

Project documentation and further development of an electromagnetic-navigated ultrasound system

created by Thanh Nam Bach, CaMed summer term 2020

This document describes the course of the master project CaMed over the winter term 2019 and summer term 2020 of Thanh Nam Bach. Further documentation can be accessed from the MS Teams / Sharepoints of the CaMed project group. Code & Code documentation is provided via Github (<https://github.com/Zailux/TTRP>) and readthedocs (<https://ttrp.readthedocs.io/>).

Content

1. Overview.....	3
2. Project idea.....	4
3. Background.....	5
3.1. Electromagnetic tracking	5
3.2. Quaternion (simplified!)	6
4. Analysis.....	7
4.1 Predecessor project.....	7
4.2 New requirements.....	8
5. Concept	11
5.1 Aurora API Debugging Prototype	11
5.2 Uvis Application.....	11
5.3 Experiment Design.....	16
6. Implementation	18
6.1 Setup.....	18
6.2 Software architecture and components	19
6.3 Application GUI outline.....	20
7. Results	24
8. Reflection.....	29

9. Conclusion	31
Further applicable documents	32
References.....	32

1. Overview

Ultrasound imaging (or sonography) is a commonly used medical imaging technique, which is based on high frequency sound waves. Its application varies a lot and besides imaging has also therapeutic application. One of its major advantage is that it is a cheap and low-risk imaging technique for showing muscle or soft tissues. As a drawback of sonography is the challenging image interpretation. Absorption and reflection effects causes noise in the image and hence distorting the “correct” representation of the object. This problem makes reproducible ultrasound examinations and diagnoses challenging. In this project we develop a prototype for a navigated ultrasound system, which shall enable reproducible ultrasound examinations in consideration its specific use case. This particular work is based on the project by Peter Grupp [3] which chose an electromagnetic navigation approach, but increases the functionality and improves the navigation visualization.

This work starts with explaining the project idea which is followed by giving some theoretical background. Next the analysis and concept for the project is given and how the project scope developed throughout the project. Afterwards the implementation is explained and development challenges and decisions are explained. The code documentation is provided otherwise via a sphinx project. Next the conception and the results of the experiments are presented. The work is then critically reflected and concluded with an outlook.

2. Project idea

The original idea is based on Herr Dr. Eckhart Fröhlich, who expressed interest in reproducible ultrasound examinations. A translated quote is as followed:

“The goal of this project is to enable an operator-independent and reproducible ultrasound examination with an electromagnetic navigation system. By attaching EM sensors (EM = electromagnetic) to the ultrasound probe, it is possible to extract precise translational and orientation data within the magnetic field. Examined ultrasound objects and structures should be saved with the probe’s position. These structures should be examined in a repeated examination. The data will also be time-, location and operator-independent compared.”

The original quote can be found it [3].

In the next iteration of the project (meaning this project) it was intended to improve the usability and also introduce a new navigation concept. The new navigation concept should utilize a cone in order to navigate the user to the intended point and orientation. This will be further elaborated in Chapter 5. The high level plan of the project was to get to old project running and then further improve and implement the new requirements. The plan changed during the project progress due to problems with the old project and new necessary requirements. Throughout the project Tobias Baader also joined the project in which he focused on implementing the new navigation concept.

3. Background

3.1. Electromagnetic tracking

Electromagnetic tracking systems are nowadays used for supporting a variety of surgeries (Endoscopy, Cardiology, Interventional imaging ...). In most cases it is used for tracking (the tip of) surgical instruments. The instruments are usually visualized during a surgery in order to correctly position and navigate a needle for example. Such data is often used in combination with overlaying pre-operatively acquired MRI or CT images. A major advantage of electromagnetic tracking systems compared to other tracking systems are, that no line-of-sight between the field generator and the sensor / tracking point is necessary. Optical tracking systems have to deal e.g. with such occlusion.

In electromagnetic tracking systems, a field generator is used in order to cover an object within varying magnetic fields. With that such a known volume is created in which sensor can be tracked. Due to the varying magnetic fields the sensors are induced with a certain voltage, which can be interpreted for position and orientation. In this project we'll use a Table Top Field Generator of the NDI Aurora series.

In the left picture are two different field generators of the NDI Aurora series. In the right image it is shown, how such generator is used to create a known measurement volume (in blue). The circles are the varying magnetic field. If a sensor coil is attached to a surgical instrument and moved inside the volume, the tracking system can return the positional and orientation data.



Figure 1 - NDI Aurora systems

Source: <https://www.ndigital.com/medical/products/aurora/>

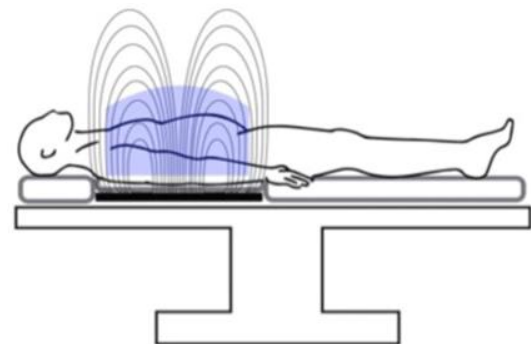


Figure 2 - Tabletop Magnetic field with human

Source: [7]

The sensor data for the NDI Aurora system which our project uses have six degrees of freedom (6DOF). The three translation values on the x, y and z-axes (in mm), and the three rotation values roll, pitch and yaw (in degree). The raw sensor data describes the orientation in quaternion, which will be explained in the section below. To be precise it is denoted as *unit quaternions*.

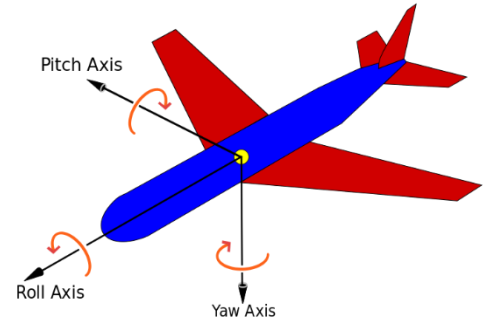


Figure 3 - Roll Pitch Yaw

source:

https://en.wikipedia.org/wiki/File:Yaw_Axis_Corrected.svg
– 10.8.20

3.2. Quaternion (simplified!)

Quaternions can be used for describing a **rotation around an arbitrary axis**. Advantages of quaternion are the avoidance of gimbal lock (problem with systems like Euler angles) and it is faster and more compact than matrices.

A quaternion q can generally be represented as followed:

$$q = a + bi + cj + dk$$

where a, b, c, d are real numbers and i, j, k represent complex number and are also called *quaternion units*. The quaternion can also be describing in two parts: In green the real part (or scalar) and in red the imaginary part (or vector part).

The conjugation of a quaternion is similar to the conjugation of complex numbers as is noted as followed:

$$\bar{q} = a - bi - cj - dk$$

The norm of a quaternion is calculated with the square root of the product of a quaternion. If the norm is one, such quaternion is denoted as **unit quaternion**.

$$\|q\| = \sqrt{q\bar{q}} = \sqrt{a^2 + b^2 + c^2 + d^2} = 1$$

Regarding rotations quaternion describe the two important parts:

- Rotation angle Θ – represented with the real part a
- Rotation axis (X, Y, Z) - represented with the imaginary part $bi - cj - dk$

For easier explanation we will apply this with the received values from the NDI Aurora sensor coils and denote this as following:

$$\bar{q} = a - bi - cj - dk = Q_0 + Q_x + Q_y + Q_z$$

The rotation angle Θ can be calculated as followed:

$$\Theta = 2\cos^{-1}(Q_0)$$

The rotation axis (X, Y, Z) will be represented as:

$$(Q_x, Q_y, Q_z)$$

For further reading please refer to <https://mathepedia.de/Quaternionen.html>.

4. Analysis

4.1 Predecessor project

The predecessor project implemented as an end-result an EM navigated ultrasound system. For that they used a technology stack: Python 2.7, tkinter as GUI library and pyserial for communicating with the NDI Aurora System. The system uses a frame grabber to access the image data from the ultrasound machine. The tracking data is acquired via serial communication with the Aurora system.

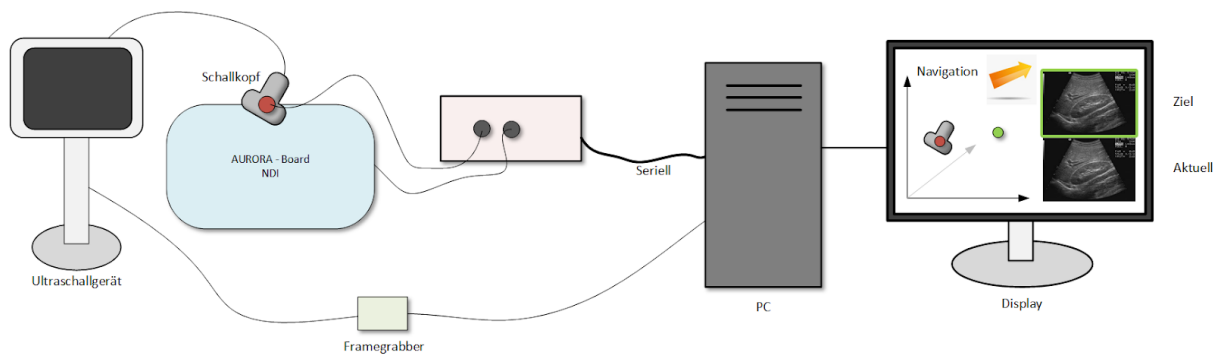


Figure 4 - System architecture - predecessor project

The first step for the analysis, was to read up the predecessor project code and documents. The first goal was to get the predecessor project running in a newer Python version (3.7) at the time and to understand the implementation.

The following improvements were identified

- UltraVisGui.py is the major module, which contains all three classes and is based on MVC architecture. It was labour intensive to comprehend the code.
- Sparse setup and installation documentation
 - Necessary libraries for running the project were not explicit defined
- The communication with the aurora system was hard coded and focused on short term functionality

After a month of trying to get the predecessor running, I decided to not try to get the old project running again and just implement a small GUI prototype and redevelop the application. Due to the inexperience with Python and tkinter the new goal was to implement a GUI prototype with which the Aurora API could be tested (regarding response duration etc.).

The system architecture (Figure 4) of the predecessor was carried over to this very project. The main reason for this was, that previous goal was to carry on with the predecessor and not restructure it from the base.

4.2 New requirements

As the project idea didn't change essentially we could reuse the previously defined requirements and refine it as follows. We separate in between the *application requirements*, which requirements were for the in chapter 0 defined and *project requirements*, which requirements are "internally" which goal is to enable a long-term value from this master project.

Project requirements

One major goal which I set for myself was that the project should have a long-term value and should be easily given to another student for continued development. In order to achieve such maintainability, I defined the following requirements:

- This project shall implement a separate API for communicating with the NDI aurora system, which then can be easily installed and maintained but a community or future students.
- The project shall introduce an easily accessible and maintainable code documentation

Application requirements

Based on previous analysis new requirements were defined. In order to achieve such we created an requirements diagram based on the SysML approach by Weikiens [8].

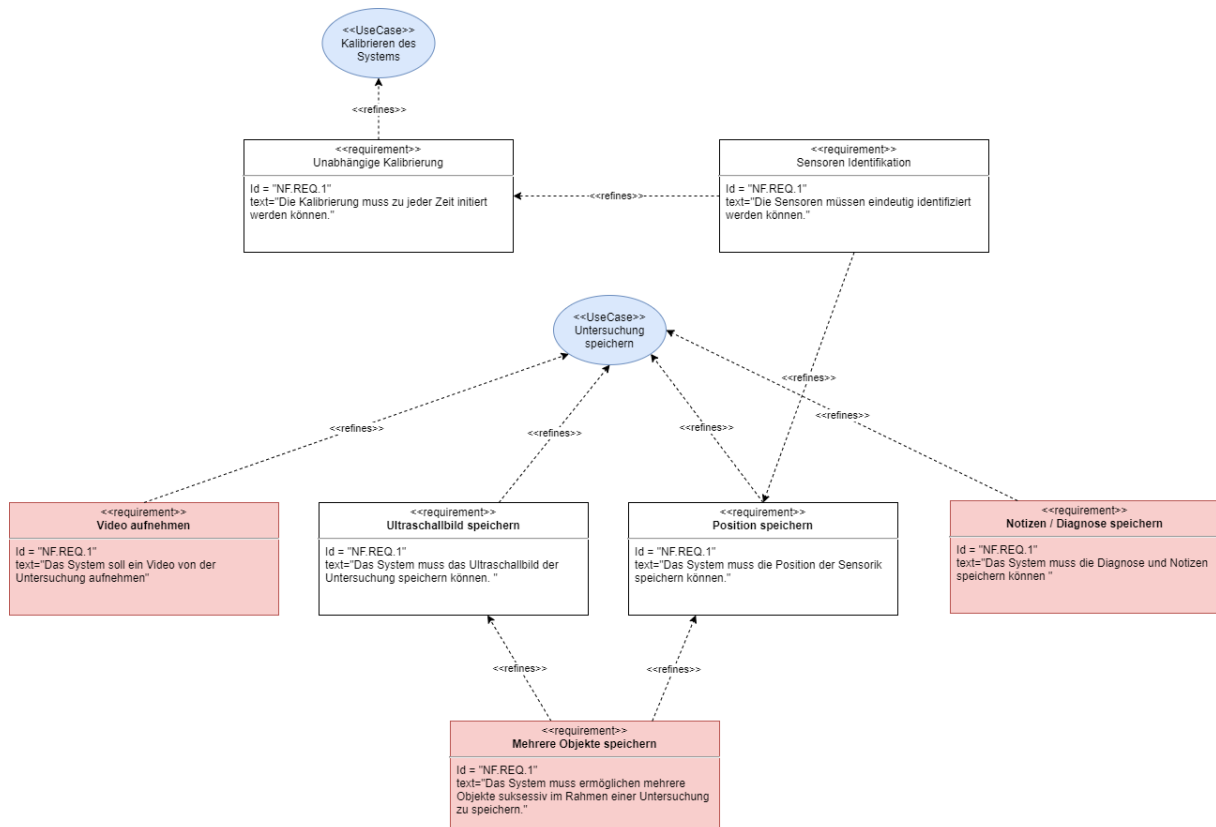


Figure 5- Requirement diagram - Calibrating system and save examination

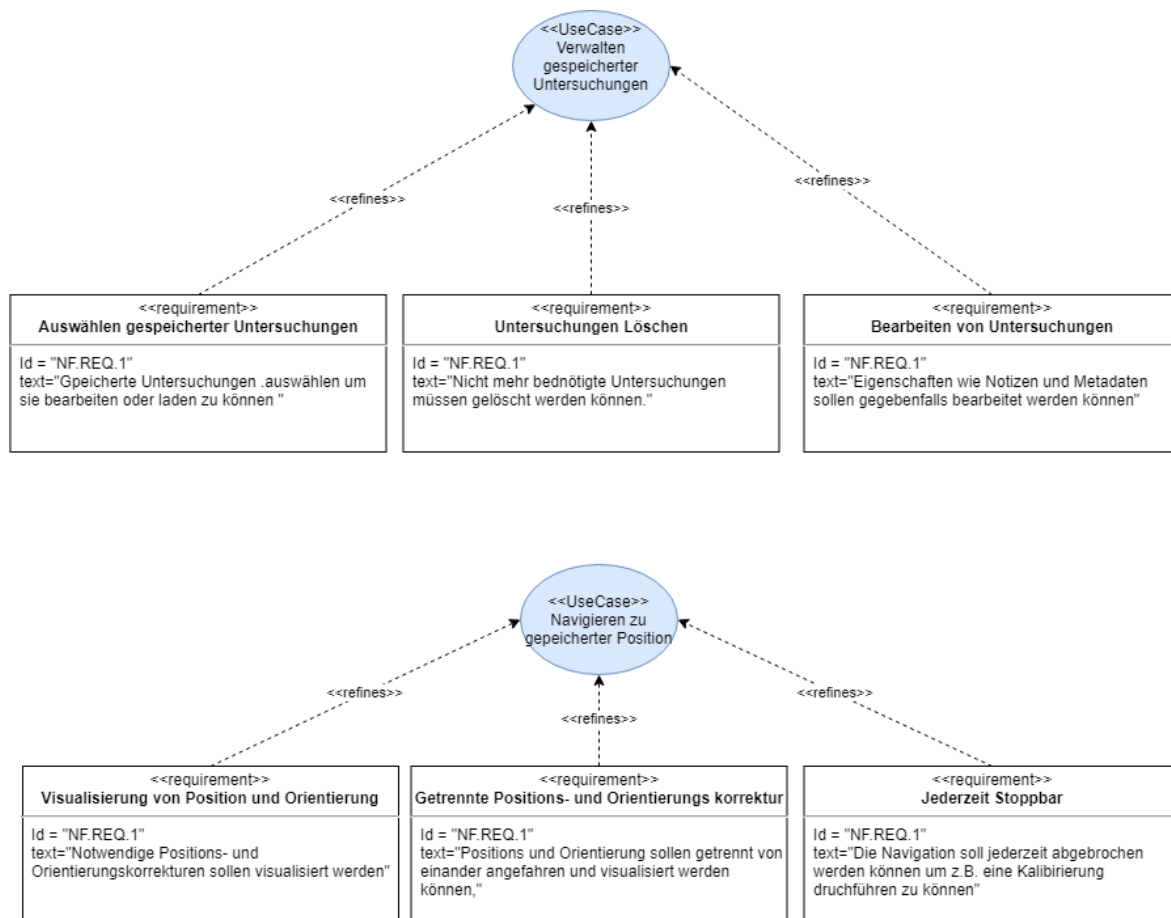


Figure 6 – Requirement diagram - Manage saved examination and navigate to saved position

At the beginning of the project we defined 4 use cases for the system:

- **Calibrate system**
Calibrate the navigation system and attach the sensors to the patient.
- **Save examination**
Save the examination data to the disk. This includes the meta data of the performed examination (operator, patient info etc.), the ultrasound image and the sensor data.
- **Manage saved examination**
CRUD operation on such saved examinations.
- **Navigate to saved position**
Load a saved position and guide the operator to the saved position via navigation.

The detailed requirements can be read from Figure 5 and Figure 6

The primary goal of this system is to enable reproducible sonography examination with a navigation system. Such system could be used for teaching students to find structures in body dummy with an ultrasound probe. In this project we will assume that the system will be used in a sonography seminar or a similar situation and we will provide the necessary functionality for that. In the further development of the application we added new necessary functionality to project, like comparing several examination accuracy data to each other. This was especially relevant for the experiments we

conducted in order to evaluate our system and also correctly adjust the tolerance setting of our system. We also included simple plots for a useful visualization of our data.

Non-functional requirements

As for non-functional requirements we considered performance and maintainability in our implementation the most.

Performance

We want a real-time application which navigates the user previously saved positions.

Maintainability

We want an application, which can be easily maintained and extended in a long-term perspective.

5. Concept

In this chapter the concepts for the several prototypes are presented. Besides the application itself, the experiment concepts are also introduced.

5.1 Aurora API Debugging Prototype

One of the first goals was to develop an Aurora API in Python which could be further maintained or used by other projects. In order to conduct some tests regarding single commands and or procedures I implemented a GUI in tkinter. This also had the second purpose of getting to know the programming language as well the library.

The commands based on the official NDI Aurora API Guide v4 [6] and recommendation from the Aurora User Guide [5]. In such the different possibilities of communicating with the Aurora System are explained as well which responses are to be expected. I tested the different command regarding the processing time, in order to set correct blocking time. This served the purpose of avoiding concurrent requests to the system.

This prototype didn't need any particular concept, due to its nature of using it to try out the libraries and possibilities of the given libraries.

5.2 Uvis Application

The hardware architecture will not be further described because we adapted it from the predecessor project and explained it in chapter 4.

Software Architecture

Based on the analysis described in the previous chapter a design decision was made to split the MVC Architecture into q were defined, where the MVC Architecture (Model, view, controller) was represented, as well as the self-contained Aurora module.

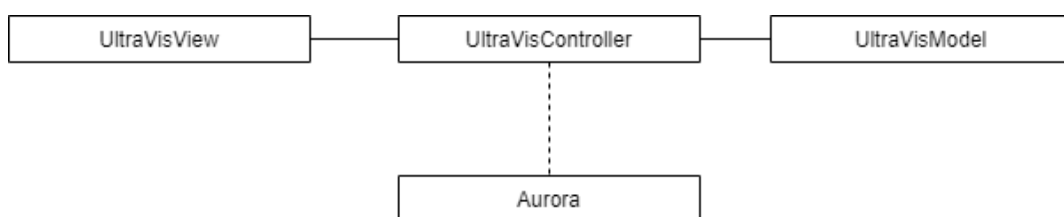


Figure 7 - Uvis Module Architecture MVC - Initial concept

Data Model

Due to necessity of persistent saving and loading data, a data model was introduced. The data model itself can be saved in a relational database, but for easier setup and installation it was decided to simply write in csv files. In Figure 8 the entity relationship diagram for the application is presented.

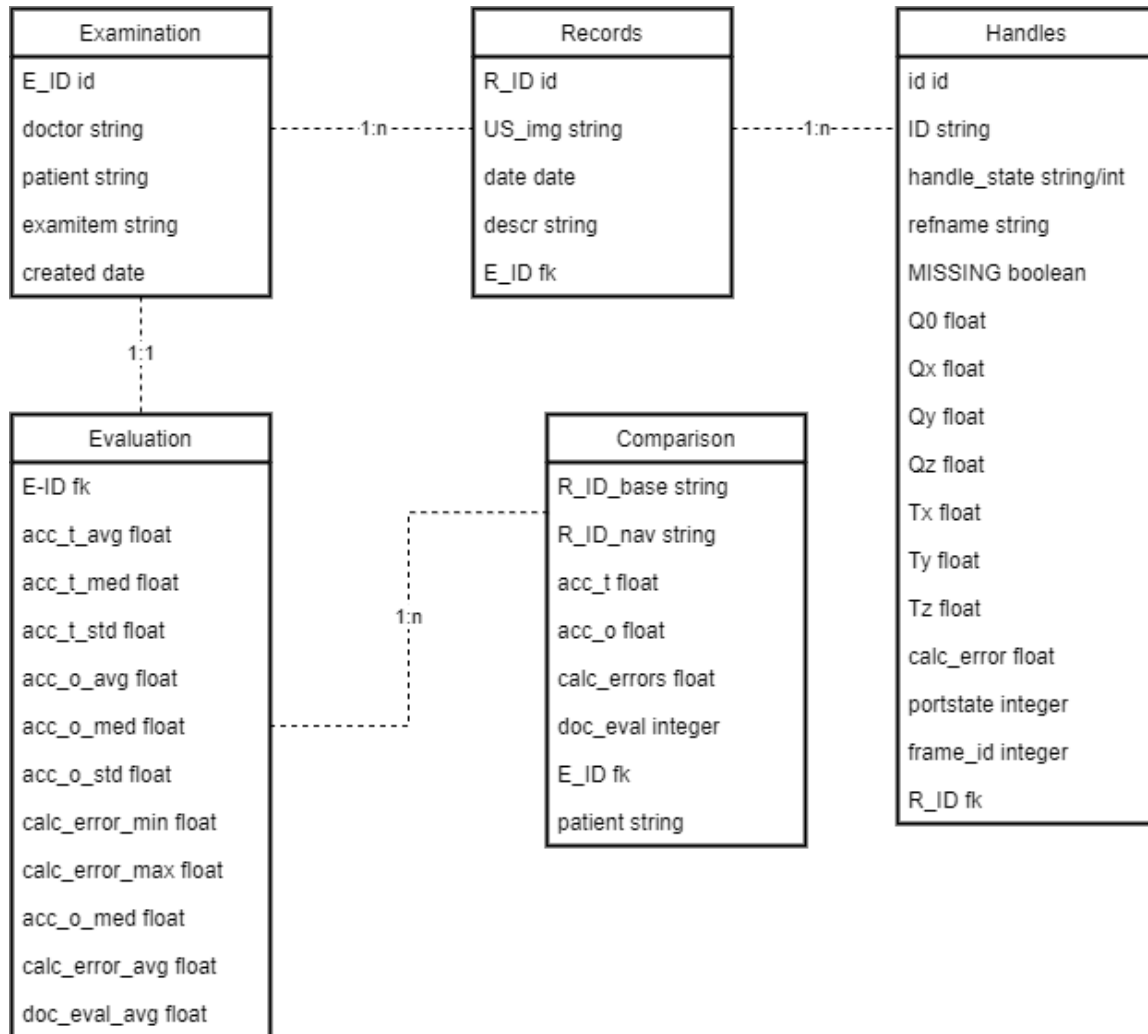


Figure 8 - Uvis App ERD

The *examination* entity is the base entity for the Uvis App. It contains meta data about the examination, like doctor and patient information.

The *record* entity, represent a performed record during an examination. It contains the recorded image and the reference to the recorded sensor data.

The *handle* entity represents the handle or a single sensor from the aurora system. It contains all data that the sensor held at a given time.

The *evaluation* entity holds the evaluation data of a performed examination. It describes for example the translational and orientation accuracy of the examination.

The *comparison* entity holds evaluation metrics of two compared records. The comparison shows the difference between two records (in general a goal record and the measured navigated record).

Paper Prototype

After having defined the based software architecture and the data model, we created a paper prototype for the describing the front end as well as the workflow.

We have 3 paths the user can take in our application

New examination

This path is used, to create a new examination for a patient.

The user first types in Examination meta data about the examination (like patient data). Afterwards the system is calibrated and the handles are attached to the patient. After the calibration the system is ready and the patient can be examined. Meaning records can be made with saving the different ultrasound images.

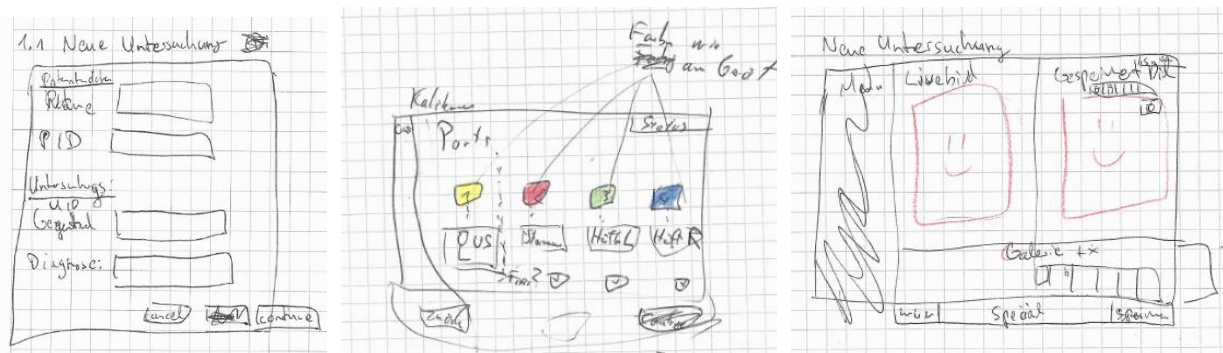


Figure 9 - New Examination Paper Mocks | Examination meta data (left) and Calibrate System (middle) and examine patient (right)

Load Examination

The user chooses via a dropdown the last available examination and loads the examination into the application. In such examination, the user can navigate to all available records. Being led to a recorded point, the user then chooses whether he wants to save a new record with the found structure, which hopefully reproduced the old ultrasound image.

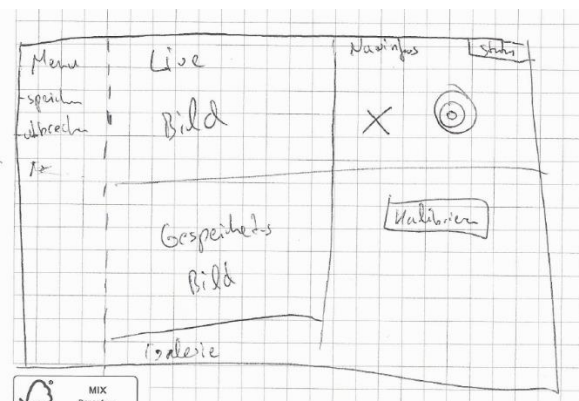


Figure 10 - Paper Mock - Load Navigation

Evaluate Examination

The user loads an examination, which already used the navigation functionality. The app displays evaluation criteria and shows for example how precise the navigation was. This functionality was added in the later stage of the project and therefore a paper prototype was not created.

Navigation Concept

As described, the predecessor project the usability of the navigation was not optimal. Therefore, professor Burgert suggested a cone-like navigation approach to support the user. The navigation concept is shown in the following figures.

Figure 11 explains the navigation concept and how it will be visualized in our application.

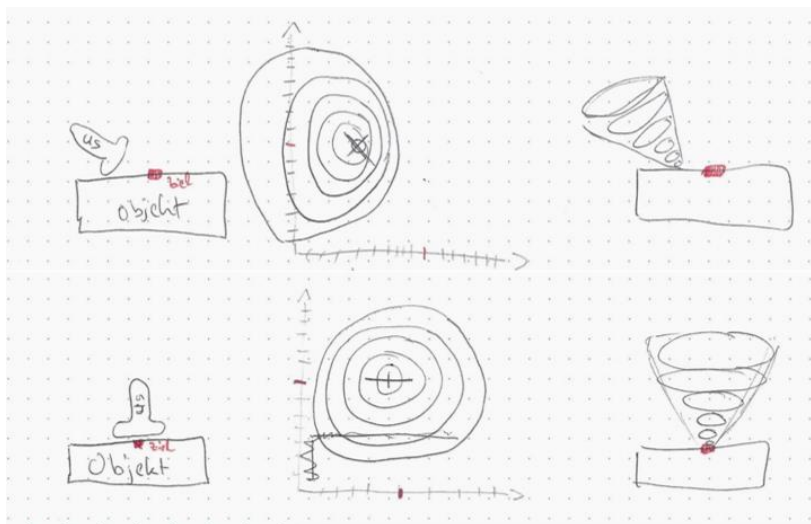


Figure 11 - Navigation concept

First column shows the realworld representation.

Second and third column shows the visualization in 2D and 3D.

First row shows an initial misplaced ultrasound probe and how it is visualized. All three rotation axis are misplaced and position is incorrect. Second row displays the ideal position when the navigated record is reached.

The position and rotation axis roll are described via the uneven X. The pitch and yaw rotation axis can be translated as tilt of the cone or the circles.

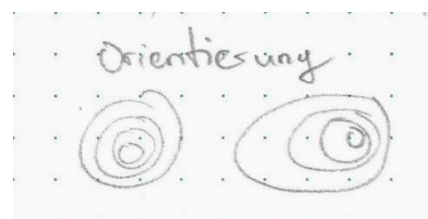
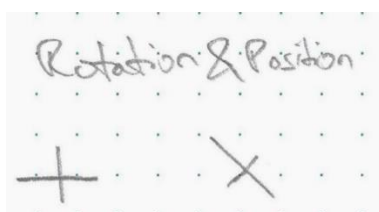


Figure 12 -Visualization techniques splitted up | Rotation and Position (left) and Orientation (right)

Human reference system concept

This figure describes how the reference system is calculated. In each examination the examined object (the patient) will be differently placed and orientated at the given time. In order to reproduce the examination, it is important save the relation between the ultrasound probe and the human.

For saving the human position appropriately, Dr Fröhlich recommended us to use reference points on the human body, which won't be moving after a period of time. These reference points are the hip bones and the sternum. In our system we will save the information of 4 points / handles. The ultrasound probe and the 3 sensor coils attached to the human body (left, right hip and sternum). With the three sensor information which were attached to the patient, we can create a reference system of the patient, which should be in the same proportion to each other. We can then create a plane, which will we can use as reference coordinate system of the patient. The relative position of the ultrasound probe compared to the reference system can then be reproduced, meaning a navigation to such point is possible.

This implementation of the concept was contributed by T. Baader and documented by the author.

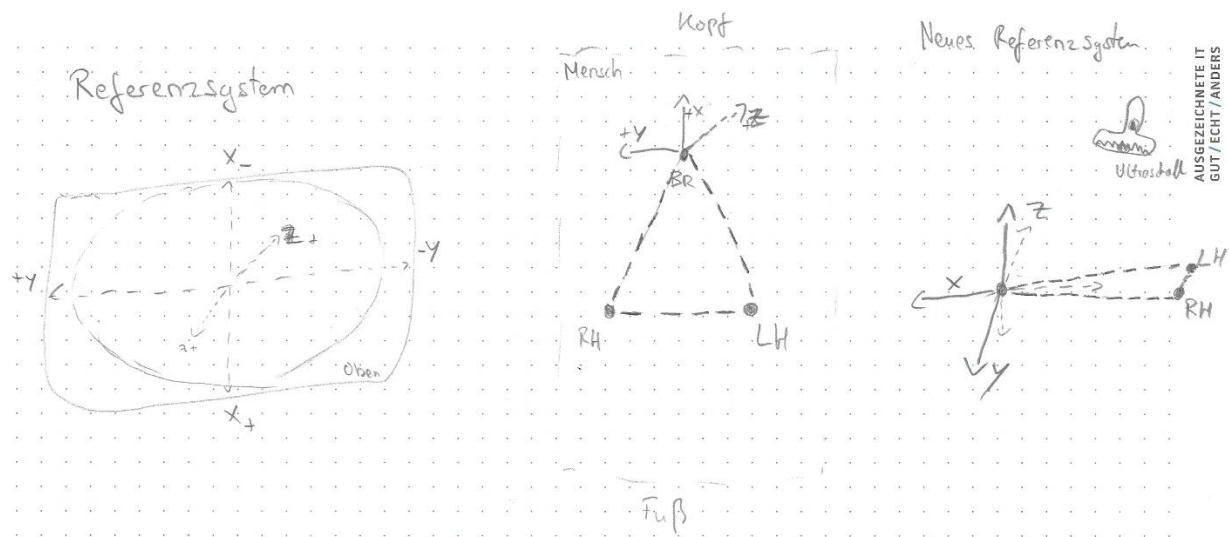


Figure 13 - Human Reference system | Left the original coordinate system of the NDI Aurora Board. In the middle and right are the calculated reference system displayed. It uses three points left hip and right hip bone (LH, RH) and the sternum (german Brustbein, BR). With these three points a new plane can be calculated and the reference coordinate system of the human can be calculated.

5.3 Experiment Design

As stated before the main goal of the project is to enable reproducible sonography examinations. Therefore, we designed and conducted experiments with the following superordinate goals:

- F-1** – How well can an operator reproduce a sonography examination with a common ultrasound system.
- F-2** – How well can an operator reproduce a sonography examination with our navigation system?

Before we could answer these questions we had to limit the given challenges for the sonography. We also had to conduct pre experiments in order to find our limits of our systems and identifying the ground truth. In this project we were only able to conduct the pre experiments but also created the ground work for further research.

In the following I will describe key points of the experiment design. I will not describe all the relevant measured variables or how the experiments were conducted. Please refer to the further applicable document [2] for details to the experiment design.

The major challenge which we tried to limit was the human body, anatomy and movement which are relevant factors during an examination. First of we intend to attach the sensor coils on the human skin of the reference points on the given reference points (hip bones and sternum). Finding the points via touching (*palpation*) will cause a certain human error. This registration error (finding the human anatomy and correct attaching the sensors to body), will have a negative impact on the precision of the system. Besides that, we also don't consider the growth younger patients for example, whose anatomy will change a lot depending on the age and time. Another challenge which we simplified in the current experiment design is the inner body movement. During sonography examinations the structure inside the body is fluid / is moving. This has an impact on the reproducibility of such examinations. Also the movement or rather the body posture changes for different examination is not considered at the moment. The different postures can distort the position and orientation of our reference points / system, meaning the relative position of the ultrasound probe is moving.

In order to solve all the challenges, we simplified the setup as followed:

- We fixed our reference system on a wooden frame
- We didn't move our reference system (patient) during recording
- Translative movement in between two records are only of x, y axis and not z-axis



Figure 14 - wooden frame with string phantom (Fadenphantom) | the points A,B,C are the reference points according to the anatomy of young human male adult (hip bones and sternum)

The wooden frame and the string phantom would enable us to conduct experiments in an optimal setting, in which the described challenges wouldn't influence the examination. The string phantom would be particularly useful to create good ground truth / reproduced images which are very accurate, due to its hard to find but visually distinct reproducible image (see Figure 15).

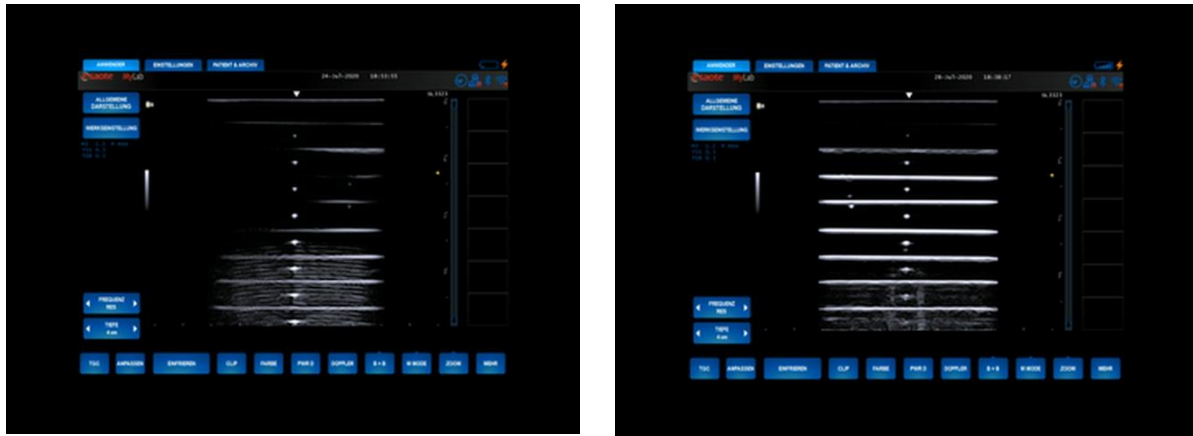


Figure 15 - String phantom images | left unprecise image and right perfectly aligned image

With this general set-up we designed the following pre-experiments:

- G_0 Grenzwertmessung – What is the technical accuracy of the system?
- G_1 Grenzwertmessung – What is human error in an optimal setting?
- G_2 Registrierungsfehler – What is the registration error in an optimal setting?

After conducting these experiments, we would be able to reasonably set the tolerance of our system, and get an indication whether that system error would be sufficient for a real world application or not.

6. Implementation

In this chapter the implementation is presented. First, the final setup of the system is explained with the several components which were used and how they fit together. Next the final software architecture and the modules are described and which key features / concepts were implemented. As explained in the motivation of the project an easily comprehensible and detailed documentation is intended. For an deeper insight to the code please refer to code documentation [1] by N. Bach. The documentation is also most updated available on <https://ttrp.readthedocs.io/>. And finally a walkthrough to the final implementation is given.

6.1 Setup

This section gives an overview how the final setup of project looks like. The installation chapter of the code documentation will give a thorough step-by-step guide to get the software and the setup running [1] or <https://ttrp.readthedocs.io/>.

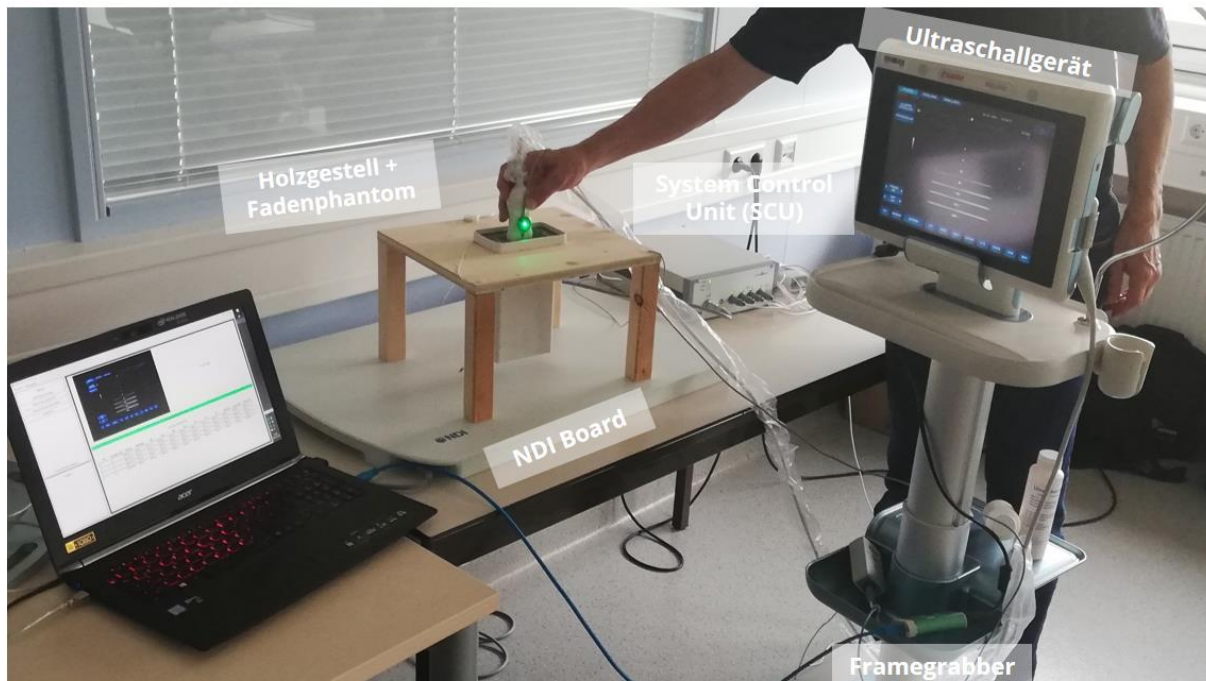


Figure 16 - Project Setup

Figure 16 - Project Setup describes final setup of the project. First of all, the NDI Tabletop Board in combination with the system control unit (SCU) is seen. The SCU connects the board with the sensors (and theoretically the SIU, which are left out due to simplicity). The board generates the magnetic fields and initializes the 4 sensor coils. Via an USB-B cable the SCU is connected with the notebook on which the Uvis Application runs. The 4 sensors are attached to the wooden frame and the Ultrasound probe. In order for the NDI Aurora system to appropriately work with our notebook, the NDI toolbox and additional drivers need to be installed. In our setup we use the ultrasound system from Esaote MyLabSat. The image from the ultrasound system is connected via a DVI cable to the frame grabber. The AV.io frame grabber from epiphan is then connected via USB to the notebook to which the frame grabber streams the video data from the ultrasound system. The frame grabber normally should

work as a plug&play system and doesn't require drivers. If any issues occur the installation chapter of the code documentation also contains the drivers for manual setup.

6.2 Software architecture and components

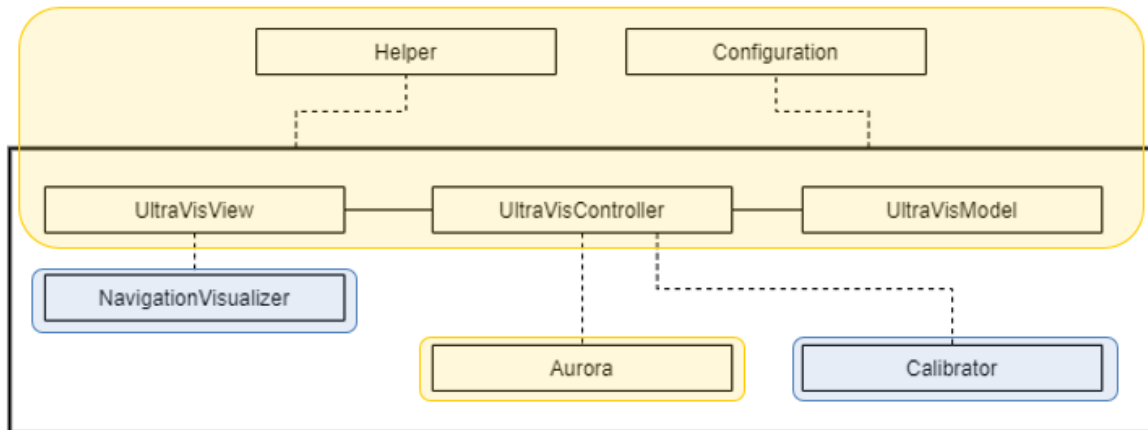


Figure 17 - Software architecture | the different modules and their relationships. The main responsibility of the modules is colored. Yellow for N. Bach and blue for T. Baader.

The uvis app is based on a MVC architecture. In the following a short description of each module is given. Details to the modules and how they work can be accessed in the code documentation [1].

The **uvis_view** module is responsible for building the view for the application. It primarily uses tkinter and matplotlib for visualizing the app.

The **uvis_controller** module controls the application and defines the processing flow. Its responsible for exchanging data with the Aurora API and intertwining the logic with the view and model modules. It also initializes the frame grabber and manages the threading.

The **uvis_model** module is responsible for managing the data model of the uvis application. It implements CRUD operations of the data and manages the data itself in csv files. Additionally, it contains functionality for calculating the evaluation of the examination data.

The **aurora** module is a python implementation of NDI Aurora System API Guide v4. It enables a serial communication with the NDI Aurora System. It also contains additionally functionality which enables an easier handling of the handles and returned information from the Aurora system.

The **helper** module contains all miscellaneous functionality which is used throughout all modules in the uvis application.

The **config** module contains the configurations for setting up the uvis application. Setting correct paths, Logging configurations, serial configs etc.

The **Calibrator** class build transformation matrices, which are used for creating the reference system for the uvis_app.

The **NavigationVisualizer** module creates the view for the navigation visualization.

Most relevant concepts / „features“:

- Threading + Queueing – parallel processing of logic (essentially I/O of NDI communication)
- Structured and reusable frame in combination with decorators
- Logging framework (important for reading logs in combination with several threads)
- Observer pattern
- Quaternion calculation for comparing one and another (getting distance etc.)

6.3 Application GUI outline

Before getting into the final GUI outline, I just show the two minor prototypes. On the left the Aurora debugging GUI is shown, for testing the API. On the right side the first GUI for recording examination is shown. Unfortunately, the 3D visualization with matplotlib had to be removed due to performance issues. The plotting was too processing intensive for the single core.

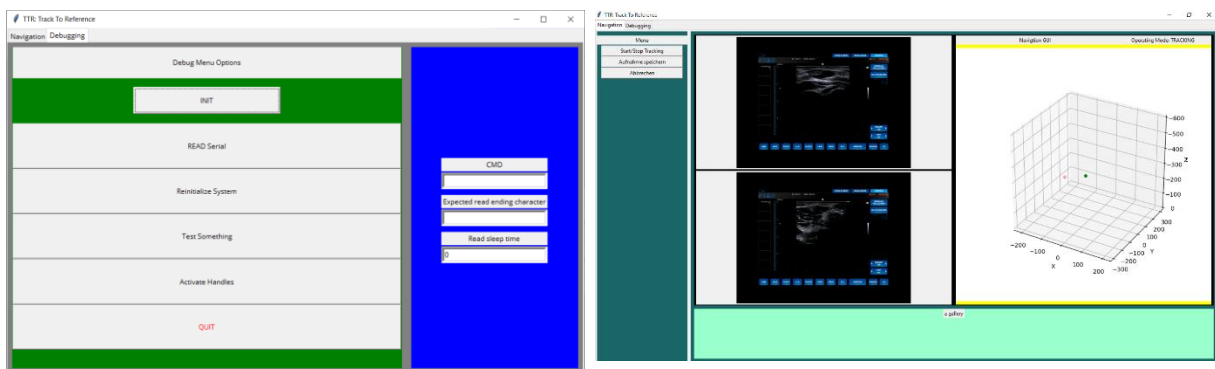


Figure 18 - Previous Prototypes

As explained in the concept chapter above the Uvis Application has three user flows.

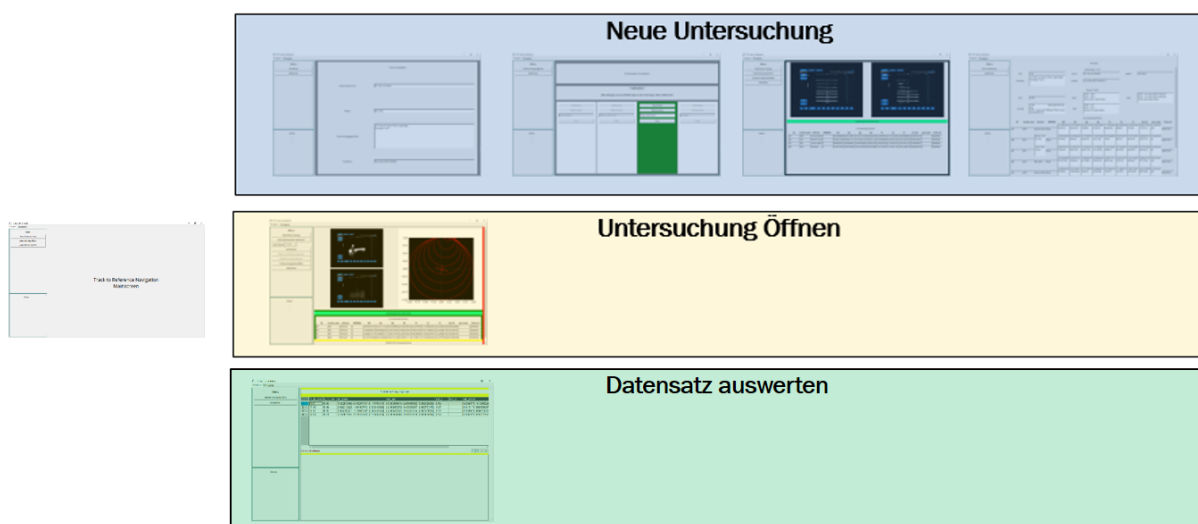
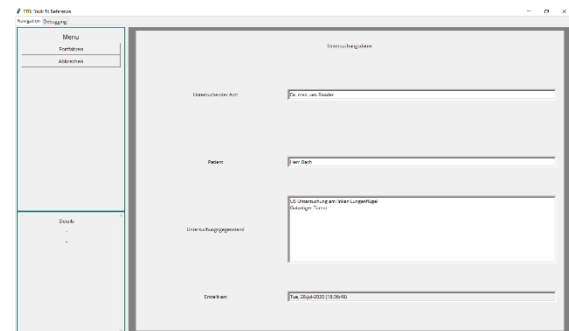
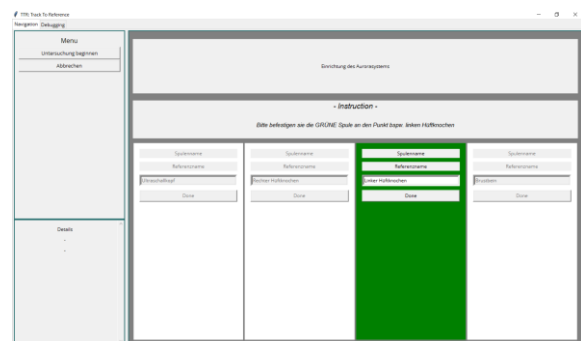


Figure 19 - Uvis App outline | Starting from the main menu three user paths are available. New examination, Open examination, Evaluate Dataset

New Examination

The user starts with simply typing in meta data about the examination (doctor, patient etc.). Next the user is guided to attach the sensors to patient. Ideally the color of the application is equivalent to sensor colors inside the application. In the background the application initializes and activates the 4 handles via the Aurora API. After the enabling the sensors the Aurora System is ready for tracking.

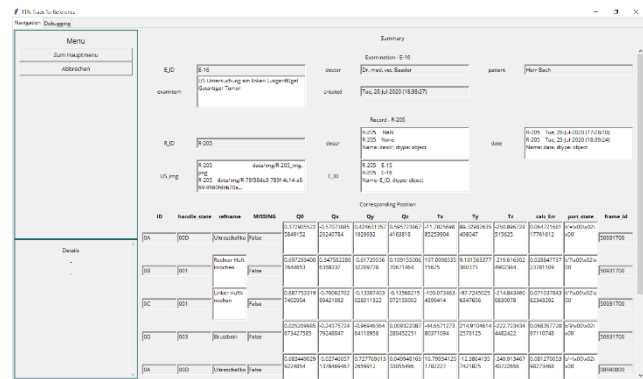



The application then starts the frame grabber in the background and shows the video stream in the app. The user can now start the tracking, which is displayed in tabular form below. A thread in background is started, which will frequently acquire the sensor data from the tracking system. The user can then save several positions (or structures) during the examination and finally can finish it.



ID	handle_state	refname	MISSING	Q0	Qx	Qy	Qz	Tx	Ty	Tz	calc_Err	port_state	frame_id
0A	000	Ultraschall	0	0.687488496	0.00178401	0.72571474	0.026355766	18.94907951	-12.9125394	-249.689682	0.079391136	=	F8690000
0B	001	Rechter Hüft	0	0.506017206	0.642911911	-0.29745125	0.492071181	145.4499966	144.5512396	-220.514022	0.036318503	?	F8690000
0C	001	Linker Hüft	0	0.662859976	-0.45608922	-0.54474639	0.236327052	149.0522308	-170.642120	-216.192123	0.150028020	?	F8690000
0D	003	Brustbein	0	0.019073605	0.362029965	-0.93162059	0.025573896	-128.708572	-15.9791736	-220.199874	0.065844222	9	F8690000

Finishing the examination, a summary screen is shown with all the information regarding the examination.



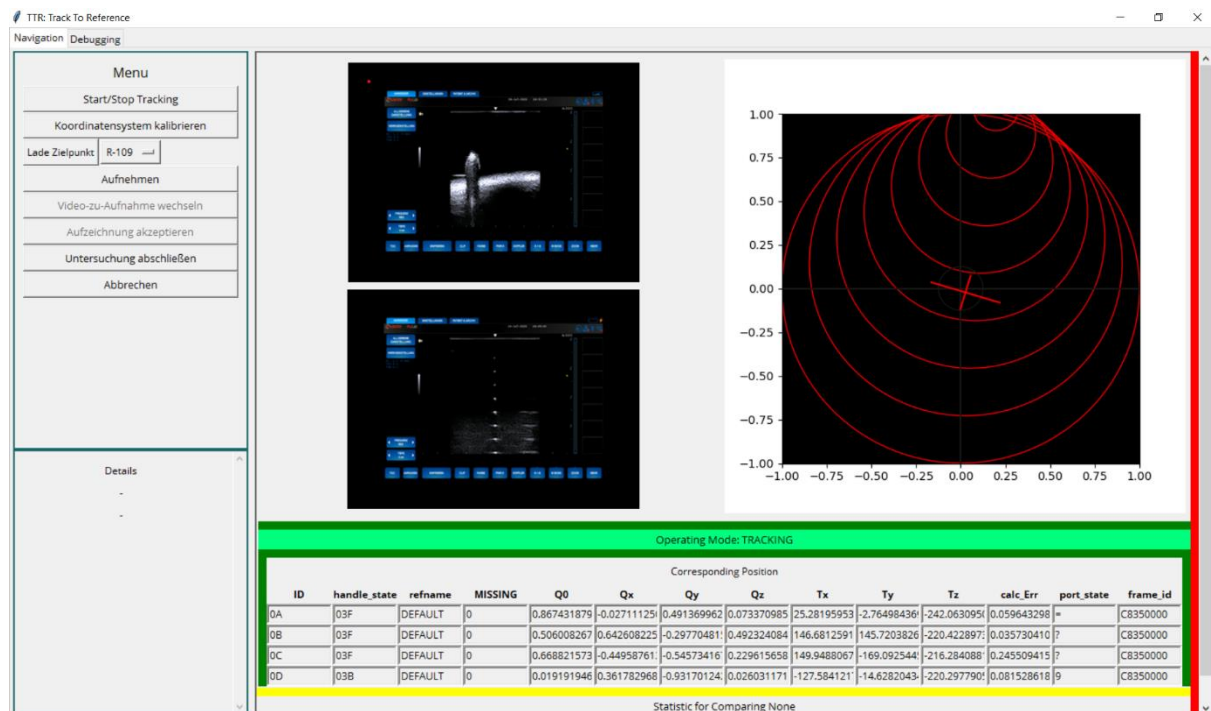
ID	handle_state	refname	MISSING	Q0	Qx	Qy	Qz	Tx	Ty	Tz	calc_err	port_state	frame_id
0A	03F	DEFAULT	0	0.867431879	-0.02711125	0.491369962	0.073370985	25.28195953	-2.76498436	-242.063095	0.059643296	8	C8350000
0B	03F	DEFAULT	0	0.506008267	0.642608225	-0.29770481	0.492324084	146.6812591	145.7203826	-220.422897	0.035730410	7	C8350000
0C	03F	DEFAULT	0	0.668821573	-0.44958761	-0.54573416	0.229615658	149.9488067	-169.092344	-216.284088	0.245509415	7	C8350000
0D	03B	DEFAULT	0	0.019191946	0.361782968	-0.93170124	0.026031171	-127.584121	-14.6282043	-220.297790	0.081528618	9	C8350000

Open Examination

The user chooses an examination via a dropdown to load examination in to application. The first record of the examination is then loaded into the application, which is ready for navigation. In the background the system resets the handles and initializes the handles for tracking.

The user then can start the tracking. Once the system starts tracking the user then calibrates the system in order to be able to navigate. To be precise the system sets the coordinate system based on the 3 reference points on the patient. After calibrating navigation visualizer can be correctly displayed and navigates the user to goal. The upper ultrasound image shows the current input Ultrasound image. The lower image shows the loaded ultrasound image from the record (the target).

The user then can navigate to the target position and save the position. The user then has to accept image in order to persist the record. Once accepted the base record from the examination is compared to newly created record, regarding e.g. accuracy. The statistics for each comparison are shown on the bottom.



Unfortunately, the navigation visualization is still bugged and doesn't correctly display the orientation yet.

Evaluate Examination

This path was only partly implemented due to prioritization. Only a table with the data is shown. But the base methods for comparing the data are at least implemented.

TTR: Track To Reference

Navigation Debugging

Menu

Secret Blowup Button

Abbrechen

Details

Statistic for Comparing None

	R_ID_bas	R_ID_nav	vec_base	vec_nav	acc_t	acc_o	calc_errors
1	R-31	R-38	[0.42207049 -0.76267757 -0.17375377]	[-0.00966675 -0.46566962 0.08232525]	0.58		[0.0840° 0.101856224
2	R-18	R-39	[-0.82213222 -1.06480743 0.43340026]	[-0.00997272 -0.46530267 0.08277347]	1.07		[0.1151° 0.099059097
3	R-17	R-40	[-0.85678331 -1.11942397 0.42653198]	[-0.00925868 -0.46534618 0.08222958]	1.12		[0.1519° 0.099673576
4	R-33	R-41	[0.41601967 -0.76227231 -0.17560793]	[-0.00938848 -0.46454244 0.08363438]	0.58		[0.1036° 0.099471367

4 rows x 9 columns

7. Results

In this chapter the results of project are summarized and explained. In order to measure the results, we need to see how well the requirements were satisfied.

Project requirements

- This project shall implement a separate API for communicating with the NDI aurora system, which then can be easily installed and maintained but a community or future students.
- The project shall introduce an easily accessible and maintainable code documentation

We created a reusable API for communicating with the NDI Aurora system in Python. The API does cover the most relevant commands for the tracking use case. Not all commands are covered as of this moment, but these commands are not relevant for most use cases. Additionally, we created an easily accessible and maintainable code documentation which is hosted public on readthedocs.io. If further development on the project is done, the documentation will semi-automatically build after each push on the master branch. Meaning we have overall focussed on creating a long-term solution and hence satisfied both requirements. This is also the major contribution to the non-functional requirement of *maintainability*.

Application requirements

We created a prototype for an electromagnetic navigated ultrasound system. The prototype satisfies the defined use case as followed:

- **Calibrate system - satisfied**
Calibrate the navigation system and attach the sensors to the patient.
- **Save examination - satisfied**
Save the examination data to the disk. This includes the meta data of the performed examination (operator, patient info etc.), the ultrasound image and the sensor data.
- **Manage saved examination - satisfied**
CRUD operation on such saved examinations.
- **Navigate to saved position – partially satisfied**
Load a saved position and guide the operator to the saved position via navigation.

Our application can satisfy 3 out of the 4 use cases. Regarding the navigation to a saved position we still have issues regarding the orientation visualization. The calculation between the currently measured orientation and the target orientation isn't working properly. At the end of project, we weren't able to get the calculations with the quaternions running. The translational navigation works fine as of this moment.

- **Evaluate examination – partially satisfied**

This is the latest feature we were working on and this use case could not be sufficiently satisfied. Most of the calculus is worked through. Meaning we can calculate the translational difference between two vectors and orientation difference between two quaternions or rather measurements. For the orientation the transfer to the reference system is at the moment missing. But once the transfer is done, the calculation of accuracy can be done (translational in mm and orientation in

degrees). We also created useful plots for visualization the data, but such is not appropriately included in the application yet.

Non-functional requirements

- **Maintainability**

As explained before we have satisfied this requirement with the sphinx code documentation and the self-contained developed Aurora API. Besides that, we also created a requirements.txt, which can be used by the future team for setting up the application. With virtualenv the correct dependencies and version will be included in the application.

- **Performance**

Performance was a major issue in our application. Our first problem was with matplotlib in combination with tkinter. Those libraries were not made for real-time plotting meaning, our visualization required too much time to be plotted, meaning we had to find an alternative. Next we assumed that constantly refreshing the frame grabber and the getting current tracking data was I/O bound. So we implemented threading which enabled us with parallel processing, but not true parallelism. The limit is the GIL (global interpreter lock), because we were essentially still running on only one core. We have tried to get multiprocessing running, but this caused too many issues with the tkinter and due to prioritization we stopped investing in that topic anymore. Besides tracking we managed to improve performance by reducing certain loading of GUI elements and also utilizing the BX command instead of the TX command in the Aurora API. The response from TX is longer than from the BX command but both contain the information (TX as string and BX in bytes).

In the end we have achieved a near-time visualization at the moment which is “acceptable” for the user but this means that we only partially satisfied the performance requirement.

Experiment

As described in the chapter 5.3 we have designed and thoroughly analyzed the experiment for our project. We have conducted two of three ground truth studies which results we describe as followed.

- G_0 Grenzwertmessung – What is the technical accuracy of the system?
- G_1 Grenzwertmessung – What is human error in an optimal setting?
- ~~G_2 Registrierungsfehler – What is the registration error in an optimal setting?~~

Before interpreting the results of our study, one should know the limitations of the aurora system (see figure below). For this study we used 6DOF sensor coils and we measured only the accuracy in consideration of ISO standard 5725-1 [4].

Table 4-2 Dome Volume (Tabletop Field Generator) - Position Errors

Position Errors	5DOF		6DOF	
	RMS (mm)	95% CI (mm)	RMS (mm)	95% CI (mm)
Position Accuracy	1.2	1.8	0.8	1.2
Position Precision	0.7	1.3	0.4	0.9
Position Trueness	1.0	1.3	0.7	0.9

Table 4-3 Dome Volume (Tabletop Field Generator) - Orientation Errors

Orientation Errors	5DOF		6DOF	
	RMS (°)	95% CI (°)	RMS (°)	95% CI (°)
Orientation Accuracy	0.5	0.7	0.7	0.8
Orientation Precision	0.2	0.3	0.3	0.5
Orientation Trueness	0.5	0.6	0.7	0.8

Figure 20 - Error measures of the NDI Aurora system

For our current ground truth errors, we now that the system itself has a position accuracy of 1.2 mm (CI 95%). All statements below are in consideration to this error for each single handle. Meaning this these values in Figure 20 show the maximal accuracy our system can reach due to the tracking limitations.

In our ground truth measures (G_0) we conducted an experiment for finding out the base accuracy of our system, without any influence. In G_1 we conducted experiments, in which we had a series of measures, once operated by Layman and once by an expert. A more detailed summary can be accessed in Table 1 - Ground truth experiments results.

	G_0 system precision	G_1 Human error	
Conditions	<ul style="list-style-type: none"> All sensors are fixed on the wooden frame For each measurement the wooden frame is moved in a translative manner 	<ul style="list-style-type: none"> The reference system (3 sensors) are fixed on the wooden frame A user tries to reproduce the optimal phantom image in each measurement For each measurement the wooden frame is moved in a translative manner Each measurement is roughly done after a minute 	
Number of measurements	61	44	59
Position accuracy	Fehler: Avg = 0,53 mm Med = 0,38 mm Std = 0,4 mm CI (95%) = 1.32 mm	Layman error: Avg = 2.39 mm Med = 1.9 mm Std = 1.5 mm CI (95%) = 4,32 mm (5.65 mm – G_0 error)	Expert error: Avg = 2.10 mm Med = 1.94 mm Std = 1.04 mm CI = 3.88 mm (4,7 mm – G_O error)

Table 1 - Ground truth experiments results

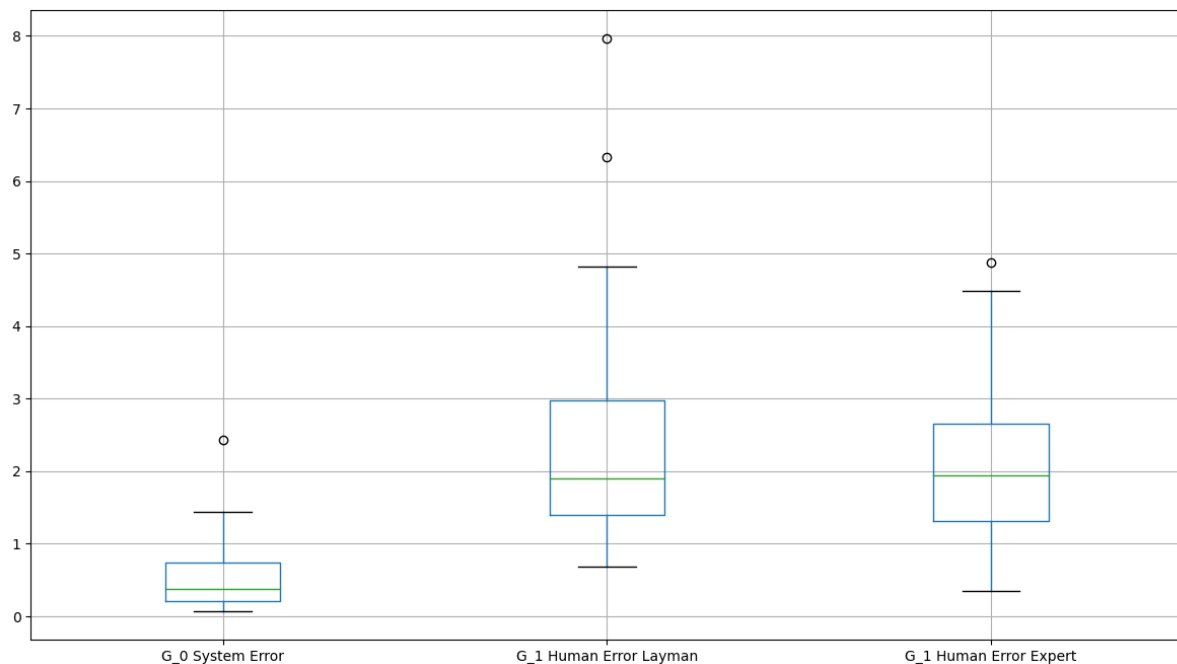


Figure 21 - Boxplots of baseline accuracy measurements

We also visualized the measurements in boxplots in order to better understand the distribution of the measurements. It is noticeable that the measurements made by the expert are much more “stable” because the median and average are much closer to the expert values than the layman measurements. The main insight from our pre-experiments is, that our system can have a positional tolerance of 4.7 mm in order to “perfectly” cover a saved structure, which can be accepted by human expert standards. But if we were to choose another tracking alternative which had a better precision than electromagnetic tracking, we can reduce the tolerance to 3.88 mm.

The third experiment was not conducted due to time limitations.

Further details regarding the experiments are available in the further applicable document [2].

8. Reflection

In this chapter I reflect on project and what the main challenges were throughout the project. In the beginning of the project it took me some time to get used to project and its documentation. The high level goal was to get the old project running and then implementing the new navigation concept. After spending a lot of time to “upgrade” the project and to actually get the setup for the system aurora system etc. I set new project goals. First of all, creating a general API for the Aurora system and secondly creating a long-term value code documentation, so future teams wouldn’t need to struggle as much as I did. The documentation had high level instructions for setting up the project, which weren’t that helpful because I had to research the things myself anyways. Another which needs to be considered when reading this is, that I myself never developed in Python before and also had to get used to programming language and its ecosystem.

The next issue was the chosen technology stack. I personally don’t think that tkinter is a good frontend library, because it had caused many issues throughout the project and it is as well not an industry standard. For example, the visualization capabilities of tkinter with matplotlib work for static applications or if the visualization is meant to be rendered once. But for applications which need real-time interactions (in 3D), it wasn’t working that well. Another issue is that tkinter doesn’t really support nowadays standards like responsive design appropriately and also simple GUI elements like scrollbars had to be used via a workaround. In hindsight it would have better if I researched other system architectures first and not “blindly” trusting the predecessor setup. Because in the end the predecessor project was mainly used as an orientation and the project itself is developed from scratch. This had the advantage of getting to know a lot about the Python itself and its possibilities. Overall I still learned a lot in this project.

The next topic I want to discuss is the Aurora API which was written. In order to get the project running as fast as possible I had to made some cuts to API and couldn’t not implement the API as whole. I prioritized functionality, over completeness meaning reasonable “features” like implementing the CRC16 was left out at the current state. The API as of this moment is good for using the Aurora system and is helpful as itself. But I think in order to get the most out of it, the next iterations should invest some time to improve the API in order to get to a professional level. Alternatively they should also look into the <https://github.com/PlusToolkit/ndicapi>, which is the C implementation for communicating with the aurora system. One reason, why I couldn’t use that project at the time was, that after a quick overview I read that the only NDI Polaris API was available in Python, not the Aurora system. Maybe that have changed by now.

Regarding performance, I’ve spent a lot of time of getting my head around the threading module and its possibilities. A challenge which I had was, how to appropriately use the capabilities especially considering concurrent access and locking. In the end I managed to introduced a Queue + threading concept into the application but I wasn’t able to get multiprocessing done. I achieved parallel processing on a single core, but not true parallelism by utilizing several cores. Reason for that is, that my initial advances with multiprocessing and tkinter, caused to create several tkinter frontend instances, which was a total faulty behaviour. It was comprehensible why it happened, but I didn’t invest time into finding a solution, because it seemed to me that it would take a lot of time and we were generally under time pressure to deliver functional software for our partners in University Clinic Tübingen.



Generally speaking, I tried to apply many useful concepts into the application, like reusable frames, threading, decorators, observer pattern etc. which took a lot of time, but they were hard to explain / justify for our “client”. It was definitely right to implement them, but as said again they were time consuming and also caused our project to be delayed. But I have to say all the delays were accepted and we had the support to continue our project the way we planned.

The next issue I want to address is the data handling in our project. Initially I thought by using csv for persisting and handling our data, we would overall gain speed in our development. Which was initially correct, but the further the project advanced I had to implement more methods for handling creating unique indexes and the like. This effort would be avoided if I used a common database. In hindsight it might have been better to just setup a database and to create scripts for installation.

The algorithmic most challenging topic in this project were by far the quaternions. We had to include our math professor in order to understand that topic. For me it was relevant for calculating the differences between to measurement, so I could evaluate the orientation accuracy of our system.

Concluding this chapter, the documentation of the project also costs a lot of time. Especially the sphinx documentation part. The setup of the semi-automatic documentation, in combination with the autodoc extension, cost me 2-3 days to get it properly working. And also learning the rst (reStructuredText) syntax and correctly using it with the docstrings was time consuming. I also documented all methods which were implemented. In hindsight it might have been better to selectively document the methods. Besides for Visual Studio Code there is a docstring generator(<https://marketplace.visualstudio.com/items?itemName=njpwerner.autodocstring>), which might work well with autodoc (<https://www.sphinx-doc.org/en/master/usage/extensions/autodoc.html>) and would have saved me a lot of time regarding creating the doc string. The VSCode extension can document in the sphinx format via configuration.

9. Conclusion

In this work we address the challenge of reproducible ultrasound examinations with an electromagnetic navigation approach. For that we first give theoretical background of electromagnetic tracking and as well as Quaternions for describing arbitrary rotations. Next we analyzed the predecessor project and defined new requirements for this project. We created 4 major artifacts: the self-contained API for the Aurora system, the electromagnetic navigated ultrasound system prototype, the experiment design for validating navigated ultrasound systems and a semi-automatic code documentation with sphinx. Afterwards we explain our concepts for our prototypes as well as our experiment design. Next we explain the application itself how the different components work together. Our application is able to fulfill most requirements well, but the cone navigation concept has still issues. We also implemented features for evaluating the measurement regarding their accuracy. Additionally, we conducted pre-experiments to identify the limits of our system under simplified optimal conditions. We found out that the positional accuracy of our system is 1.32mm and that the human error from experts is at 3.88mm for a CI of 95%.

After further development we will be able to compare the human examination with navigated measurements. With that we can answer our research question, whether our system will improve the reproducibility of ultrasound examinations. In the next chapter we reflect project and what major challenges we had to face throughout the project.

For the future we intend to finish the evaluation functionality, so orientation accuracy can be given. We also intent to implement the cone visualization with the quaternion data. With that we can conduct experiments for comparing the navigation system with manual examinations. Regarding our pre-experiments, we need to conduct the G_3 registration error experiment. Besides that, we could also conduct experiments regarding the human anatomy. At the moment we assumed that the reference system is fixed on the wooden plane with one particular "anatomy setup" (hip bones and sternum). By making the triangle bigger or smaller we could simulate different body shapes and test whether this will affect the system accuracy.

For the application we could try to improve the performance by introducing multiprocessing. Alternatively, we could also try to replace the frontend as whole with other frontend libraries like PyQt or create a web application. For the backend a database could be introduced in order to simplify the model module of the application.

Overall I personally think this work created a good foundation for further development on the project. Next teams should be able to faster understand the system and continue the work on the project or the experiments.

Further applicable documents

The following documents also count as part of this project documentation.

[A] – Thanh N. Bach. 2020. *Vorversuchsreihe zur navigierten Ultraschalluntersuchung*.

[B] – Thanh N. Bach. 2020. *TTRP Master Project Documentation*. – also available on <https://ttrp.readthedocs.io>

The sphinx code documentation may contain duplicates to this documentation.

References

- [1] Thanh N. Bach. 2020. *TTRP Master Project Documentation*. Reutlingen University.
- [2] Thanh N. Bach. 2020. *Vorversuchsreihe zur navigierten Ultraschalluntersuchung*.
- [3] Peter Grupp. 2018. *Projektdokumentation Track To Reference. Camed Master Projekt WS 2018*.
- [4] ISO. 1994. *ISO 5725-1:1994(en), Accuracy (trueness and precision) of measurement methods and results — Part 1: General principles and definitions (1994)*. Retrieved July 10, 2020 from <https://www.iso.org/obp/ui/>.
- [5] NDI Europe. 2013. *Aurora V2 User Guide - Revision 5*.
- [6] Northern Digital Inc. 2011. *Aurora Application Program Interface Guide - Revision 4*.
- [7] David Sindram, Kerri A. Simo, Ryan Z. Swan, Sharif Razzaque, David J. Niemeyer, Ramanathan M. Seshadri, Erin Hanna, Iain H. McKillop, David A. Iannitti, and John B. Martinie. 2015. Laparoscopic microwave ablation of human liver tumours using a novel three-dimensional magnetic guidance system. *HPB : the official journal of the International Hepato Pancreato Biliary Association* 17, 1, 87–93. DOI: <https://doi.org/10.1111/hpb.12315>.
- [8] Tim Weilkiens and Richard M. Soley. 2014. *Systems Engineering mit SysML/UML. Anforderungen, Analyse, Architektur* (3., überarb. und aktualisierte Aufl.). dpunkt.verl., Heidelberg.