equation 3. This is a valuable parameter to compare two models with a different number of coefficients, as are being generated in this case, as it takes the used coefficients into account. As such a higher adjusted R-square value indicates a better fit.

Equation 3: Formula for calculation of adjusted R-squared of a data set and its fitted model

The last variable considered, to evaluate the goodness of fit, was the Root Mean squared Error (RMSE), which like the SSE indicates a better fit if the value is smaller. The RMSE gives a estimate of the standard deviation of the random component in the data and is defined as the square root of the SSE divided by the number of residual coefficients v.  
For both polynomials the best fits, which were providing the optimal degree of goodness of fit were found. For the transition between lossy to no transmission, a model was found that was optimal for the Adj. R-squared and RMS values and second best for SSE and its R-squared values, while for lossless to lossy one model was found that provided the optimal results in all four categories. The model that is being fit to the data is described by ‘polyij’ where i is the degree in x (angle θ) and j is the degree in y (the off-axis distance).

Table 1: Data Fitting - Goodness of fit results showing the best fits for both transitions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Transition | Model | SSE | R-squared | Adj.  R-squared | RMSE |
| Lossless - Lossy | ‘poly45’  ‘LAR’ | 36.6046132 | 0.98438072 | 0.97832427 | 0.86431066 |
| Lossy - No Transmission | ‘poly45’ ‘Bisquare’ | 30.6242129 | 0.99539757 | 0.99386343 | 0.73298492 |

The found adjusted R-squared values indicate that the fit of both models describes more than 97.5% of the data variation can be accounted for by using the two models. The RMSE value indicates that the standard deviation of the residuals is less than one millimetre, indicating that the model will be able to provide precise estimates of the data transmission quality, when the off-axis displacement and angle are known.

The found number of degrees in x and y indicate that the polynomials are to be of the form shown in Equation 4, with the coefficients shown in Table 2:

Equation 4: Polynomial equation for i = 4 and j = 5 for the data fitting of the transitionary on-axis distances, where x is the angle and y is the off-axis displacement

Table 2: Coefficients of the best fit polynomials

|  |  |  |
| --- | --- | --- |
|  | Lossless -Lossy | Lossy - No Transmission |
| p00 | 39.1967742 | 57.61367 |
| p10 | 0.00440629 | 0.00556022 |
| p01 | 0.08901657 | -0.453678 |
| p20 | -0.0041869 | -0.005113 |
| p11 | 0.02041878 | 0.01180269 |
| p02 | -0.0242063 | 0.06466352 |
| p30 | -2.80E-06 | -2.07E-06 |
| p21 | -0.0004348 | -1.08E-05 |
| p12 | -0.0008122 | 0.00108252 |
| p03 | -0.0001339 | -0.0068191 |
| p40 | -3.52E-07 | -4.17E-07 |
| p31 | 4.84E-06 | 5.16E-06 |
| p22 | 4.53E-05 | 9.03E-06 |
| p13 | 1.38E-05 | -9.14E-05 |
| p04 | 4.16E-05 | 0.0002831 |
| p41 | -3.82E-08 | -5.07E-08 |
| p32 | -1.00E-07 | -1.23E-07 |
| p23 | -1.24E-06 | -7.26E-07 |
| p14 | 5.40E-07 | 2.53E-06 |
| p05 | -1.18E-06 | -4.47E-06 |

As can be seen from the p00 values in Table 2, the polynomial coefficient for the lossy to no transmission is larger than the one for the transition between lossless and lossy, which is in line with the theory discussed earlier, that the transmission worsens with increasing on-axis displacement.

# STL model evaluation

To apply the found polynomial models to the STL-model generated from the MRI-scan of the patient’s brain the STL-file first had to imported into MATLAB, which was accomplished by finding whether the model was saved as an ASCII or binary file and then reading in the vertices and face information saved in the file. The three-dimensional model is saved in an STL file as triangles described by three vertices making up a single face. This information is then broken up into arrays containing the vertex positions and the information of which three vertexes make up a face in a secondary array.

## Preparation of faces for the application of the polynomials

To allow for the calculation of the off-axis and on axis distances as well as the angle between the norm of a face and the norm of the transmitter, the midpoint of each face had to be found.   
To find the midpoint of a face its three known 3d-points (A,B,C) were used. First the cross product of the two vectors AB and AC (ABxAC) is found. This is then used to find the displacement caused by the point B and the point C by using Equation 5, where Norm() describes the 2-Norm or the length of the vector.

Equation 5: Midpoint calculation of a face given its three corner points

The normal of a face can be found saved in the STL-file and as such the angle α between the normal of the face NF and the normal of the transmitter NT can be found using Equation 6.

Equation 6: Angle θ calculation of a face given the Normal of the transmitter and the Normal of the face

Finding the angle α by using the Equation 6 and substituting TF with the hypothenuse, allows for the calculation of the angle α to then find the on-axis distance (z) and off-axis distance (x) of the midpoint to the transmitter as shown in Figure 3 using simple trigonometry (Equation 7).

Equation 7: Off-axis and on-axis distance calculation based on the angle θ, the transmitter position and the midpoint

The Midpoints are saved into a separate array under the assumption that the model will not be displaced in the MATLAB calculation, to reduce the run time of the code.

## Testing on simple geometry

To prepare for testing the application of the Polynomials on a complex brain surface STL-file, multiple simple geometries were analysed first with set transitionary distances of the off axis distance, to allow for a first visualisation of the different transmission regions in MATLAB and test the performance of the code and the validity of the earlier discussed equations.   
From the data fitting, the values 40 and 60mm were chosen and used as transitionary values for a test of the off-axis distance calculation.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Figure 7: Flat surface test to test the operation of the code and the different transitionary off-axis distances set to 40 and 60mm, where the transmitter (blue) is positioned at [15,50,38.5]

As Figure 7 shows the off-axis distance is calculated correctly and can then be used to find the different regions of operation.

One can therefore assume, that the calculation of all the different variables associated with each face is executed correctly as the off-axis distance is calculated using the angle α.

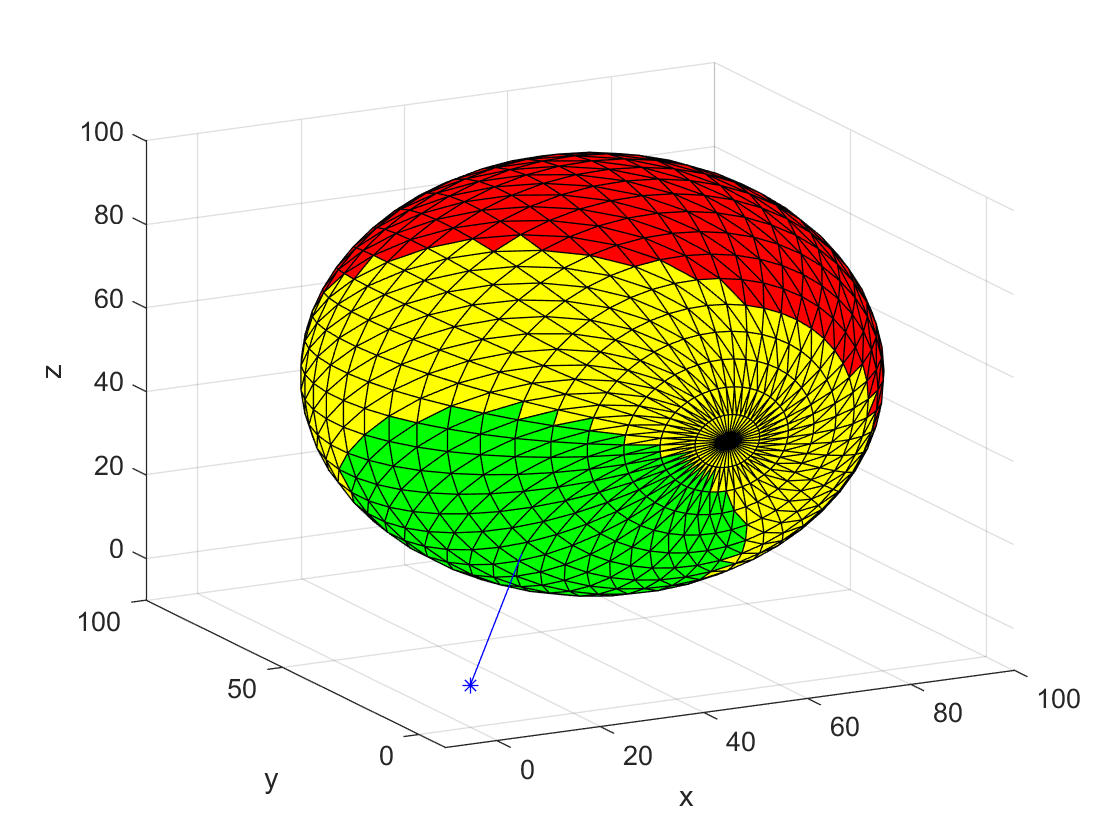
To further ensure correct operation and the correct calculation of the angle θ, a similar test was run on a sphere as shown in Figure 8, where the transitionary angle was set to θ=45 and 90 degrees. The transmitter was positioned at the origin with its normal pointing towards the centre of the sphere. This allows for an evaluation of the different angles between the normal of the face and the normal of the transmitter.

Figure 8: Curved surface test to test the operation of the code and the different set transitionary angles set to 45 and 90degrees, where the transmitter (blue) is positioned at [0,0,0] with a norm of [1,1,1]

Figure 8 indicates that the angle θ associated with every face is calculated correctly, as the transition between ‘lossless’ to ‘lossy’ as well as from ‘lossy’ to ‘no transmission’ is found to be visualized at the correct position of the sphere. It is important to note that the test also indicates that the angle between the different norms is correctly found to be negative when the faces on the opposite site of the sphere are evaluated, which is correctly indicated as that no transmission would be possible on those faces.

## Testing validity of the found polynomials on simple geometry

Having confirmed that the distances and angles are calculated correctly, the next step was to implement the calculation of the on-axis cut off distances using the generated polynomials and test them on the same surfaces. As discussed prior, finding the off-axis distance and the angle θ allows for the calculation of the transitory on-axis distances.

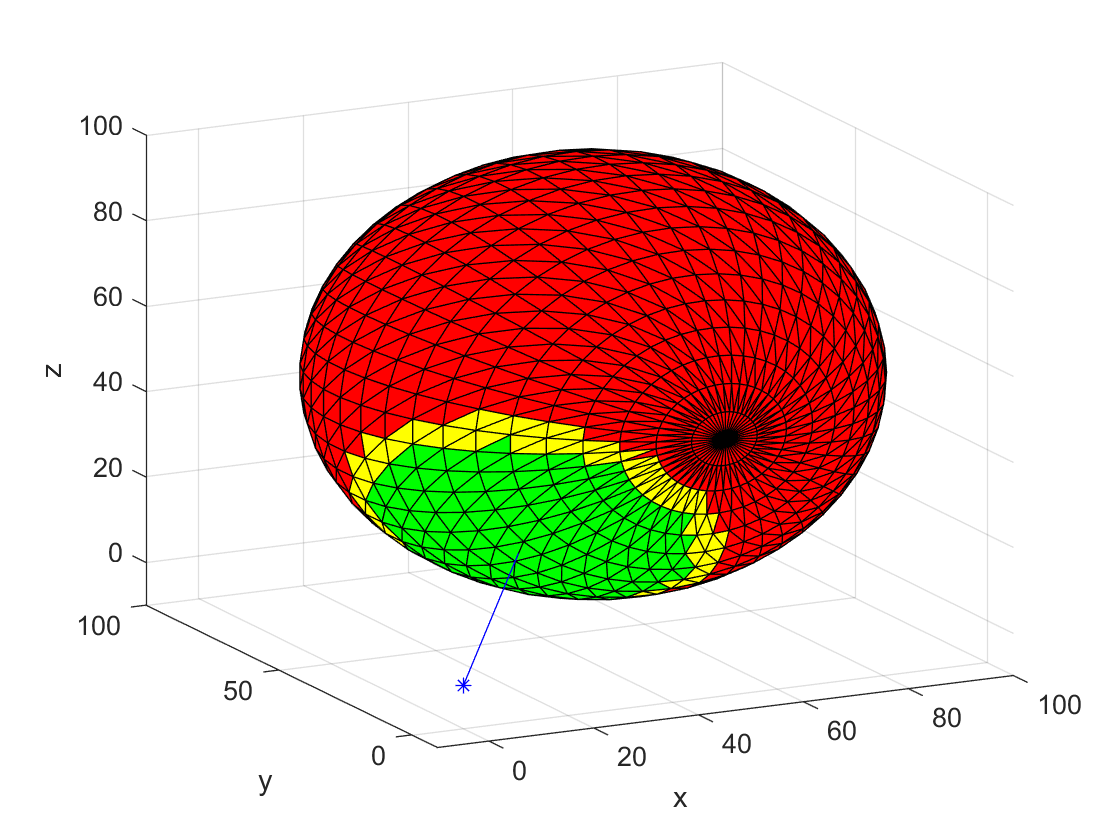
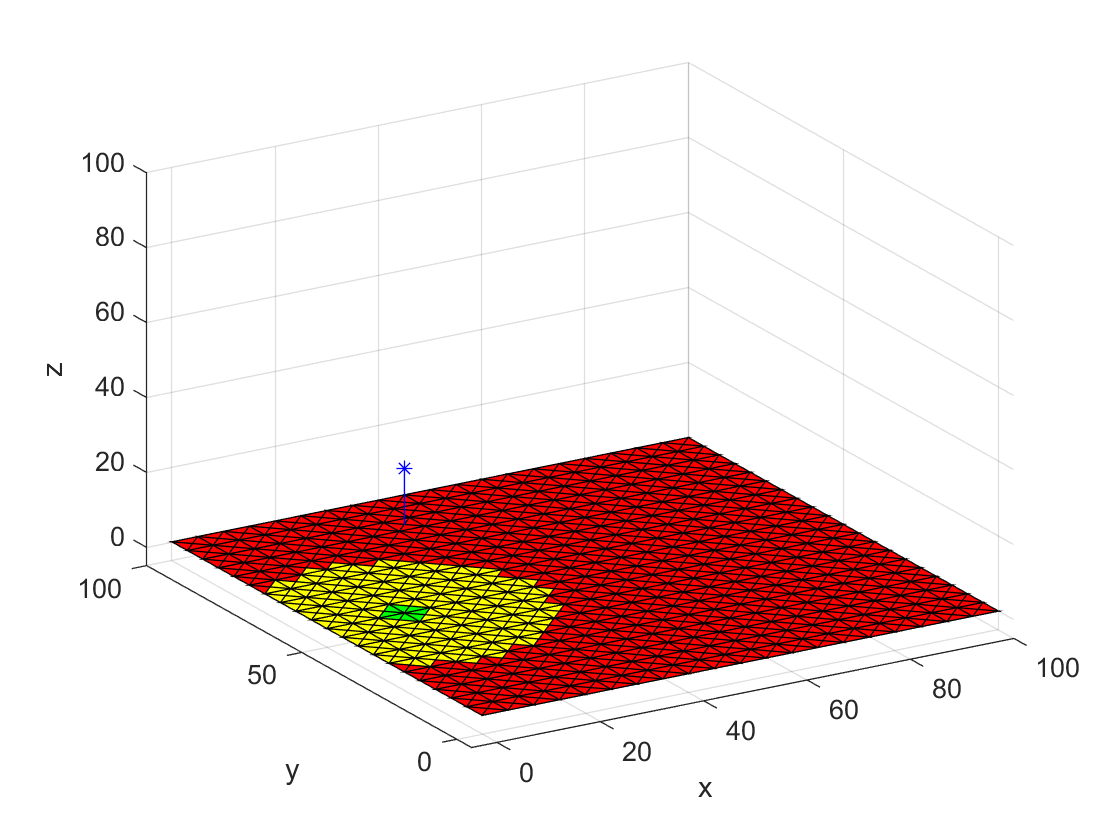


Figure 9: Testing of the polynomials on both the flat surface and the sphere with the transmitter position remaining unchanged.

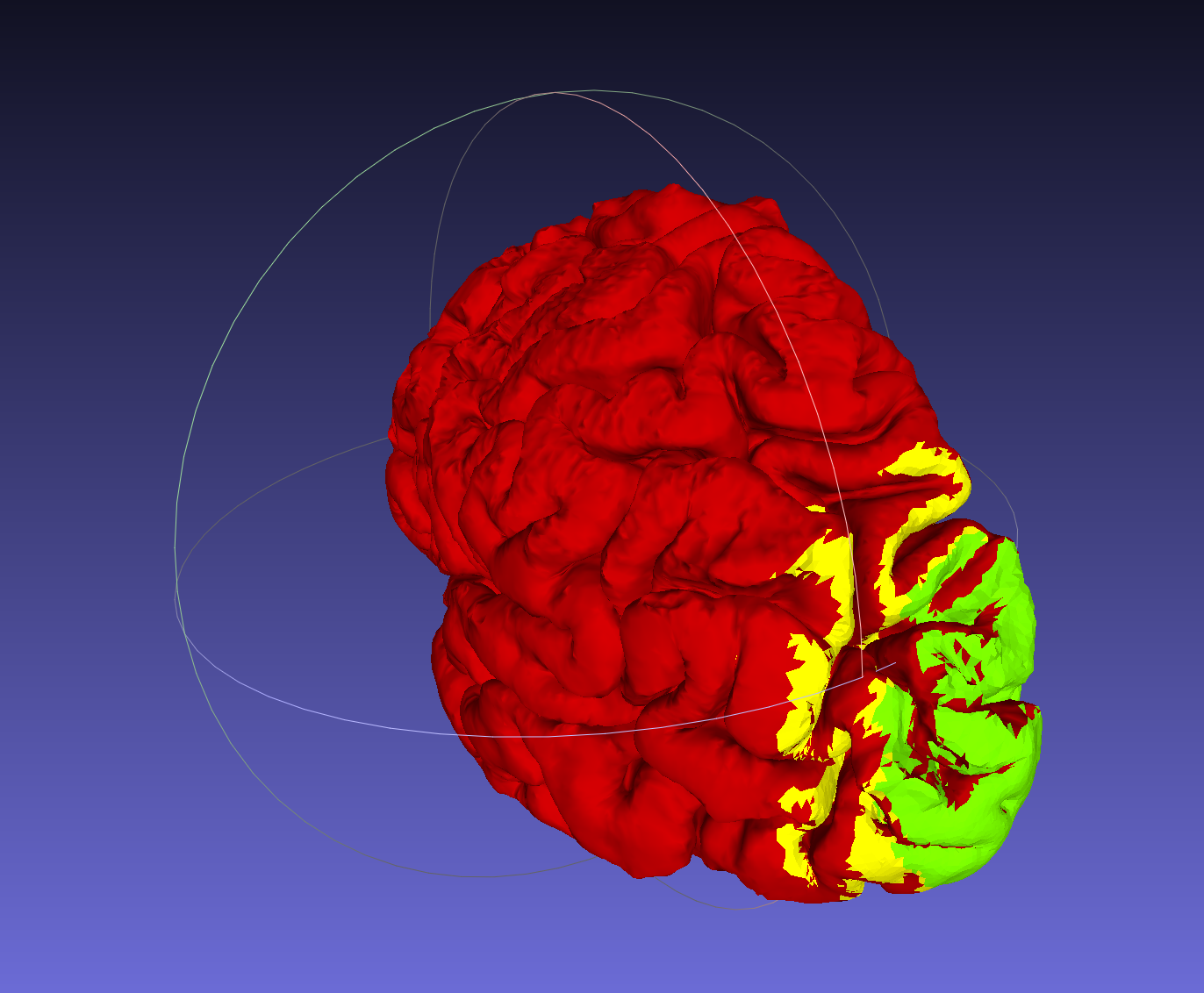
As shown in Figure 7 the transitionary on-axis distances are found to be affected by both variations in the off axis-distance and the angle θ. The off-axis transitionary distances with a fixed angle θ, that is shown on the left of Figure 9, can be cross-referenced with the results of the polynomial calculation shown in Figure 6, which show the same drop of transmission quality with increasing off-axis distance, as displayed in the results on the simple geometry.

These tests indicate that the polynomials are applied to the evaluated surface correctly and may as such be applied to the more complex brain topology scans.

## Testing of polynomials on brain topology scan

The application of the code to the left-hand side of the exemplary brain topology resulted in the results shown in Figure 10.

Figure 10: Left hand brain topology with the calculated quality scores displayed in green (lossless) , yellow (lossy) and red (no transmission)



The increased complexity of the model lead to execution problems, such as a significant increase in processing time due to the increased number of faces and vertices evaluated as well as the inability of MATLAB to display the calculated results. The processing the left half of the exemplary brain topology, with 286,210 faces, resulted in a MATLAB code execution time of around 30minutes, which failed to correctly display the results. This execution time in comparison to around 10s when processing simple topology, such as a sphere with 2352 faces, meant that the increase in processing time was linear, which lead to the assumption that the execution time of evaluating a singular face had to be reduced to reduce the processing time. To overcome this problem the 3D-rendering software MeshLab was used.

# Visualisation of results using MeshLab

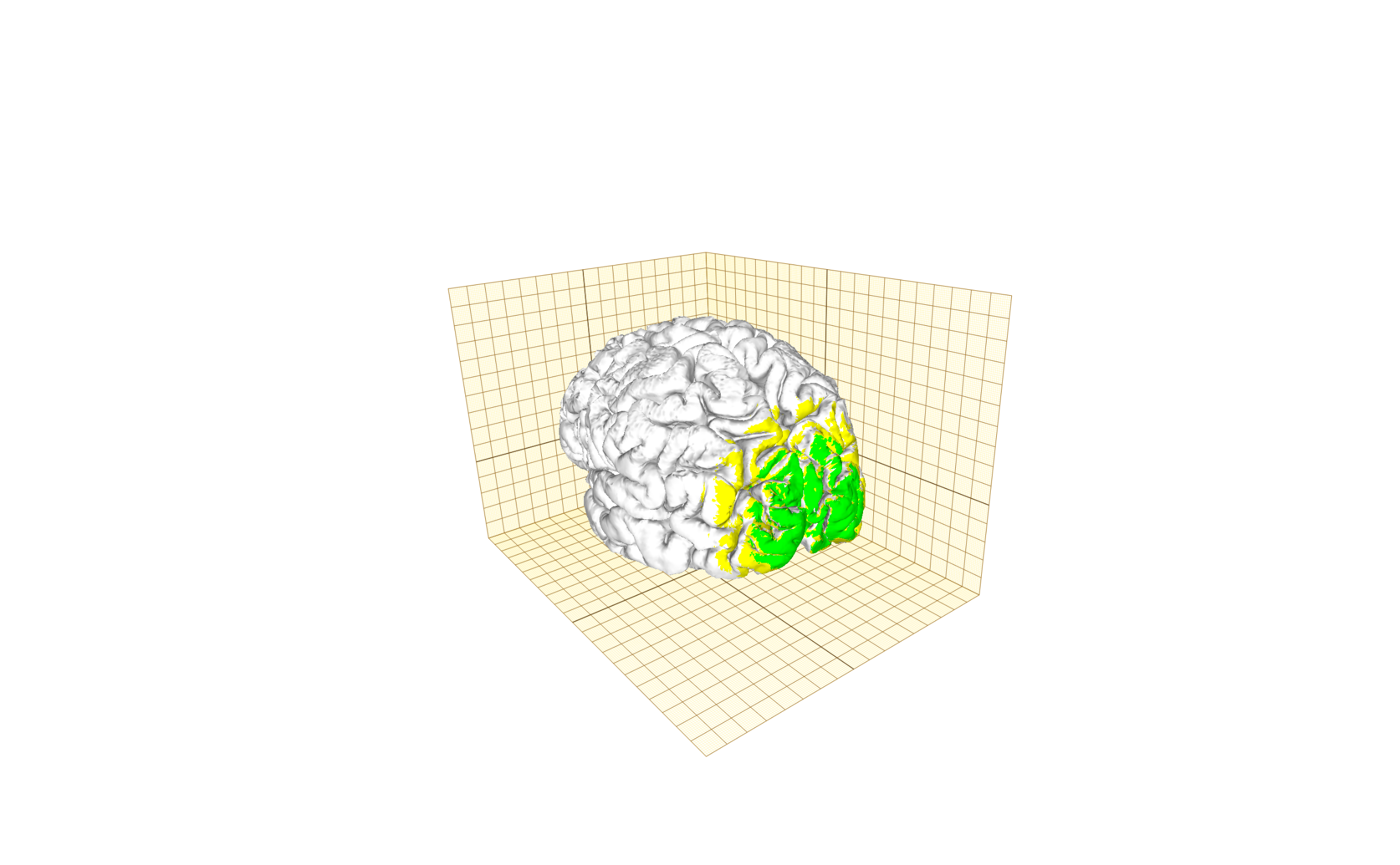
To visualize the results of the calculations in a manner that is easy to understand and can be updated, the results from MATLAB were exported into a 3D-rendering software called MeshLab. As such a MATLAB script was developed to write STL-files and save them in a certain colour. The explanation of this script can be found explained in Section 6.3. These generated STL-files were than to be imported into MeshLab to show the different transmission regions.   
After the functionality of this script was confirmed, the polynomials were applied to both halves of an exemplary brain surface scan created from an MRI-scan. The results that were created are shown in Figure 11.

Figure 11: Meshlab render of the MATLAB application of the polynomials on a STL-file of a brain surface

To simplify the recalculation of the different transmission regions a MATLAB script was created that took the saved Meshlab project file and imported any changes to the position of the transmitter and recalculated the transmission values of the different files and saved new STL-files. This script was found to be the first implementation of a case in which the user could change the position of the transmitter and the transmission data would change adaptively and is described in Section 6.4.  
As discussed, calculating the colour of the different faces of one half of the brain topology took around 30 mins due to the large quantity of data and as such the code performance had to be improved significantly. Hence, a way to embed the quality score calculation into Meshlab and to remove the use for any calculations in MATLAB had to be found, as well as simplifying the complexity of the code.

## Simplification of the STL input

To improve the performance of the software, the first strategy implemented was to reduce the number of faces and vertices that were taken as input, as most of the faces lay outside the region of maximal lossy on-axis displacement and were assigned a score of ‘no transmission’. To implement this, the STL-files was to be broken up into smaller subsets that could be evaluated as a whole, if their outermost edges were saved. Hence, if they contained faces that are closer than the maximal on-axis displacement the subset was to be evaluated. This approach proved to be difficult to implement, as the faces within a STL file are not saved in a coherent order and as such the calculation of the outermost vertices of a subset, in relation to a new transmitter position, proved to be a task too complex to implement under the improvement of the performance of the code.   
As reducing the number of the processed faces in that way was found not to be practical the simplification of the model through pre-processing was considered. This was accomplished using one of the filters that is inbuilt into MeshLab and is described in Section 8.1, but resulted in the loss of significant detail of the model, without a significant reduction of the execution time of the MATLAB scripts.   
When reducing the number of faces used to represent the model, the time used to processes it reduced linearly, which indicates that the use of a pre-processing filter before the application of the different transmission range models will consistently reduce the run time and should be considered if the model is found to be of a very high resolution.   
As no significant reduction of the runtime was accomplished by simplifying the STL input file the next strategy to be considered, was the embedding of the code into MeshLab.

## MeshLab filters

As M MeshLab is an open source program, the first approach to this problem was to find a way to implement the generated MATLAB code in one of MeshLab pre-existing filter functions to render the different faces. The filters inbuilt into MeshLab allow the user to process a 3D-triangular mesh and display the results.   
The best inbuilt filter for such an application was found to be the ‘Per Face Colour Function’ filter, which is part of the ‘Colour Creation and Processing’ filters. This specific filter allows the user to specify a string function for each of the RGB-values of all the faces of the model. This can then be used to render the STL-file and visualise the transmission data.

The string input required to control the RGB values, uses muparser, a math parser library for C++ applications, which contains many of the required functions, such as trigonometric and comparator statements, and offers a starting point for the implementation of the polynomials in MeshLab.

After multiple trial runs of generating different patterns on the STL-files to understand the functionality of the filter as shown in Figure 12, a MATLAB script was generated that created a string of characters that would function as the input to the filter, given the position and the normal of the transmitter.

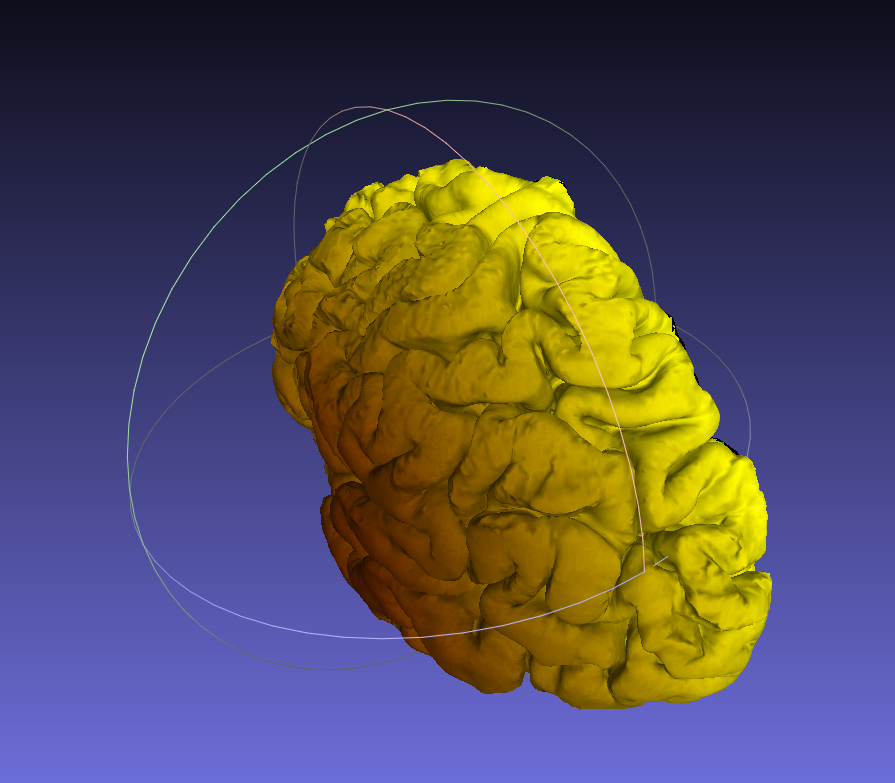


Figure 12: Output of the 'per face Function' filter after processing the left-hand side of the exemplary brain surface with r= 255\*x1/100 and g=255\*y1/75

This input applied polynomials, that were fitted to the data with 2 degrees of freedom in both x and y to reduce the complexity of the terms required, to the STL faces. This significantly reduced the recalculation time from around an hour for both halves to 10.6s, for the recalculation of the colour of the 572,298 faces of the exemplary model.

The same filter was then applied with the calculated polynomials of higher degrees and using robust fitting methods and an execution time of the around 11s was found. This small increase of execution time was not proportional to the increase in complexity of the polynomial by 14 coefficients. The reason for the unproportioned increase was found to be that MeshLab was rounding the numbers that are smaller than 0.0001 to 0 to reduce processing time. As such the polynomials used to assign the transmission scores were decided to be left at a complexity of 2 degrees in either direction.

## Visualisation using 3D-printing/Colour Gradient

A different approach taken to visualise the data was the 3D-printing of the results.   
This was considered as a suitable way to communicate the results due to its ease of communicating the results to the user/surgeon that would be able to take the model of the brain into surgery to check the transmission regions when performing the surgery and choosing an area of implantation.

3D-printing the results in different colours failed to be accomplished due to the complexity of the different regions associated with the different colours to be used.   
Considering an expected result such as the colour scheme shown in Figure 11, the model of the brain would have to printed using a standard layer by layer printing approach by moving along either one of the 3D-axes. In this case the best way to print would be moving along the y-axis in positive direction, so that the ‘no transmission’ region may be printed first, while adding on the differently coloured layers, when the ‘lossy’ and ‘lossless’ regions were reached. This meant having to swap the filament, being used to visualise the regions, multiple times in one layer towards the right side of the model, to achieve an accurate representation of the results. This in turn meant a high degree of loss of detail and complexity of the communicated results.  
The implementation of a 3D-printed version to visualise the results effectively shall be considered if a 3D-printer that allows the fixed use of three different colours can be constructed or used, which was found to be outside the scale of this project.

The most suitable material used for 3D-printing in this application is PLA-filaments, which offer a wide variety of colours, dimensional accuracy and are available at a low cost. The result would be a hard-plastic model of the brain topology that would ideally be sterilised and taken into surgery.   
The sterilisation of the model can be accomplished using the Ethylene Oxide (EtO) Process. EtO is used to sterilize medical products, which are not suitable to undergo the commonly employed high temperature steam sterilization [5]. EtO gas is effective at sterilising the material as it infiltrates both the package and the product and kills microorganisms effectively due to its toxic nature [5]. As such safety measures must be taken when applying the sterilisation process of Pre-Conditioning, Sterilizing and Degassing the used chamber. A process like the one to be employed when a suitable model is generated, was used by Hackaday.io as part of the Hospital + Makerspace Initiative to sterilize PLA Plastic models for Medical Use [6].

Another method considered to be employed to visualize the results of the transmission score transmission, was to implement a scale from ‘no transmission’ to ‘lossy’, that would use a colour gradient to implement the colouration of the different faces. This was realized as a prototype using a score that split the distance between the different transitory on axis values of the LL and LN transition in four different regions to apply different colours. The implementation of these added regions, lead to a significant increase in processing time due to the increased number of comparator statement. As such the implementation of a version displaying these results was removed from the final version.

# Results

Applying the transmission data to an exemplary STL-file containing the brain topology of a patient generated from an MRI-scan was found to yield two different strategies for visualisation.

The first being a calculation of the transmission quality results using MATLAB only, which provides a very reliable result with an adj. R-squared value of 0.9783 for the LL-transitory values and 0.9939 for the LN-transitory values. As such the model explains more than 97% of the variation of the on-axis drop-off distance. In addition, a RMSE value of 0.8643 for the LL-transitionary values and a RMSE value of 0.7330 for the LN-transitionary values meant that the polynomial estimated the transmission quality results correctly with a standard deviation of less than one millimetre in the on-axis displacement measure.

The second method, using the ‘poly22’ polynomials as input for the MeshLab ‘Per Face Colour Function’ filter, provides a significantly reduced calculation time and estimates the model transmission qualities correctly with an adj. R-sq. value of 0.9692 for the LL-transitory values and 0.9767 for the LN-transitory values. The RMSE values were found to be 1.0292 for the LL-transitory values and 1.4279 for the LN-transitory values.

Therefore, the second method does provide results which are less accurate but allow the user to recalculate the results within a reasonable timeframe of less than 11s for a model of the provided complexity. As such to get a rough idea of the transmission quality zones the second method may be used, but the first method of applying the best fit polynomials should be used to verify the calculated results.

Table 3: Comparison of the goodness of fit results for the two different sets of polynomials used

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Transition | Model | SSE | R-squared | Adj.  R-squared | RMSE |
| Lossless - Lossy | ‘poly45’  ‘LAR’ | 36.6046132 | 0.98438072 | 0.97832427 | 0.86431066 |
| Lossy - No Transmission | ‘poly45’ ‘Bisquare’ | 30.6242129 | 0.99539757 | 0.99386343 | 0.73298492 |
| Lossless - Lossy | ‘poly22’  ‘LAR’ | 66.7365098 | 0.97152337 | 0.96926332 | 1.02922771 |
| Lossy - No Transmission | ‘poly22’  ‘LAR’ | 144.764163 | 0.9782438 | 0.97671167 | 1.42791172 |

# Code execution

To use the full functionality of both methods created to visualise the transmission quality results a few MATLAB scripts were created, of which a top-level approach of the execution is described.

## Reading STL-files

To be able to apply the polynomial functions to the faces encoded by the vertices, the file firstly must be read correctly. The function ‘readStl’ takes an STL-file name as input and firstly checks if this file exists. If it does it checks which encoding the file is using, to use the correct skeleton format to read in the data stored in the STL file. This data is either encoded in ASCII or in Binary. This can be differentiated using the fact that ASCII encoded files the first characters must be ‘solid’ and the last characters of the file have to be ‘endsolid’, whereas binary encoding begins with the data of the first vertices immediately [7] , [8].

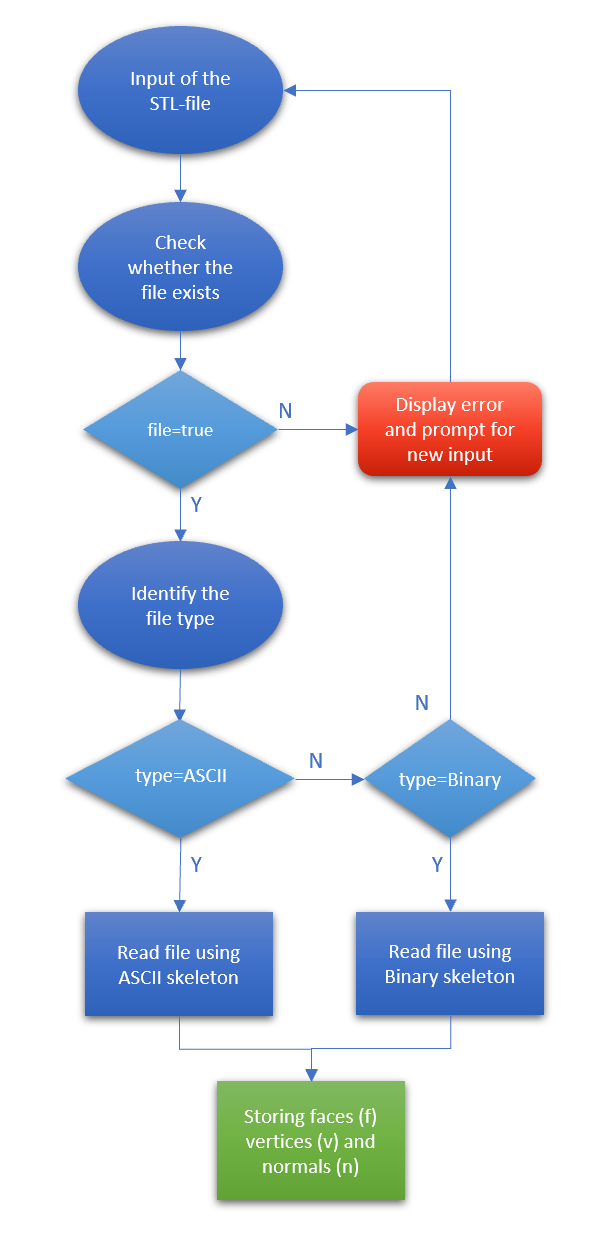


Figure 13: Flowchart showing the correct reading of an STL-file into MATLAB

## Calculation of the transmission scores

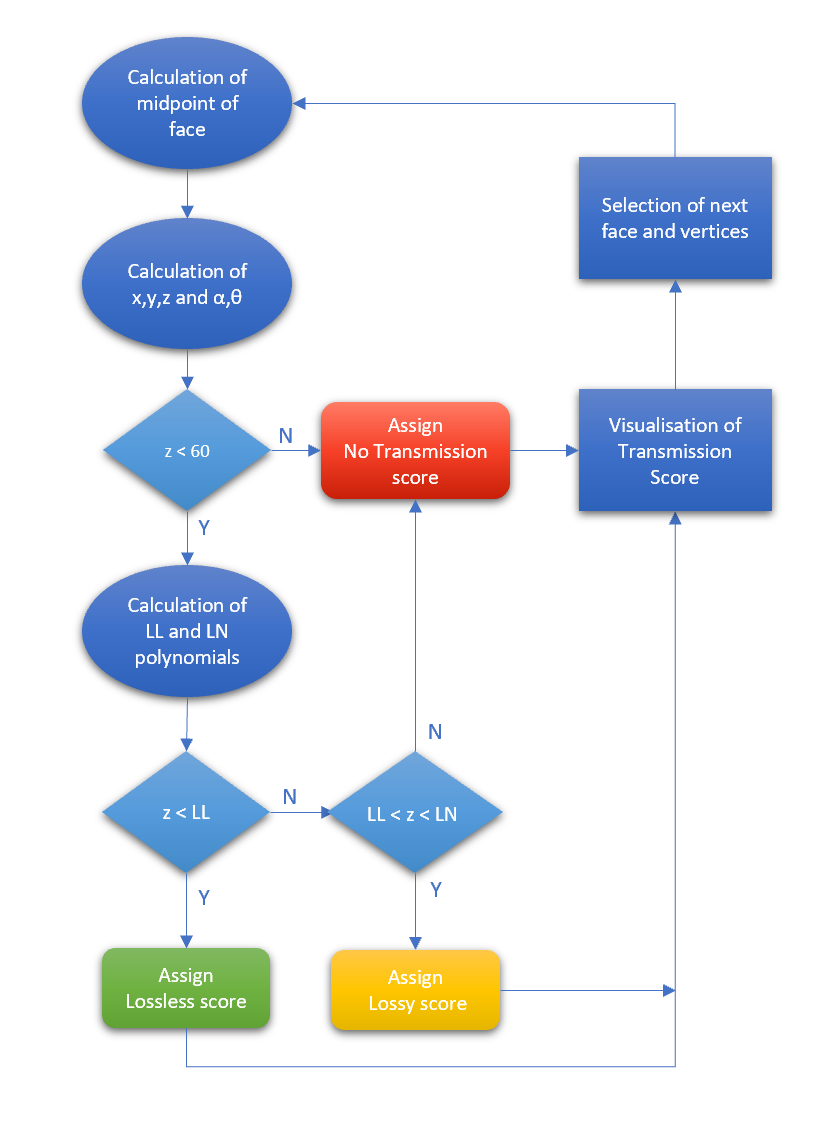
The code functionality of the calculation of the different transmission scores can be broken up into three main parts. The calculation of the variables, the assigning of the transmission quality score and the visualisation of the results. The mathematics behind the calculation of the variables associated with each face were discussed in Section 3.1, along with the generation of the polynomials that are used to assign the transmission quality score. The visualisation of the results depends on the software being used.

Figure 14: Flowchart showing the assignment of transmission scores

## Saving STL-files in colour

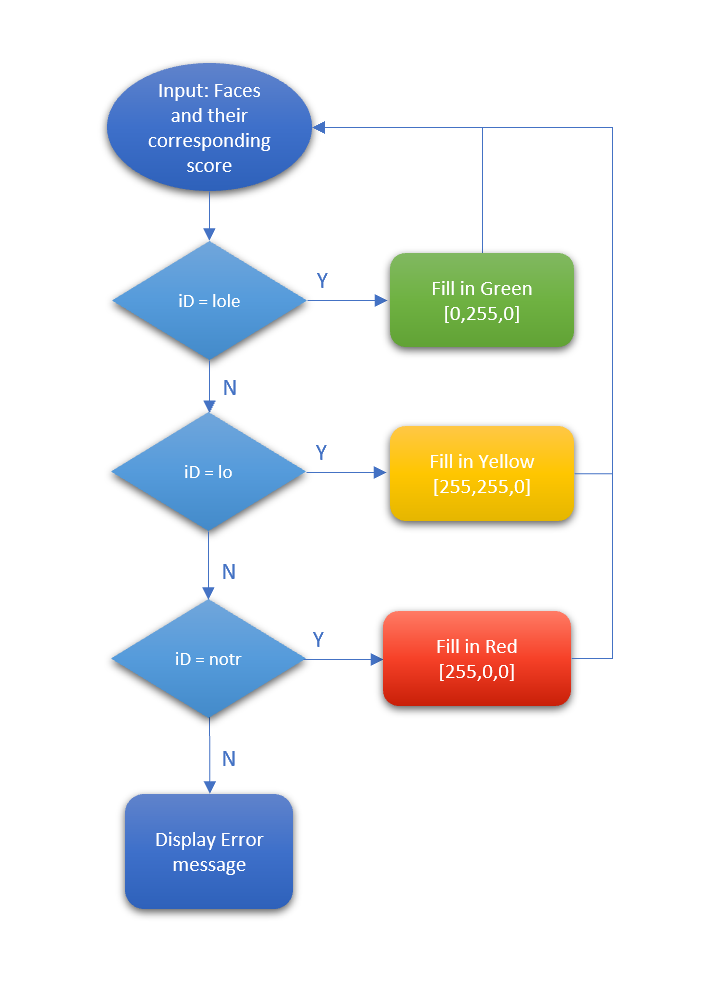
To visualize the different transmission quality regions of models with a high complexity, three different STL-files had to be saved with three different colours to indicate their associated transmission score. The saving of the different STL-files was accomplished using a function created by Sven Holcombe in 2011 using binary encoding [9].

Figure 15: Flowchart showing the saving of different STL-files in their corresponding colour where lole = 'lossless', lo = 'lossy and notr = 'no transmission'

## Recalculation of the transmitter position

After updating the transmitter position in MeshLab, the recalculation of the transmission quality scores is dependent on a new transmitter position and normal. As such the new position and normal will have to be calculated using the transformation matrix, which is generated by MeshLab when a Mesh is moved. As such to calculate the new variables associated with the transmitter the MATLAB script which is used to calculate the new transmission scores firstly checks whether a transformation has occurred and then if it does detect a change in the position of the transmitter computed the new variables by convolving the transformation matrix with the vectors which store the original position and orientation of the transmitter. To check whether a transformation of the transmitter has occurred the script reads in the MeshLab project file and finds the transformation matrix associated with the transmitter.

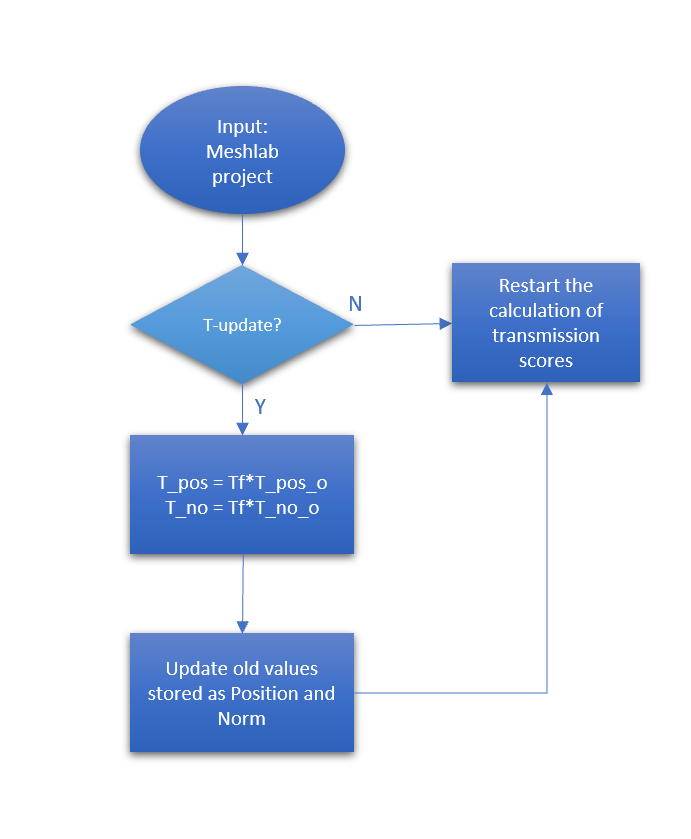


Figure 16: Flowchart showing the recalculation of the transmission scores based on an update from the Transmitter orientation or position where  
 T\_pos = Transformer position, Tf = Transformation matrix, T\_no = Transmitter norm and \_o indicates the old position

# Requirements for use

To use the generated MeshLab filter that allows the recalculation of the transmission score results, the fundamental requirement is that the computer must have MeshLab installed and use a graphics card that is powerful enough to display 3D-rendered models. To generate an updated result of the transmission scores based on a different transmitter position the computer must have MATLAB installed as well.

## Requirements for the imaging techniques

As the resolution of the results provide accurate calculation of the results within a standard deviation of less than 1.5mm, a resolution that is as high or higher would be ideal.

This can be achieved with all of the most commonly used imaging techniques for the soft tissue of the brain.   
The most commonly used imaging techniques are a Computed Tomography (CT) scan and Magnetic Resonance Imaging (MRI) scan, wherein the CT scan is applied far more commonly, due to its reduced cost in comparison to the MRI [10]. Their main difference is the method used to generate the image as the CT relies on the use of radiation, while the MRI uses magnetic resonance to create its image, as the name implies [10]. Other imaging techniques such as a Positron emission Tomography (PET) scan or a functional near infrared spectroscopy (fNIRS), add different benefits such as being able to analyse the different brain fluids and showing promising results in being able to replicate the brain topology consistently, indicating a high level of accuracy [11].

The best imaging technique, among the Medical Imaging techniques used to analyse the brain topology, was identified to be MRI, due to its high resolution and superior performance of displaying soft tissue structure such as the brain topology [10].   
Another significant factor for choosing MRI as imaging technique for the project is the already existing and development of algorithms specifically for MRI scans, which allow the reconstruction of the brain topology as a 3D-model of the brain [12]. A development of a similar set of software that can be applied to other imaging techniques such as fNIRS is likely to be implemented soon but does not exist currently at the extent it does for MRI-techniques. These algorithms allow for an accurate reconstruction of the outer cortices brain topology and allows for the construction of an STL-file suitable as the input for the project. As such as the resolution requirement is fulfilled, MRI scans provide the best imaging technique for this specific application.

## Requirements for the STL-file input

The STL-file must be in a 1:1 scale to ensure the correct sizing of both the model and the transmitter. This will ensure that the calculation of the transmission scores is correctly performed. In addition, the size of the STL-file shall be within a range of 250,000-750,000 faces, which ensures that the resolution of the brain surface model is within an acceptable range, as well as limiting the run time of the recalculation of the model face scores.   
If the file is found to be outside the considered range, the protocol for reducing the number of faces in the model described in Section 8.1 can be applied before processing the model.

Table 4: Overview of the requirements specified for the processed STL input file

|  |  |
| --- | --- |
| Requirements for the STL-file input | |
| Scale | 1:1 |
| No. of faces | 250,000-750,000 |
| Encoding | ASCII, Binary |

## Hardware requirements

As the rendering of 3D-models within the discussed size range is computationally expensive the computer that is used to recalculate the transmission scores as well as display the visualized results needs to fulfil some hardware requirements.

The operating system used by the computer is specified to be Windows, MacOS or Linux Snap, to use MeshLab and its inbuilt Filters. To be able to use the MeshLab server commands the operating system has to be Windows, as command prompts are required to be used that do not exist in MacOS or Linux. If MacOS or Linux is being used the protocol described in Section 0 for the calculation of r and g muparser statements using MATLAB must be employed.

To allow for the rendering of the results, MeshLab does not specify a set requirement for the GPU of the used computer, but testing of the code execution on different computers, yielded results that indicated that the newer and more powerful the GPU the quicker the recalculation of transmission scores can be executed.

# Protocol for the replication of the results

To replicate the results described in Section 4.3 and apply them to a new STL-file the following protocol shall be followed.

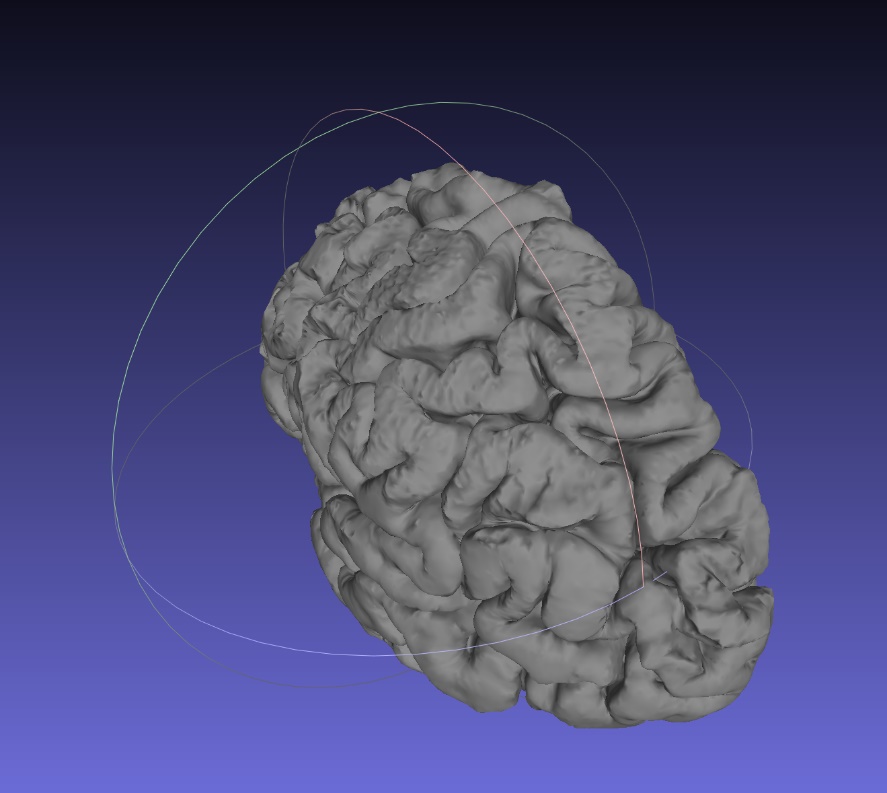
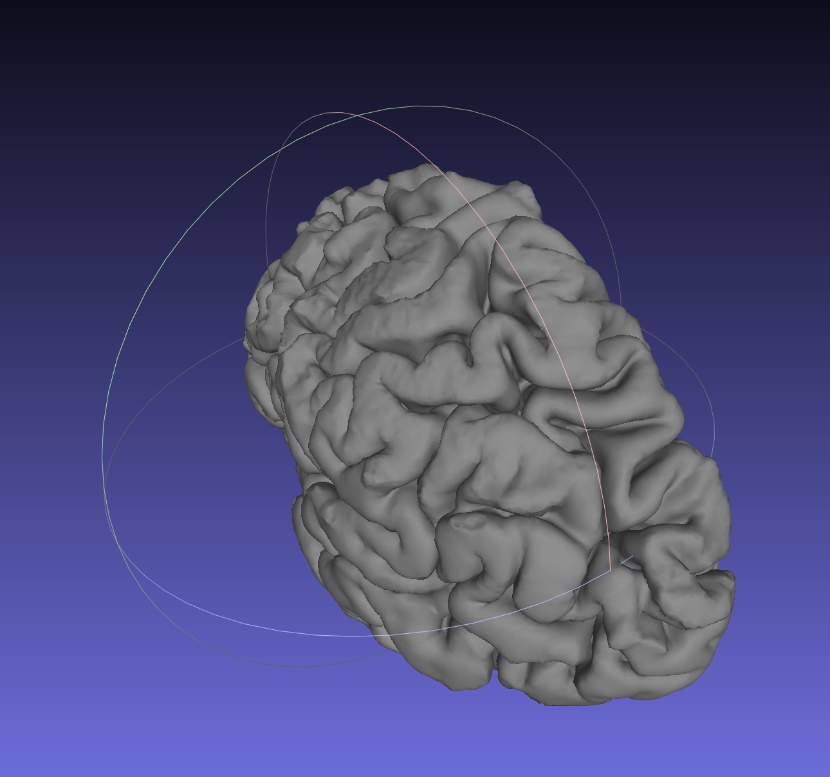
## Pre-processing of a large resolution STL-file

This process is applicable for STL-files which have a resolution larger than 500,000 faces, which is likely to slow down the calculation of the new results significantly and will as such not provide ideal results.

To check the size of the STL-file import it into MeshLab by starting MeshLab and then choosing   
**File->Import Mesh** and choosing the file to be processed. In the bottom right of the UI MeshLab will show a prompt of how many faces it has processed.  
If this number is larger than 500,000 choose the option **Filters -> Remeshing, Simplification and Reconstruction -> Simplification: Quadric Edge Collapse Decimation** from the dropdown menu at the top of the screen.   
Enter 500,000 as the desired **Target number of faces** and checkmark the **Preserve topology** option.  
After choosing to Apply this filter the mesh will be reduced to the target number of faces.

This filter can also be used to reduce the time of executing the MATLAB script to find the final assignment of the transmission scores to different faces but will reduce the resolution quality of the model as shown in Figure 17.

Figure 17: Visible loss of detail through Simplification using the Quadric Edge Collapse Filter and reducing   
the number of faces from 286210 (left) by 50% to 143105 faces (right)



## Calculation of transmission scores

To calculate the transmission scores associated with all the different faces in the model the transmitter position and the transmitter normal must be known.   
If the position of the transmitter is currently unknown, follow the first two steps described in Section 8.3.

### MATLAB Calculation of transmission scores

To calculate the transmission scores using MATLAB and the associated ‘poly45’ models, the STL-file to be processed should be saved in the same folder that the MATLAB files are saved in. The folder should also contain the MATLAB backup of the polynomials called ‘polynomial.mat’ and the function to write STL-files called ‘stlwrite.m’. The process to create a different polynomial to reproduce different results based on different collected transmission data is explained in Section 8.4.

Process the files by opening MATLAB and the .m file called **‘calculate\_transmission\_scores.m’** and running it by using the MATLAB editor or ‘F5’. The code will prompt the user for the name of the STL-file to be processed.

The **input file directory** is required to have parenthesis around it to signify a string input, or it will otherwise return an error message.

The user is then prompted to input the **transmitter position and its normal** of the form of a horizontal vector [x,y,z].

Once these three inputs have been specified the code will automatically resume execution and evaluation of every face within the STL-file and assigning it its corresponding transmission value.   
The output of the code are three different STL-files that are saved in the colour corresponding to the transmission score.

The three output files created are:

* lole.stl which contains all the faces with a quality score of ‘lossless’ saved in green
* lo.stl which contains all the faces with a quality score of ‘lossy’ saved in yellow
* notr.stl which contains all the faces with a quality score of ‘no transmission’ saved in red/white

To display the results, open all three meshes in MeshLab by opening MeshLab and choosing   
**File->Import Mesh.**To display the position of the transmitter, load the ‘transmitter.stl’ file into MeshLab and save the Meshes as a MeshLab project into the same folder.

### MeshLab Calculation of transmission scores

To calculate the transmission scores using the MeshLab filter, the STL-file to be processed should be saved in the same folder that the MATLAB files are saved in. The folder should also contain the MATLAB backup of the 22-polynomials called ‘polynomial\_22.mat’. The process to create a different polynomial to reproduce different results based on different collected transmission data is explained in Section 8.4.

To calculate the transmission scores using the MeshLab filter open MeshLab. Import the Mesh to be processed and import the desired STL file to be processed.

Import the **‘transmitter.m’** file and position it at the desired position using the translate/transform option.

Once the transmitter is in the correct position save the Meshes as a MeshLab project.

Open MATLAB and open the file called **‘meshlab\_string\_writing.m’**.

Follow the prompts by specifying if the transmitter position has been updated and modifying it if necessary.

Choose what colour scheme you want the output of the filter to be as shown in Figure 18. Once the transmitter position is known the code will generate three strings called r,g,b and use these to calculate a .mlx filter script.

Apply the generated Script file by choosing **Filter->Show current Filter Script->Open Script** and selecting the filter that was created by the MATLAB file**.**

Click apply and the model should be updated in the selected colour scheme within seconds.

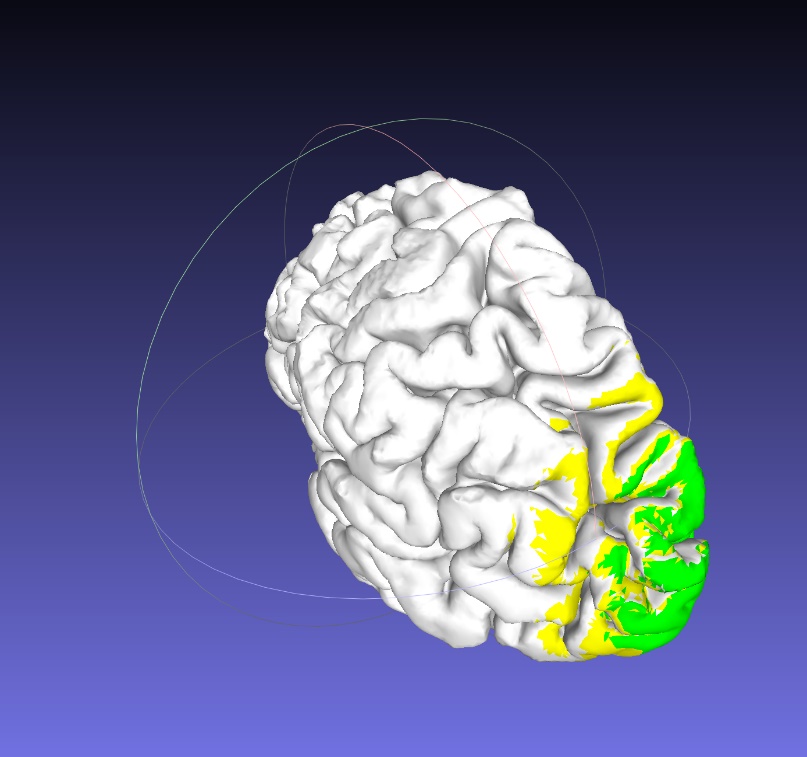
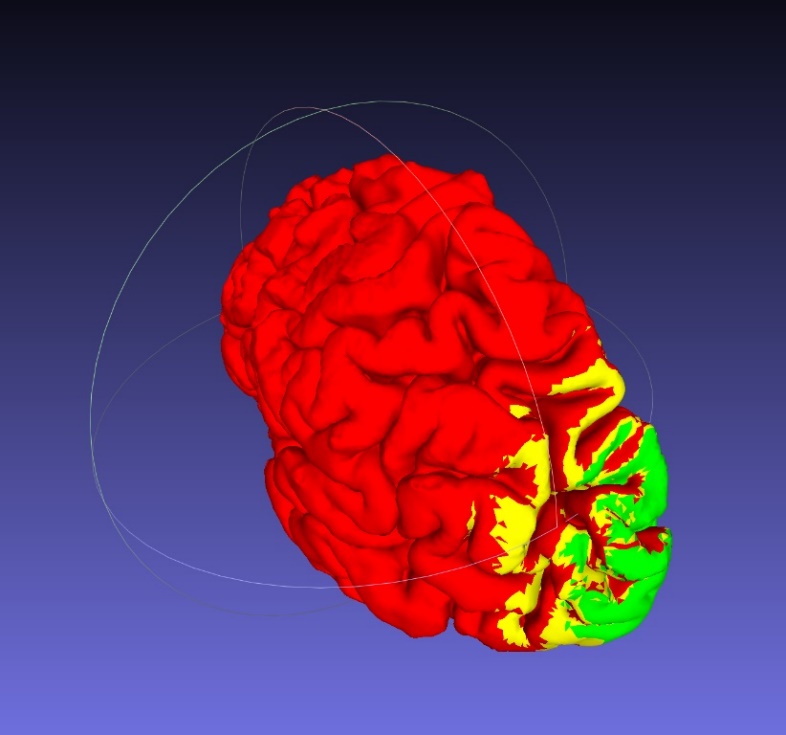


Figure 18: Comparison between the two different color schemes employed with [red/yellow/green] on the left and [white/yellow/green] on the right where both schemes correspond to [no transmission/lossy/lossless]

## Recalculation of transmission scores

If the transmitter position and normal is currently unknown, import the processed STL-files into MeshLab first and then import the **‘transmitter.STL’** file in MeshLab and transform/translate the transmitter to the desired position.

Save the MeshLab project within the same folder.

To recalculate the Transmission scores with a new transmitter position, open the MATLAB file called **‘recalculate\_transmitter\_pos’**. This file will prompt the user to then input the name of a MeshLab project file. Specify the directory using parenthesis.

The transformation matrix saved in the MeshLab project will be used to calculate the new transmitter and transmitter normal and save these variables into **‘transmitter.mat’**.

These values can be used to apply the Calculation of transmission scores by using either of the protocols described in Section 8.2.

## Recalculation of polynomials

To recalculate the polynomials using a different data set collected, firstly open MATLAB and the MATLAB script **‘datafitting.m’**.

Save the data in a text file called ‘data.txt’ or adjust the file directory specified in the ‘datafitting.m’ file accordingly. The data shall be in the same format specified in the document outlining the collection of the data points [2].

Specify the name of the data file in which you want to save the new polynomials.   
Use ‘polynomials’ for a recalculation using the MATLAB script using any degree of freedom along x and y desired.   
Use ‘polynomials\_22’ for a recalculation of the transmission using the MeshLab filter and specify ‘poly22’. Any number of additional degrees of freedom will be ignored when the MATLAB script to write the muparser R,G,B-strings are generated.

Specify the desired robust fitting method using the variables fit\_lole and fit\_lo.

Verify the correct fit of the data points with the generate plots.

To generate the same evaluation of the goodness of fit open the Curve fitting tool in the Apps tab of MATLAB. Iterate the number of degrees of freedom used for the polynomials and observe the changing of the goodness of fit. Choose the model which provides the highest Adj. R-squared score as well as the lowest RMSE values, indicating a better data fit.

# Reflection and ideas for further improvement

The project produced a suitable and easy to use application to visualise the transmitter range on the patient specific brain topology of a patient. It fulfilled all the high-level and data analysis requirements specified in the Requirement Analysis [13]. The project fulfilled all the requirements for the Mapping of the Visual Cortex as well as the Creation of STL-files. The optional requirement of generating a spectrum score to remove the confinement of the three different transmission scores was achieved but chosen to not be implemented, as discussed in Section 4.3. As outlined in the same section, the requirement R.05.3 was not completed due to problems associated with the printing of the 3D-model in three different colours.

To improve the project an embedding in MeshLab, which removes the need for MATLAB processing of the data following an updated Transmitter position, would significantly improve the ease of use of the program. This may be achieved using MeshLab server, as soon as MeshLab supports an import of STL files saved with an individual face colour.

Similarly, the research of an alternative technique for visualizing the results during the surgery, such as the generation of a 3D-printed of the topology, as soon as suitable alternative to the traditional layer-by-layer approach becomes available, may improve the project.

# Conclusion

The project led to development of a strategy to analyse the provided transmission data and the generation of polynomials encoding these. Two different methods of assigning the transmission scores to the STL-files were generated and visualised in MeshLab. Both methods yield results that are more than 96% accurate, given the provided data, while one of the methods yields faster results at the trade-off of 1% of accuracy.   
The implementation of the two strategies was completed using MATLAB and MeshLab to generate a effective visualisation of the result that allows the user to easily identify different brain topology regions of transmission quality. The resulting software may be used as a guidance for the implantation of the receiving microelectrode tiles as part of the Monash Vision Group project.

Link to Code repository:

<https://github.com/cdie3/FYP>

Link to Video:

<https://www.youtube.com/watch?v=al_0mTDmdN0&t=462s>

# References

|  |  |
| --- | --- |
| [1] | M. University, „Bionic Vision,“ [Online]. Available: https://www.monash.edu/mime/research/flagship-projects/bionic-vision. [Zugriff am 10 08 2018]. |
| [2] | T. Feleppa, „Tile Transmission Range,“ Monash Vision Group, Melbourne, 2016. |
| [3] | Z. Wang, Y. Liu, Y. Wei und Y. Song, „Study on electromagnetic characteristics of the magnetic coupling resonant coil for the wireless power transmission system,“ *Journal of Applied Biomaterials & Functional Materials,* Bd. 16, Nr. 1, pp. 140-149, 2018. |
| [4] | MathWorks, „Evaluating Goodness of Fit,“ MAthWorks, 2019. [Online]. Available: https://www.mathworks.com/help/curvefit/evaluating-goodness-of-fit.html#bq\_5kwr-3. [Zugriff am 30 May 2019]. |
| [5] | Eurotherm by Schneider Electric, „Ethylene Oxide (EtO) Sterilization Process,“ 2016. [Online]. Available: https://www.eurotherm.com/life-sciences-cpg-processes-applications-us/ethylene-oxide-eto-sterilization-process/. [Zugriff am 30 May 2019]. |
| [6] | Progress TH - Hackada.io, „3D Printing: Sterilizing PLA Plastic for Medical Use,“ July 2016. [Online]. Available: https://hackaday.io/project/12687-hospital-makerspace-initiative/log/42643-3d-printing-sterilizing-pla-plastic-for-medical-use. [Zugriff am 30 May 2019]. |
| [7] | C. Iancu, D. Iancu und A. Stancioiu, „From CAD Model to 3D print via "STL" file format,“ EBSCO, 2019. |
| [8] | D. Harlev, „stlread,“ MathWorks, 2005 January 2005. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/6678-stlread. [Zugriff am 2019 May 2019]. |
| [9] | S. Holcombe, „stlwrite - write ASCII or Binary STL files,“ MATLAB, 04 May 2018. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/20922-stlwrite-write-ascii-or-binary-stl-files. [Zugriff am 30 May 2019]. |
| [10] | E. Carver und B. Carver, Medical imaging : Techniques, reflection & evaluation, 2 Hrsg., Oxford: Churchill Livingstone, 2012. |
| [11] | H. Niu, J. Wang, T. Zhao, N. Shu und Y. He, „Revealing Topological Organization of Human Brain Functional Networks with Resting-State Functional near Infrared Spectroscopy,“ *PLoS One,* Bd. 7, Nr. 9, 2012. |
| [12] | D. L. Pham, P. L. Bazin und J. L. Prince, „Digital Topology in Brain Imaging,“ *IEEE Signal Processing Magazin,* Bd. 27, Nr. 4, pp. 51-59, 2010. |
| [13] | C. Diederichsen, „Requirement Analysis,“ Monash Uni, Melbourne, 2019. |
| [14] | A. J. Lowery, J. V. Rosenfeld, P. M. Lewis, D. Browne, A. Mohan, E. Brunton, E. Yan, J. J. Maller, C. Mann, R. Rajan, M. G. P. Rosa und J. Pritchard, „Restoration of vision using wireless cortical implants: The Monash Vision Group project,“ *Annual International Conference of the IEEE Engineering in Medicine and Biology Society,* 2015. |

# Appendix

## Appendix A - Poster Submission

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

## Appendix B – Data Fitting – Goodness of fit results

Lossless-Lossy (LAR)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| polyij | SSE | R-square | DFE | Adj R-sq | RMSE | Coeff |
| poly11 | 2310.04002 | 0.01430021 | 66 | -0.0155695 | 5.91613104 | 3 |
| poly21 | 104.100101 | 0.95558023 | 64 | 0.95280399 | 1.27536821 | 5 |
| poly31 | 154.22535 | 0.93419166 | 62 | 0.92782311 | 1.57718282 | 7 |
| poly41 | 182.523221 | 0.92211689 | 60 | 0.91173248 | 1.74414841 | 9 |
| poly51 | 188.920224 | 0.91938727 | 58 | 0.90548853 | 1.80478399 | 11 |
| poly12 | 2044.32663 | 0.12768077 | 64 | 0.07316082 | 5.6517788 | 5 |
| poly22 | 66.7365098 | 0.97152337 | 63 | 0.96926332 | 1.02922771 | 6 |
| poly32 | 76.8507909 | 0.96720758 | 60 | 0.96283525 | 1.1317449 | 9 |
| poly42 | 80.4781978 | 0.96565975 | 57 | 0.95903269 | 1.18823323 | 12 |
| poly52 | 46.6765419 | 0.980083 | 54 | 0.97491934 | 0.92972061 | 15 |
| poly13 | 2080.50189 | 0.1122447 | 62 | 0.0263329 | 5.79279571 | 7 |
| poly23 | 70.0429615 | 0.9701125 | 60 | 0.96612749 | 1.08045485 | 9 |
| poly33 | 73.1240854 | 0.96879777 | 59 | 0.96403811 | 1.11327951 | 10 |
| poly43 | 70.9999114 | 0.96970416 | 55 | 0.96254333 | 1.13618109 | 14 |
| poly53 | 36.2815597 | 0.98451857 | 51 | 0.97935809 | 0.84344717 | 18 |
| poly14 | 1973.22461 | 0.15802018 | 60 | 0.04575621 | 5.73472552 | 9 |
| poly24 | 79.3181986 | 0.96615473 | 57 | 0.95962318 | 1.17963865 | 12 |
| poly34 | 76.0893958 | 0.96753247 | 55 | 0.95985832 | 1.17619878 | 14 |
| poly44 | 73.4601949 | 0.96865435 | 54 | 0.9605277 | 1.16635071 | 15 |
| poly54 | 36.7553758 | 0.98431639 | 49 | 0.97823499 | 0.86608874 | 20 |
| poly15 | 2009.71082 | 0.14245143 | 58 | -0.0054018 | 5.88644091 | 11 |
| poly25 | 57.410357 | 0.97550286 | 54 | 0.96915175 | 1.03109396 | 15 |
| poly35 | 48.3616508 | 0.97936396 | 51 | 0.97248529 | 0.97379036 | 18 |
| poly45 | 36.6046132 | 0.98438072 | 49 | 0.97832427 | 0.86431066 | 20 |
| poly55 | 37.084355 | 0.98417601 | 48 | 0.97758269 | 0.8789714 | 21 |

Lossless-Lossy (Bisquare)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| polyij | SSE | R-square | DFE | Adj R-sq | RMSE | Coeff |
| poly11 | 2072.27051 | 0.11575705 | 66 | 0.08896181 | 5.60339523 | 3 |
| poly21 | 261.207888 | 0.88854195 | 64 | 0.88157582 | 2.02024089 | 5 |
| poly31 | 279.992852 | 0.88052636 | 62 | 0.86896439 | 2.12509147 | 7 |
| poly41 | 268.768005 | 0.88531603 | 60 | 0.87002483 | 2.11647508 | 9 |
| poly51 | 268.337563 | 0.8854997 | 58 | 0.86575827 | 2.15093229 | 11 |
| poly12 | 1817.80514 | 0.22433805 | 64 | 0.17585918 | 5.32946576 | 5 |
| poly22 | 97.6717476 | 0.95832322 | 63 | 0.95501554 | 1.24512859 | 6 |
| poly32 | 94.8173301 | 0.95954121 | 60 | 0.95414671 | 1.2570954 | 9 |
| poly42 | 86.5976901 | 0.96304855 | 57 | 0.95591757 | 1.23258173 | 12 |
| poly52 | 51.0461804 | 0.97821847 | 54 | 0.9725714 | 0.97226521 | 15 |
| poly13 | 1782.39428 | 0.23944796 | 62 | 0.16584615 | 5.36174363 | 7 |
| poly23 | 84.041126 | 0.96413944 | 60 | 0.95935804 | 1.18350557 | 9 |
| poly33 | 84.1041953 | 0.96411253 | 59 | 0.95863817 | 1.19394088 | 10 |
| poly43 | 74.8854024 | 0.96804621 | 55 | 0.9604935 | 1.16685593 | 14 |
| poly53 | 38.304144 | 0.98365553 | 51 | 0.97820737 | 0.86663813 | 18 |
| poly14 | 1776.00508 | 0.24217424 | 60 | 0.14113081 | 5.44059598 | 9 |
| poly24 | 85.7644831 | 0.96340408 | 57 | 0.95634171 | 1.2266377 | 12 |
| poly34 | 81.1453924 | 0.96537506 | 55 | 0.95719098 | 1.21464841 | 14 |
| poly44 | 76.9925577 | 0.96714709 | 54 | 0.95862966 | 1.19406369 | 15 |
| poly54 | 38.1950859 | 0.98370206 | 49 | 0.97738245 | 0.88288819 | 20 |
| poly15 | 1791.77061 | 0.23544705 | 58 | 0.10362758 | 5.55811089 | 11 |
| poly25 | 61.2452891 | 0.97386648 | 54 | 0.96709113 | 1.06497513 | 15 |
| poly35 | 50.8228926 | 0.97831375 | 51 | 0.97108499 | 0.99826214 | 18 |
| poly45 | 38.1079732 | 0.98373923 | 49 | 0.97743404 | 0.8818808 | 20 |
| poly55 | 38.468474 | 0.98358541 | 48 | 0.97674599 | 0.8952243 | 21 |

Lossy-No Transmission (LAR)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| polyij | SSE | R-square | DFE | Adj R-sq | RMSE | Coeff |
| poly11 | 6937.98722 | -0.0426907 | 74 | -0.0708715 | 9.68279836 | 3 |
| poly21 | 162.358633 | 0.97559958 | 72 | 0.974244 | 1.50165942 | 5 |
| poly31 | 230.879334 | 0.96530179 | 70 | 0.96232766 | 1.81611569 | 7 |
| poly41 | 295.927548 | 0.95552588 | 68 | 0.95029363 | 2.08611498 | 9 |
| poly51 | 343.732487 | 0.94834141 | 66 | 0.94051435 | 2.28211919 | 11 |
| poly12 | 6848.40388 | -0.0292274 | 72 | -0.0864067 | 9.75278014 | 5 |
| poly22 | 144.764163 | 0.9782438 | 71 | 0.97671167 | 1.42791172 | 6 |
| poly32 | 116.151895 | 0.98254386 | 68 | 0.98049019 | 1.30694916 | 9 |
| poly42 | 98.460408 | 0.98520266 | 65 | 0.9826985 | 1.23076217 | 12 |
| poly52 | 50.4636346 | 0.99241596 | 62 | 0.99070344 | 0.90218046 | 15 |
| poly13 | 6678.90958 | -0.0037546 | 70 | -0.0897907 | 9.76795751 | 7 |
| poly23 | 113.062238 | 0.98300819 | 68 | 0.98100916 | 1.28944948 | 9 |
| poly33 | 123.829333 | 0.98139004 | 67 | 0.97889019 | 1.35948483 | 10 |
| poly43 | 54.1018723 | 0.99186918 | 63 | 0.99019139 | 0.92669298 | 14 |
| poly53 | 34.9317427 | 0.99475021 | 59 | 0.99323755 | 0.76945659 | 18 |
| poly14 | 6875.00479 | -0.0332252 | 68 | -0.1547811 | 10.0549993 | 9 |
| poly24 | 165.350289 | 0.97514997 | 65 | 0.97094458 | 1.59494533 | 12 |
| poly34 | 151.129439 | 0.97728718 | 63 | 0.97260041 | 1.54883181 | 14 |
| poly44 | 55.8328176 | 0.99160904 | 62 | 0.98971431 | 0.94896223 | 15 |
| poly54 | 37.6346325 | 0.994344 | 57 | 0.99245866 | 0.81256182 | 20 |
| poly15 | 7041.4073 | -0.0582334 | 66 | -0.2185718 | 10.3289878 | 11 |
| poly25 | 78.1916618 | 0.98824879 | 62 | 0.9855953 | 1.12301195 | 15 |
| poly35 | 112.719052 | 0.98305977 | 59 | 0.97817869 | 1.38220563 | 18 |
| poly45 | 38.4036177 | 0.99422843 | 57 | 0.99230457 | 0.82082134 | 20 |
| poly55 | 39.4165889 | 0.99407619 | 56 | 0.99196055 | 0.83896821 | 21 |

Lossy-No Transmission (Bisquare)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| polyij | SSE | R-square | DFE | Adj R-sq | RMSE | Coeff |
| poly11 | 6488.8147 | 0.02481423 | 74 | -0.0015421 | 9.36411689 | 3 |
| poly21 | 440.840062 | 0.93374738 | 72 | 0.93006668 | 2.47442491 | 5 |
| poly31 | 469.786039 | 0.92939717 | 70 | 0.9233455 | 2.59060401 | 7 |
| poly41 | 379.995064 | 0.94289161 | 68 | 0.93617297 | 2.3639295 | 9 |
| poly51 | 411.608842 | 0.93814046 | 66 | 0.9287678 | 2.49729806 | 11 |
| poly12 | 6371.76705 | 0.04240499 | 72 | -0.0107947 | 9.4072719 | 5 |
| poly22 | 229.153982 | 0.96556109 | 71 | 0.96313582 | 1.79653023 | 6 |
| poly32 | 160.365645 | 0.9758991 | 68 | 0.9730637 | 1.53568171 | 9 |
| poly42 | 125.504389 | 0.9811383 | 65 | 0.97794632 | 1.38954552 | 12 |
| poly52 | 60.2923377 | 0.99093883 | 62 | 0.98889276 | 0.98613237 | 15 |
| poly13 | 6428.22279 | 0.03392042 | 70 | -0.0488864 | 9.58288861 | 7 |
| poly23 | 159.568081 | 0.97601896 | 68 | 0.97319766 | 1.53185816 | 9 |
| poly33 | 156.565884 | 0.97647015 | 67 | 0.97330943 | 1.52866093 | 10 |
| poly43 | 60.1015047 | 0.99096751 | 63 | 0.98910367 | 0.97672521 | 14 |
| poly53 | 39.3334219 | 0.99408869 | 59 | 0.99238543 | 0.8164975 | 18 |
| poly14 | 6489.1637 | 0.02476178 | 68 | -0.0899721 | 9.76877054 | 9 |
| poly24 | 131.805317 | 0.98019135 | 65 | 0.97683912 | 1.42399933 | 12 |
| poly34 | 133.918863 | 0.97987371 | 63 | 0.97572067 | 1.45797676 | 14 |
| poly44 | 58.8907707 | 0.99114947 | 62 | 0.98915096 | 0.97460307 | 15 |
| poly54 | 31.4067898 | 0.99527996 | 57 | 0.99370662 | 0.74229126 | 20 |
| poly15 | 6576.71344 | 0.01160417 | 66 | -0.1381528 | 9.98234307 | 11 |
| poly25 | 97.1385403 | 0.98540132 | 62 | 0.98210485 | 1.25169911 | 15 |
| poly35 | 83.3594043 | 0.98747215 | 59 | 0.98386243 | 1.18864261 | 18 |
| poly45 | 30.6242129 | 0.99539757 | 57 | 0.99386343 | 0.73298492 | 20 |
| poly55 | 30.4318609 | 0.99542648 | 56 | 0.99379308 | 0.73717439 | 21 |

## Appendix C – Code

MATLAB Code – Datafitting

%datafitting

clear

clc

close all

save\_name = 'polynomials';

poly\_form = 'poly45';

fit\_lole = 'LAR';

fit\_lo = 'Bisquare';

%import data from excel spreadsheet

fileID = fopen('data.txt');

A = fscanf(fileID,'%f');

array = zeros((length(A)/11),11);

for i = 1:length(A)/11

array(i,:) = A((i+((i-1)\*10)):(i+10+((i-1)\*10)));

end

data = array;

%define variables

p\_val = zeros(2,3,7);

angle\_lole =[]; on\_axis\_lole=[]; off\_axis\_lole =[];

angle\_lo =[]; on\_axis\_lo=[]; off\_axis\_lo =[];

cnt = 1;

for k = 0:5:30 %make a plot for each increment of x, 0 to 30mm

%copy data for temporary working

data2 = data;

%delete rows where x value is not of interest

data2(data2(:,2)~=k | isnan(data2(:,2)),:) = [];

%copy rows to produce symmetrical results for negative angles if x=0

if(k==0)

sz = size(data2);

for j = 1:sz(1)

sz2 = size(data2);

if(data2(j,1)~=0)

data2(sz2(1)+1,:) = data2(j,:);

data2(sz2(1)+1,1) = -data2(sz2(1)+1,1);

end

end

end

%sort rows by angle

[Y,I] = sort(data2(:,1));

data2 = data2(I,:);

%populate x axis data

xx = data2(:,1);

%set maximum z value

zmax = 60;

%populate y axis data for each x axis value - five bar length values:

%[unknown, lossless, lossy, dropout, unknown]

%check size of data array

sz = size(data2);

yy = [];

for j = 1:sz(1)

if(data2(j,10)==0 || data2(j,10)==1)

if(data2(j,10)==0)

yy(j,1) = 0;

yy(j,2) = data2(j,8);

else

yy(j,1) = data2(j,8);

yy(j,2) = 0;

end

if(data2(j,11)==0)

yy(j,3) = data2(j,9)-data2(j,8);

yy(j,4) = zmax-data2(j,9);

yy(j,5) = 0;

else

yy(j,3) = data2(j,9)-data2(j,8);

yy(j,4) = 0;

yy(j,5) = zmax-data2(j,9);

end

elseif(data2(j,10)==2);

yy(j,:) = [0,data2(j,8),0,0,zmax-data2(j,8)];

elseif(data2(j,10)==3);

yy(j,:) = [zmax,0,0,0,0];

elseif(data2(j,10)==4);

yy(j,:) = [20,0,0,zmax-20,0];

elseif(data2(j,10)==5);

yy(j,:) = [data2(j,8),0,0,zmax-data2(j,8),0];

end

end

xtes = -100:1:100;

y2 = yy(yy(:,2)>0,2)+yy(yy(:,2)>0,1);

x2 = xx(yy(:,2)>0);

y3 = yy(yy(:,3)>0,3)+yy(yy(:,3)>0,2)+yy(yy(:,3)>0,1);

x3 = xx(yy(:,3)>0);

cnt = cnt+1;

%populate the variables

angle\_lole = [angle\_lole;x2];

on\_axis\_lole = [on\_axis\_lole;y2];

angle\_lo = [angle\_lo;x3];

on\_axis\_lo = [on\_axis\_lo;y3];

off\_axis\_lole = [off\_axis\_lole;k\*ones(length(x2),1)];

off\_axis\_lo = [off\_axis\_lo;k\*ones(length(x3),1)];

end

%fit the raw data with the best polynomial

sf\_lole = fit([angle\_lole, off\_axis\_lole],on\_axis\_lole,poly\_form,'Robust',fit\_lole)

sf\_lo = fit([angle\_lo, off\_axis\_lo],on\_axis\_lo,poly\_form,'Robust',fit\_lo)

%plot the LL transition surface

figure

plot(sf\_lole,[angle\_lole,off\_axis\_lole],on\_axis\_lole);

xlabel('x')

ylabel('z')

zlabel('y')

title('Lossless - Lossy')

figure

ind = find(on\_axis\_lo==16|on\_axis\_lo==17.2|on\_axis\_lo==19.5);

angle\_lo(ind) =[];on\_axis\_lo(ind) =[]; off\_axis\_lo(ind) =[];

%plot the LN transition surface

plot(sf\_lo,[angle\_lo,off\_axis\_lo],on\_axis\_lo);

xlabel('Angle in degrees');

ylabel('Off axis distance in cm');

zlabel('On axis distance in cm')

title('Lossy - No Transmission')

%plot both transitionary surfaces

figure

h = plot(sf\_lole,[angle\_lole,off\_axis\_lole],on\_axis\_lole);

hold on

i = plot(sf\_lo,[angle\_lo,off\_axis\_lo],on\_axis\_lo);

xlabel('Angle in degrees');

ylabel('Off axis distance');

zlabel('On axis distance')

title('Data fitting of both lossy (yellow) and lossless (green) transmission data points')

set(h(1),'FaceColor','g');

set(i(1),'FaceColor','y');

zlim([10 65])

save(save\_name,'sf\_lole','sf\_lo')

MATLAB Code – String Writing for Meshlab Input

%String writing for meshlab

clear all

close all

clc

%load the surface fit polynomials

load polynomials\_22

%Get the position of the transmitter and its normal as input

tr\_up = char(inputdlg('Has the transmitter position been updated using Meshlab? [y/n] '));

tr\_up = 'y';

if tr\_up == 'y'

tr\_pos = inputdlg('What is the file position of the transmitter file in MeshLab? ');

tr\_pos = str2num(cell2mat(tr\_pos));

mlstruct = parseXML('Example.mlp');

testing = mlstruct(2).Children(2).Children(tr\_pos\*2+2).Children(2).Children.Data ;

transformation = cell2mat(cellfun(@str2num, mat2cell(testing, ones(size(testing,1),1), size(testing,2)),'UniformOutput',0));

%[v,f,n] = stlGetFaces('transmitter\_backup.stl');

transmitter = [0,-100,20]; %transmitter midpoint location

transmitter\_n = [0,10,0]; %transmitter norm

testmat = inv(transformation);

t = transmitter;

tp = transmitter+transmitter\_n;

tpt = tp\*testmat(1:3,1:3)+[transformation(1,4),transformation(2,4),transformation(3,4)];

tt = t\*testmat(1:3,1:3)+[transformation(1,4),transformation(2,4),transformation(3,4)];

transmitter\_n = (tpt-tt)/norm(tpt-tt);

transmitter = transmitter\*testmat(1:3,1:3)+[transformation(1,4),transformation(2,4),transformation(3,4)];

save('transmitter.mat','transmitter','transmitter\_n')

clear mlstruct t testing testmat tp tpt tr\_pos tr\_up transformation tt

else

transmitter = inputdlg('What is the transmitter position in [x,y,z]? ');

transmitter\_n = inputdlg('What is the transmitter normal in [x,y,z]? ');

end

%find polynomial coefficients

p\_lo = coeffvalues(sf\_lo);

p\_lole = coeffvalues(sf\_lole);

%write the strings to calculate the angle and distances

angle\_nn = sprintf("(acos((-fnx\*%.3f+-fny\*%.3f+-fnz\*%.3f)/(sqrt(fnx^2+fny^2+fnz^2)\*sqrt((%.3f)^2+(%.3f)^2+(%.3f)^2))))",transmitter\_n(1),transmitter\_n(2),transmitter\_n(3),transmitter\_n(1),transmitter\_n(2),transmitter\_n(3));

angle = sprintf("(acos((-(fnx-%.3f)\*%.3f+-(fny-%.3f)\*%.3f+-(fnz-%.3f)\*%.3f)/(sqrt((fnx-%.3f)^2+(fny-%.3f)^2+(fnz-%.3f)^2)\*sqrt((%.3f)^2+(%.3f)^2+(%.3f)^2))))",transmitter(1),transmitter\_n(1),transmitter(2),transmitter\_n(2),transmitter(3),transmitter\_n(3),transmitter(1),transmitter(2),transmitter(3),transmitter\_n(1),transmitter\_n(2),transmitter\_n(3));

off\_axis\_dist = sprintf("(sin%s\*(sqrt((%.3f-x0)^2+(%.3f-y0)^2+(%.3f-z0)^2)))",angle,transmitter(1),transmitter(2),transmitter(3));

on\_axis\_dist = sprintf("(abs(cos%s)\*(sqrt((%.3f-x0)^2+(%.3f-y0)^2+(%.3f-z0)^2)))",angle,transmitter(1),transmitter(2),transmitter(3));

%write the strings to calculate the LL and LN distances

poly\_lo = sprintf("%.3f + %.3f\*(180/3.14)\*%s + %.3f\*%s + %.3f\*((180/3.14)\*%s)^2 + %.3f\*%s\*(180/3.14)\*%s+ %.3f\*%s^2",p\_lo(1),p\_lo(2),angle\_nn,p\_lo(3),off\_axis\_dist,p\_lo(4),angle\_nn,p\_lo(5),angle\_nn,off\_axis\_dist,p\_lo(6),off\_axis\_dist);

poly\_lole = sprintf("%.3f + %.3f\*(180/3.14)\*%s + %.3f\*%s + %.3f\*((180/3.14)\*%s)^2 + %.3f\*%s\*(180/3.14)\*%s+ %.3f\*%s^2",p\_lole(1),p\_lole(2),angle\_nn,p\_lole(3),off\_axis\_dist,p\_lole(4),angle\_nn,p\_lole(5),angle\_nn,off\_axis\_dist,p\_lole(6),off\_axis\_dist);

color\_scheme = str2num(cell2mat(inputdlg('To use [white,yellow,green] as colorscheme type 1, to use [red,yellow,green] type 2: ')));

%write the strings used as Meshlab filter input

if color\_scheme == 1

fprintf('[white,yellow,green] color scheme chosen\n')

r = sprintf("255\*(%s>%s)",on\_axis\_dist,poly\_lole);

g = sprintf("255");

b = sprintf("255\*(%s>%s)",on\_axis\_dist,poly\_lo);

clear NA angle tr\_up angle\_nn color\_scheme off\_axis\_dist p\_lo p\_lole poly\_lo poly\_lole on\_axis\_dist sf\_lo sf\_lole transmitter transmitter\_n

elseif color\_scheme ==2

fprintf('[red,yellow,green] color scheme chosen\n')

g = sprintf("255\*(%s<%s)",on\_axis\_dist,poly\_lo);

r = sprintf("255\*(%s>%s)",on\_axis\_dist,poly\_lole);

b = sprintf("0");

clear ans angle tr\_up angle\_nn color\_scheme off\_axis\_dist p\_lo p\_lole poly\_lo poly\_lole on\_axis\_dist sf\_lo sf\_lole transmitter transmitter\_n

else

fprintf('Error')

end

fileID = fopen('script.mlx','w');

fprintf(fileID,'<!DOCTYPE FilterScript>\n<FilterScript>\n <filter name="Per Face Color Function">\n');

fprintf(fileID,' <Param name="r" type="RichString" value="%s" tooltip="function to generate Red component. Expected Range 0-255" description="func r = "/>\n',r);

fprintf(fileID,' <Param name="g" type="RichString" value="%s" tooltip="function to generate Green component. Expected Range 0-255" description="func g = "/>\n',g);

fprintf(fileID,' <Param name="b" type="RichString" value="%s" tooltip="function to generate Blue component. Expected Range 0-255" description="func b = "/>\n',b);

fprintf(fileID,' <Param name="a" type="RichString" value="255" tooltip="function to generate Alpha component. Expected Range 0-255" description="func alpha = "/>\n<Param name="onselected" type="RichBool" value="false" tooltip="if checked, only affects selected faces" description="only on selection"/>\n</filter>\n</FilterScript>');

fclose(fileID);

clear fileID ans