COMPUTER AIDED PRE-PLANNING SOFTWARE IN FACIAL SURGERY

Markus Gall

Institute for Computer Graphics and Vision
Graz University of Technology, Austria

Head: Univ.-Prof. Dipl.-Ing. Dr.techn. Dieter Schmalstieg
Supervisor: Dr. Dr. Jan Egger
BioTechMed

External Medical Advisors: Dr. Dr. Knut Reinbacher
Dr. Dr. Jürgen Wallner
Department of Oral and Maxillofacial Surgery
Medical University of Graz

Technical Report
ICG–TR–xxx
Graz, July 12, 2016

contact: Markus Gall gall@student.tugraz.at

Abstract

In this contribution, a novel method for computer aided surgery planning of facial defects by using models of purchasable MedArtis Modus 2.0 miniplates is proposed. Implants of this kind are commonly used for treating defects in the facial area. By placing them perpendicular on the defect the miniplates are fixed on the healthy bone, bent with respect to the surface, to stabilize the defected area. Therefore, the developed software package is able to fit, a selection of the most common implant models, on the surgeon's desired position in a 3D computer model. To gain an appropriate result, this happens with respect to the local surface curvature capable of adjusting direction and position in any desired way. State of the art contributions show methods which are higher in consumption of time as well as in costs. Using Computed Tomography (CT) Scans, many of them generate stereolithic models serving as bending template for the implants or using a bending tool during the surgery where the implant has to be readjusted several times, leading to undesirable expenses in time. With the use of the proposed software, surgeons are able to pre-plan the out coming implant within just a few minutes in the computer-visualized model. Another advantage is that the resulting model can be stored in STL-file format which is the commonly used format for 3D printing. With this technology, surgeons are able to print the generated implant on time, or use it for generating a bending tool, both leading to an exactly bent miniplate, fitting perfectly on the desired position by spending minimal costs on time.

Keywords: *Computer Graphics, Facial Surgery, MeVisLab, Computed Tomography, Miniplates*

In Cooperation with

Medical University of Graz

Department of Dentistry and Maxillofacial Surgery
Division of Oral and Cranio-Maxillofacial Surgery
Auenbruggerplatz 12
A-8036 Graz

and within the BioTechMed-Graz cooperation and networking initiative

BioTechMed

Krenngasse 37/1
A-8010 Graz

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am …………………………… ……………………………………………..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

…………………………… ……………………………………………..
date (signature)

Acknowledgements

First, I would like to thank Univ.-Prof. Dipl.-Ing. Dr.techn. Dieter Schmalstieg for giving me the opportunity to be part of his research team and all the provided environment and support. More, I am deeply thankful for the supervising of Dr. Dr. Jan Egger who shared his trust and all his knowledge with me. I also appreciate the enthusiasm and the way he included me in his daily work and supporting me in his researches. With his guidance it was possible to achieve my comprehensive knowledge in scientific working and medical imaging. Additionally, I am thankful for the support and advice he offered me beyond all university obligations.

Also want to express my gratitude to Dr. Dr. Jürgen Wallner and Dr. Dr. Knut Reinbacher from the Medical University of Graz for their support in medical issues, their openness and the possibility they gave me to conduct my thesis with medical insight.

I also would like to express my gratitude to Dr. Dipl.-Ing. Clemens Arth, Dipl.-Ing. BSc. Philip Voglreiter, MSc. Pedro Boechat and all other members from the institute of computer graphics and vision who made my working experience to an unforgettable one and provided useful information also beyond all university obligations.

More, I am very grateful, that Jan Stanzel from MedArtis provided the dimensioning of the used miniplates.

Finally, I would like to thank my loved friends and family for their support. Especially I would like to thank my parents Konrad Gall and Margit Kitzweger-Gall who gave me the opportunity achieving a Master's degree and always had faith in me.

Contents

List of Figures

List of Tables

Listings

1 Introduction

Reconstructing facial deformations due to bone fractures or born deformations, is the daily business of a Surgeon's routine. More precisely, he has to fix all kinds of bone fractures which are results of applied outer forces, like happening due to an car accident, results of removed tumors or results from deformation treatment [36]. All those operations have in common, that the use of so called miniplates [24] is essential, independent of the defected location.

These miniplates in their raw form are straight titanium plates consisting, depending on the type, of at least two finishing ring sections at both ends, where screws for implant-bone fixation are drilled connected by a bridge section. The raw implant is available in countless different variations, consisting of further middle-ring sections, orthogonal bendings or circular ring section arrangements. Whereby in this work, I focused on the most frequently used ones, recommended by the surgeons of the MedUni (medizinische Universität) Graz, the MedArtis [7] Modus 2.0 series. With the purpose of stabilizing the defect, the implants are fixed perpendicular to the fracture on both fracture sites. The miniplates in their raw form are stiff and straight, why it is necessary to bend them along the fractured surface using a bending tool. Not only, that lots of time is lost during surgery, since the bending process happens on the opened patient, also the need of readjusting the bent sites to gain an acceptable and accurate implant, is further time consuming. Another Method is to generate a stereolithographic model out of the Computed Tomography (CT) scan data and pre-bending the implant on those. Needles to say, that this method takes even more time, especially in the preoperative phase, also leading to higher expenses in overall costs.

Therefore, this thesis proposes a novel method for computer aided planning of facial surgeries using the medical image processing platform MeVisLab [10] [27] and C++. By using this software, the surgeon has the possibility to plan the implant independent of the facial location without any expenses on a 3D computer model. Further, the planning time improves to only a few minutes being possible to readjust without any material costs. Finally, the implant model can be 3D printed, since it is stored in STL-file format, usable as bending tool or even final implant.

After CT scanning the patient, which is a mandatory task to each person suffering from facial bone fracture, the resulting file is provided in STL-format and already usable as input for the software tool. The user now, already sees

the loaded file represented in 3D. Next, the user chooses an implant type and selects any location on the surface of the facial model to place the implants center point. Using the center as a seed point, the baseline curvature is calculated by casting rays along the baseline and checking for surface intersection positions. Using the resulting curved baseline, the implant shape is generated by placing the ring sections, which are precomputed polygonal meshes generated in Autodesk's Inventor [1], at the locations along the curved baseline corresponding to the implant's dimensions. Each ring element of the implant is oriented to be aligned with the surface tangent plane at the chosen location, so that a perfect fit is guaranteed. Finally, the straight sections bridging the rings are generated by forming a template mesh with rectangular footprint. The final implant is then saved in the output directory, again in STL-file format, a common file format for any 3D printing device. Anyway, setting the initial point, the user first gains an implant showing in an arbitrary direction. However, by turning the mouse wheel, the direction of the implant can be changed until the user observes a satisfying outcome. Further, runtime is optimized by limiting computations to the region of interest (ROI) around the seed point.
The software was tested with real patient CT data provided by the Clinical Department of Oral and Maxillofacial Surgery of MedUni (medizinische Universität) Graz. The outcome shows very well aligned and with respect to the surface well bent miniplates, allowing the surgeons to get a good understanding of what a post-intervention will look like.

The thesis I propose, may help surgeons to minimize the expended time and costs by providing a flexible planning software. At this stage, the software is more a type of an visual planning tool using already pre-built, on the market available, implant types. Even though, the result is able to be 3D printed and used for further integration in the operative process, a future outlook includes an extension of the software to generate an even higher span of implant variations and may provide the possibility of generating customized and patient specific plates.

2 Medical Background

2.1 Facial Injuries

Facial injuries are results caused by various different influences where too much pressure is applied from the outside onto facial areas causing soft tissue damage or bone fractures. Independent if the applied force is the outcome of a sports accident like mountain biking, football, surfing, resulting from daily life accidents or even violence. However, one major cause in Austria and other regions where winter sports are frequently practiced is skiing and snowboarding [47]. These two sports account for most of the facial injuries overall, since wearing a helmet is not mandatory thus resulting in a high percentage of practizioners not euqipped with the head protection even though, in this type of sports one reaches high velocities not able to be absorbed by the skull in case of collision. According to Gassner et al. [46] in the article *Incidence of Oral and Maxillofacial Skiing Injuries due to Different Injury Mechanisms*, the average year of people transferred to the department of *oral and maxillofacial surgery* with facial injuries lies by 26 years with 50% between age 16 and 38. The group with the highest rate of injuries is the one of children between 7 and 12 years. Further, the statistics show a high variation between the groups of men and woman suffering from soft tissue damage or bone fractures in the facial area due to skiing accidents, stating that men account for 65.3% of the cases whereas females show a number of 34.7% injured patients. More, the types of injuries range from facial bone fractures to dentoalveolar traumas and soft tissue injuries. The statistics also show a bride range of different mechanisms including simple falls, collisions with others or objects, a struck by equipment, lift accidents and various more. For a better understanding of these data, Table 1 gives an detailed overview of these statistics.

Having a look at Table 2 which shows the number and percentage of the different bone fractures caused by skiing accidents, it is clearly observable that fractures in the orbital area are the most common ones where the zygoma (outer, right orbit bone) and the orbital floor (inside bottom orbital bone) show the highest percentage, followed by mandible fractures.

2.2 Treatment of Oral and Maxillo Facial Injuries

The following two sections are based on the chapter *Basics of Traumatology* from the book *Traumatologie des Mund-, Kiefer-, Gesichtsbereichs* [41] and discuss how an accident in the facial area is treated generally, starting from first aid interventions until the very end, followed by the second section describing how fractures are treated in detail.

2.2.1 Stages of Treatment

A patient suffering from facial damages due to an accident, independent if those harm soft tissues or cause bone fractures, has to undergo different

Variable	**579 Skiers/882 Incidents**
Age	
Mean	28.35
Standard Deviation	15.78
Min	2
Median	26
Max	81
Gender	
Male	378
Female	201
Injury Type	
Facial bone fracture	310
Soft tissue injuries	336
Dentoalveolar trauma	236
Mechanism of Injury	
Falls	263
Collision with others	135
Collision with object	46
Struck by equipment	70
Lift accidents	34
Others	31

Table 1: The table shows the statistics of the collected data in Innsbruck, Austria of oral and maxillofacial injuries caused by ski accidents.

stages of diagnosis and treatments executed by the surgeons. In the following lines those are listed along with a short description:

Pre-clinical evaluation - As the first stage in the diagnoses chain the most important vital parameters are measured to gain medical evaluation scores and check for poly trauma.

Trauma room I (minute 0-20) - The evaluated scores from the pre-clinical evaluation define the algorithm for this phase. Mandatory for this phase is the determination of the accident's circumstances and situation. Further, the information of the already taken interventions has to be transferred to specialists together with eventually occurred complications. The patient gets stabilized and monitoring includes measuring the blood pressure, pulsoximetry and ECG measure.

Basic diagnoses and therapy - If the patient is unstable, instant operations are following or the transfer to the intensive care unit. Otherwise the patient undergoes a full body CT scan and the first Trauma-Room-Conference is hold (minutes 20-35). Next, in a first stage of operations, live-sustaining measures have highest priority after transfer to the intensive care unit.

Trauma room II (minute 35-50) - In the case no emergency operations have to be done, diagnoses and therapeutic measures are completed. In a second Trauma-Room-Conference the specialists discuss about the transfer to the intensive care unit and further treatment.

Primal surgical intervention - This phase is executed after all emergency operations are taken care off and the patient is stable. In the first day the priority lies on fractures concerning heavily damaged soft tissue traumas, bleeding, complex fractures and fractures in combination with the eyes.

Secondary surgical intervention - The focus of the following days lies on fractures of facial bones including reconstructive osteosynthesis and corrections of soft tissue issues, eventually in cooperation with neurosurgeons.

2.2.2 General Fracture Treatment

Together with bone fractures, frequently soft tissues are damaged too, which have to be treated in the right manner as well. However, this thesis is focused on the reconstruction of the bony structures why the healing process of soft tissue injuries is not of big importance, thus not further discussed. Moreover, in the area of general fractures, it is differentiated between the methods of **closed reductions** and **opened reductions** described in the following sections.

Closed reductions describe the application of splints to regain the correct position of the jaw. Therefore, the patient has not to be cut open by applying these splints from the outside in three main types. One is the wired splint applied to the teeth, thus stabilizing the position with a wire around the teeth. Another one is the plastic as reduction element where the patient gets a plastic splint restoring the natural positioning. The third one is a construction where screws attached with hooks are drilled in the mandibular and maxilla to connect those hooks with a wire thus stabilizing the jaw. Further, also the application of extern fixation (fixateur externe), where for example a broken mandibular bone is fixed with a frame applied from the outside, is used in some cases. Moreover there exist other techniques used for closed reductions which are from minor importance and rather unconventional, thus not explained in detail.

Again, this thesis focuses on bone fractures in the facial area whereby those with open reduction are from the highest priority since the developed software supports surgeons in planning implants which are only applicable on the open cut patient. This method shows many advantages in the healing processes by increasing the quality level, lowering the time of treatment and also decreasing the patient's stress level. Osteosynthesis means the reduction and internal fixation of bony structures suffering from a fracture by using implants for fixation, such as miniplates. The most common methods of fixations are the wired osteosynthesis, the drill-wire-fixation and the screw and plate osteosynthesis. Further, one can distinguish between rigid and non rigid osteosynthesis.

Rigid ones describe a non moving fixation where both bones of the fracture sites are fixated in a way they can not move and have no or a small gap between each other. In this case, the bone recreates himself by gaining tissue to connect both sites. This process needs the application of rigid implants like metal plates with up to 2 mm thickness commonly referred to as miniplates. Using different systems, consisting of a variation of plates and screws, this method can transfer the naturally applied forces on the implants, thus allowing optimal condition for bone healing.

Non-rigid osteosynthesis describes the mobility of fracture affected parts by applying forces. Miniplates, applied to the fractured mandible is an example for this kind of osteosynthesis, allowing chewing movements. Also systems with wired osteosynthesis come to application in this case. Further, if the movement happens directly on the fracture site, which is also possible, the healing process is from secondary nature by first building cartilage which transforms itself to bony tissue afterwards.

Anyway, the applied system's purpose in each case is the handling of the load, naturally applied to the affected bones. As a result of fracture, these biological structures are capable of bearing this loads why the forces have to be redirected onto the implanted parts. There are big catalogs of implants in different forms and characteristics available. The following lines give a short introduction on the function of a selection:

Compression plates are preferably used in the mandible, frequently with 4-wells where those wells are modified in a way, that the screws move to the center of fracture when tightened, thus causing compression of the bones, Figure 1. As one can see in this figure, a second plate is applied which is

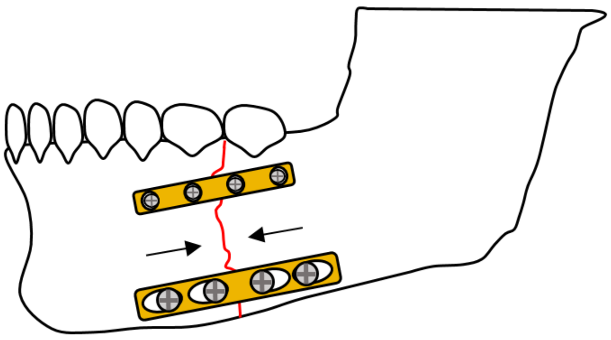


Figure 1: Standart fixation of a mandibular fracture where a compression plate (lower, golden rectangle), with the central screws (grey, crossed circles) glide to the fracture center due to the special form of the wells when drilling in the neutral screws (outer), thus enabling compression. A stabilizing plate (upper, golden rectangle), with normal wells and screws, against gaping in the dental area, is applied for support. The red line marks the fracture of the mandible bone. Adopted from [41].

not of compression type but of stabilization type, called **stabilizing plates**. This plate supports the near teeth area and is used to prevent gaping in this area. Further, to eliminate the possibility of lingual gaping the compression plate is bent in a manner that the plate-bone gap is approximately 1-2 mm in the center, Figure 2. By tighting the screws, a compression also on the lingual site is achieved. Moreover these plates can be reduced in most of the cases to so called **miniplates** also using monocortical screws with a diameter of 2 mm. However, in comparison to the mandible, where the implant positioning is more complicated due to the mobility of the jaw, in mid facial fractures miniplates give always sufficient hold. Using different thicknesses of the plates according to the local tissue, like thinner ones in soft tissue areas like the nose, in combination with different screw sizes, a broad band on different systems has evolved.

Fixed angle plates, like shown in Figure 3, are systems where the screw

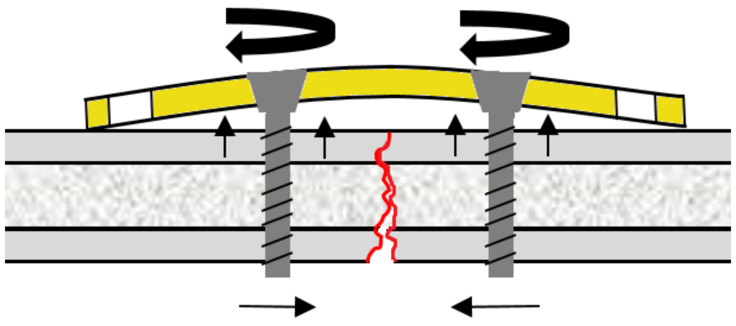


Figure 2: Applying of compression plate (golden) to prevent lingual gaping of a fractured bone (light grey, three layered structure). A bone fracture (red area) with a gap on the lingual side (lower side) of the bone, is compressed by bending the plate in a manner, so that the bone-plate distance has a value of 1-2mm in the center, thus two screws (dark grey) apply compressional forces of the lingual gaping side. Adopted from [41].

is fixated with the implant through threads or other mechanisms, thus implicating a few advantages. First, the plate shows a longer lasting fixation since conventional screws can get lose due to resorption at the contact points. Further, especially by using thicker plates, bone material gets displaced by tightening the screws which requires exact bending which is very hard to achieve but not necessary when using fixed angle plates since the displacement does not take place in this case. More, the stabilization of so called debris zones, which are areas where the fracture holds many small pieces of bone structures, is higher with this system type. However, this systems are not state of the art at the moment, since the theoretical advantages are still beaten by the lower price of the conventional miniplates. **Complications**
The rate of complications is stated with 2.5 to 3.5%. Failure in plate fixation does not only lead to aesthetic imperfections but also can harm the functionality and healing process which further causes even bigger damage. More, the risk of infections is not to underestimate and is especially increased in the case of

- immunosupression;
- after radiation therapy;
- under cytostatic therapy;
- by osteoporosis;

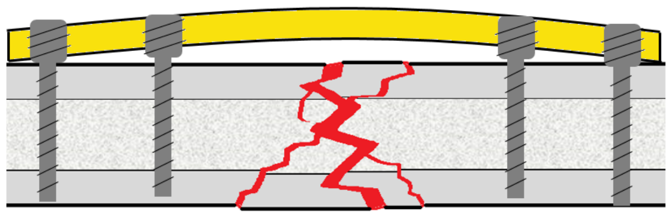


Figure 3: This figure shows the treatment of a bone fracture causing a debris zone, where the bone broke in lots of small pieces (red). Also the use of a fixed angle plate (gold) is shown which is tightened with screws (dark grey) having threads at the head section for implant screw fixation, thus allowing better fixation. Adopted from [41].

- bisphosphonatal therapy;
- or diabetes mellitus.

Further complications are also possible due to nerve or dental root damage.

2.3 Special Traumatology

In this section, special and often occurring cases of facial fractures are shown, since the proposed software is tailored for the treatment planning of these fracture types. The entire selection of cases and their treatment as described in the following sections are based on the work of Rasse's chapter *Spezielle Traumatologie* [42].

2.3.1 Osteosynthesis of LeFort-I Fracture

The typical fracture line LeFort-I, described by the equally named scientist, runs above the tooth root extending from the piriform aperture to the roots of the maxillary tuberosity and the fossa pterygopalatina in a mirrored manner on both sites. Four miniplates are fixated, each perpendicular on the nose and cheeckbone pillars forming the osteosyntehsis. Optimal placement of the miniplates is observable in Figure 4.

2.3.2 Osteosynthesis of LeFort-II Fracture

The fracture runs from the Suture nasofrontalis on Suture frontomaxillary by the Lacrimal bone or dorsal of the orbital floor and thus the maxilla, from

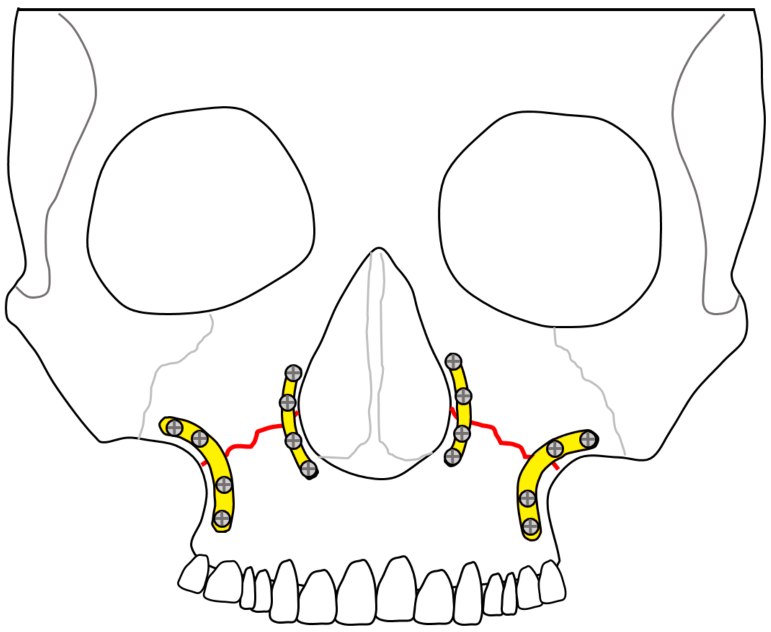


Figure 4: Osteosynthesis of a skull suffering from LeFort-I fracture (red). The fracture extends from the piriform aperture to the roots of the maxillary tuberosity and the fossa pterygopalatina in a mirrored manner on both sites where four miniplates (yellow) are fixated, each perpendicular on the nose and cheeckbone pillars. Adopted from [42].

here the infraorbital rim and on the facial antral wall and Crista zygomaticoalveolaris in the plane of LeFort-I fracture. Further extending around the Tuber maxilla in the Processus pterygoid, around the perpendicular plate of the palatine bone and the medial antral wall and reaches via the Ethmoid ascending to the medial orbital nose. From there, the fracture extends through the Septum nasi to posteroinferior and ends at the posterior edge vomer. The osteosynthesis are placed at the Crista zygomaticoalveolaris and the infraorbital rim, optionally together with a nasofrontal located miniplate. The standard application of osteosynthesis of the LeFort-I fracture is possible to be observed in Figure 5.

2.3.3 Osteosynthesis of LeFort-III Fracture

In the case of this fracture type, the facial skeleton is blasted out of the cranium. From the suture nasofrontalis on, the fracture runs on the medial orbital wall and orbital floor to the fissure inferior orbital. The zygomatic bones remain together with the blown off maxilla, so that the fracture of

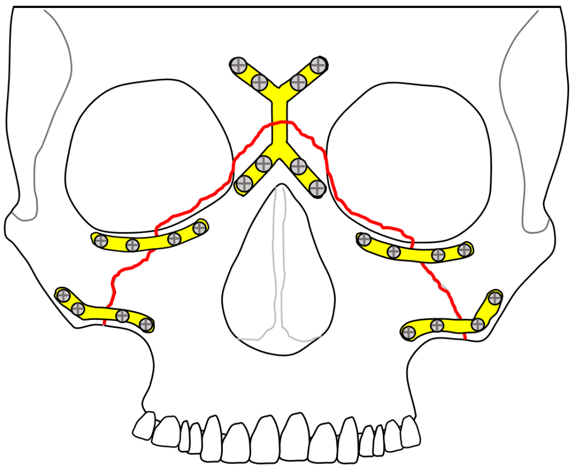


Figure 5: Osteosynthesis of a skull suffering from LeFort-II fracture (red). The fracture runs from the Suture nasofrontalis on Suture frontomaxillary by the Lacrimal bone or dorsal of the orbital floor and thus the maxilla, from here the infraorbital rim and on the facial antral wall and Crista zygomaticoalveolaris in the plane of LeFort-I fracture. It moves around the Tuber maxilla in the Processus pterygoid , around it by the perpendicular plate of the palatine bone and the medial antral wall and reaches via the Ethmoid ascending to the medial orbital nose. From there, the fracture extends through the Septum nasi to posteroinferior and ends at the posterior edge vomer. The osteosynthesis (yellow) are placed at the Crista zygomaticoalveolaris (lower pair) and the infraorbital rim (upper pair), optionally together with a nasofrontal located miniplate (middle, x-shaped). Adopted from [42].

the inferior orbital fissure to the lateral orbital wall to the Suture zygomaticofrontal runs superiorly and the zygomatic arches are fractured. From the suture nasofrontalis it reaches the inside by the Ethmoid and the perpendicular plate of the palatine bone, the pterygopalatine fossa. The Processus pterygoidei may be aborted too. The nasal septum is equally involved as in the LeFort-II fracture. The cribriform plate can as a result of Ethmoidal fractures be involved in the types LeFort-II and III. Osteosynthesis in this type of fracture is more challenging than in the others, since the face skull on the Sutures zygomaticofrontales and nasofrontales has to be fixated again on the neurocranium. Eyebrow cuts and glasses cuts are a possibility to perform osteosynthesis to these regions. However, osteosynthesis in this case results in many different variations of placement, since frequently, other fractures are present as well. Anyway, in Figure 6 a commonly used osteosynthesis is shown.

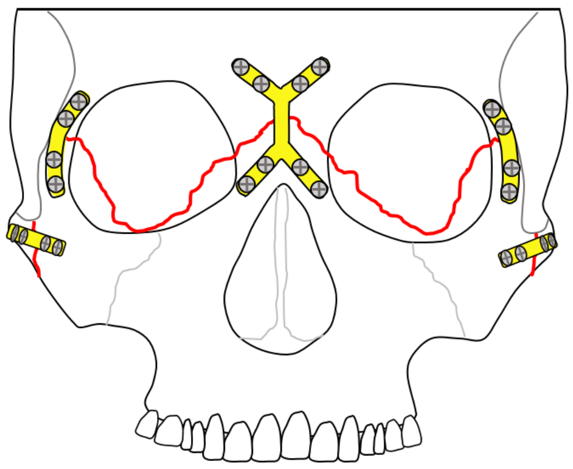


Figure 6: Osteosynthesis of a skull suffering from LeFort-III fracture (red). From the suture nasofrontalis on, the fracture runs on the medial orbital wall and orbital floor to the fissure inferior orbital. The zygomatic bones remain together with the blown off maxilla, so that the fracture of the inferior orbital fissure to the lateral orbital wall to the suture zygomaticofrontal runs superiorly and the zygomatic arches are fractured. From the suture nasofrontalis it reaches the inside by the ethmoid and the perpendicular plate of the palatine bone, the pterygopalatine fossa. The processus pterygoidei may be aborted too. The nasal septum is equally involved as in the LeFort-II fracture. Adopted from [42].

2.3.4 Osteosynthesis of Mandibular Fractures

By application of too much force to the mandibular bone, fractures occur mostly on the same spots, where the bone shows it's weakest structure. Thus, this section discusses the most common mandibular traumas together with their osteosynthetic treatment.

Mandibular media fracture is a fracture of the frontal, middle section of the mandibular bone. Generally, the apllication of two miniplates, one subapical and one caudal is sufficient. More difficult cases, where more compression is used, need a caudas fixed compression plate and a stabilizing subapical miniplate, where also lingual compression should be applied. Figure 7 shows a standard treatment with two miniplates. **Fracture in edentulous atrophic mandible** are fractures in the mandible bone where no teeth are locally surrounded, almost always appearing in a pair-wise, symmetrical to the median, fracture. Commonly used are load bearing plates for osteosyn-

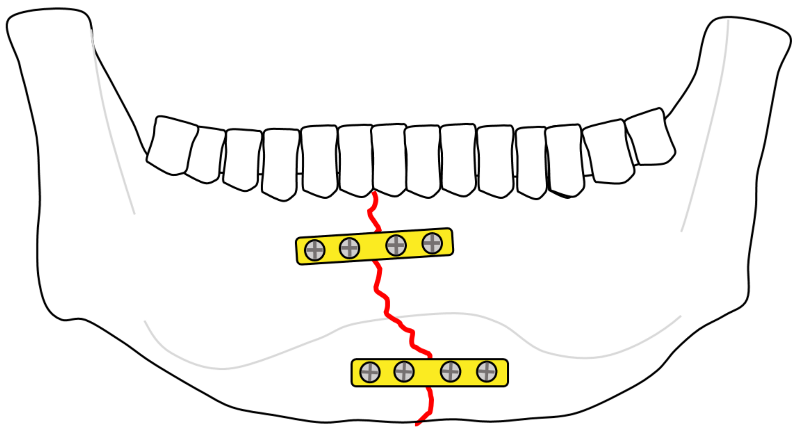


Figure 7: Osteosynthesis of a mandibular media fracture (red). In the standard treatment, the application of two miniplates (gold), one subapical and one caudal, is sufficient. Adopted from [42].

thesis which are fixed sufficiently in the side sections thus enabling optimal stabilization. Figure 8 shows such a case of a fracture. However, in bone transplantation more stable plates are used. Alternatively, if the patient is not able to undergo a bone spending surgery, prosthesis of plastic or other fabrics are applied. **Mandibular angle fracture** is a one sided fracture of

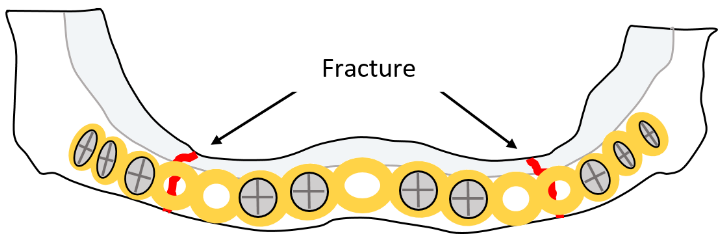


Figure 8: Osteosynthesis of a edentulous atrophic mandibular fracture (red). In the standard treatment, the application of one load bearing plate (gold) is used. Adopted from [42]

the mandible bone in the angle section. The treatment of mandibular angle fractures includes in all variations two implant plates, where one is aligned to the Linea obliqua and the other on the caudal edge. One possibility and the most preferred one is the usage of two miniplates. However, in some cases for example when the wisdom tooth is removed, the better option is to use a stronger plate. To avoid gaping the use of compression plates is a good choice. Figure 9 shows a standard treatment using two miniplates for fixation.

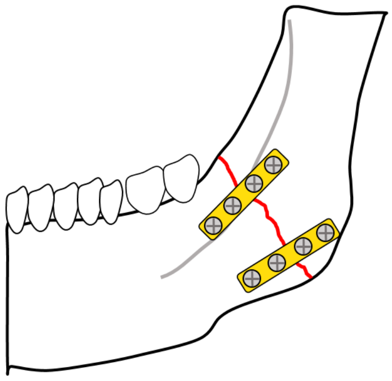


Figure 9: Osteosynthesis of a mandibular angle fracture (red). Using two plates (gold), one aligned to the Linea obliqua (light grey) and the other on the caudal edge. Depending on the stability and the need of compression miniplates and, or a combination of thicker compression plates may be the right choice in this case. Adopted from [42].

3 Technical Backgrounds

3.1 Mathematical Background

3.1.1 Ray-Surface Intersection

One important task in this thesis is the position allocation of ray-triangle intersections. Using this operation in a cascade manner, the baseline, representing the implant's centerline [23], of the surface curvature is determined. Therefore, Möller and Trumbore [37] proposed an ray-triangle intersection method, which is fast and low in storage costs, explained in the following lines.

Holding the triangle of interest, defined by its vertices $V1, V2$ and $V3$, an Ray $R(t)$ is defined by it's origin O and normalized direction D:

$$R(t) = O + tD \tag{1}$$

where t is the distance from the ray origin O, in direction D to the intersecting coordinates (u, v) of the triangle. Further, any point $T(u, v)$ on the triangle is defined by

$$T(u, v) = (1 - u - v) * V_0 + u * V_1 + v * V_2 \tag{2}$$

The barycentric coordinates (u, v) must fulfil the following criteria:

- $u \geq 0$
- $v \geq 0$
- $u + v \leq 1$

Furthermore, the intersection of the ray $R(t)$ and the triangle $T(u, v)$ is given by the equating $R(t) = T(u, v)$:

$$O + tD = (1 - u - v) * V_0 + U * V_1 + u * V_2 \tag{3}$$

leading to

$$\begin{bmatrix} -D, & V_1 - V_0, & V_2 - V_0 \end{bmatrix} \begin{bmatrix} t \\ u \\ v \end{bmatrix} = O - V_0 \tag{4}$$

By solving the linear system, the coordinates (u, v) and the distance t to the intersection point can be computed. Expressed in a geometrical manner, this calculation can be seen as a translation of the triangle to the origin followed by a transformation to unit size where the ray direction is aligned with the third axis. Figure 10 visualizes this relation. Applying Cramer's rule of matrices [18], and rewriting $C_1 = V_1 - V_0, C_2 = V_2 - V_0$ and $T = O - V_0$ the solution of equation 4 is:

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{|-D, C_1, C_2|} \begin{bmatrix} | \quad T, & C_1, & C_2| \\ |-D, & T, & C_2| \\ |-D, & C_1, & T\ | \end{bmatrix} \tag{5}$$

This equation is rewritten by using the *scalar triple product* (box product)

$$|A, B, C| = -(A \times C) * B = -(C \times B) * A \tag{6}$$

to obtain

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{(D \times C_2) * C_1} \begin{bmatrix} (T \times C_1) * C_2 \\ (D \times C_2) * T \\ (T \times C_1) * D \end{bmatrix} = \frac{1}{P \times C_1} \begin{bmatrix} Q * C_2 \\ P * T \\ Q * D \end{bmatrix} \tag{7}$$

where Q and P factors are used to speed up computational time.

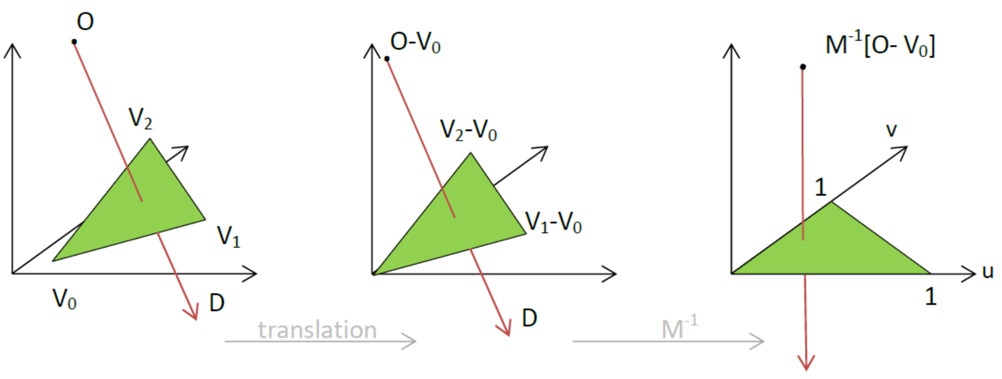


Figure 10: Calculation of the ray-triangle intersection. Starting with a translation to the origin and transformation to a unit triangle where the ray aligns to the third axis and $M_{1\times3}$ equals the first matrix (row-vector) in equation 4. Adopted from [37]

3.1.2 Quaternion Rotation

Quaternion rotation is commonly used in Computer Graphics [28] showing some important advantages compared to the better known Eulerian Rotation. Quaternions themselves are a number system extending the range of complex numbers where some of the mathematical laws, known from the real numbers, are not applicable. Anyway, through the introduction of quaternions, an elegant description of the Eucledean Space, especially concerning rotation, is possible.

Quaternions

They consist of real numbers and add three new ones describes with **i**, **j** and **k** as units in the complex-imaginary space. Thus building a 4-dimensional number system described with one real part and one imaginary part, where the real part is formed through a real component and the imaginary part is formed of three components, also named vector part. Further, each quaternion is possible to be written in the form

$$x_o + x_1 i + x_2 j + x_3 k$$

with real numbers $\mathbf{x}$. Therefore, the elements (**1,i,j,k**) form the standard basis of the quaternions over $\mathbb{R}$. Moreover, the numbers are connected with each other by the rule of Hamilton to:

$$i^2 = j^2 = k^2 = ijk = -1$$

Moreover, the scalar multiplication $\mathbb{R}$ x $\mathbb{H} \to \mathbb{H}$ allows to extend from the basis to all quaternions. Thus, the multiplication is associative, fulfills both distributive laws and forms the quaternions to a ring, but is not commutative which means for two quaternions x and y that the products of xy and yx are not the same [31].

Rotation
Further, any rotation in the 3-dimensional space can be described according do Euler's rotation theorem[TITAT Euler]: therefore any rotation about any fixed point of a coordinate system or rigid body can be written as rotation around a fixed axis (the rotation axis which runs through the fixed point) and an rotation angle θ. By definition, the rotation axis is represented by a unit vector $\mathbf{u}$. As a result, any rotation can be described as a combination of the vector $\mathbf{u}$ and a scalar θ in the 3-dimensional space. By introducing quaternions which encode this axis-angle representation in four numbers in a simple way, they give a position vector to a point, relative to the origin, which corresponds to the applied rotation.

With **i,j,k** representing the Cartesian coordinate system, a vector (a_x, a_y, a_z) can be written as

$$a_x\mathbf{i} + a_y\mathbf{j} + a_z\mathbf{k}$$

Any rotation around an axis defined by the unit vector

$$\mathbf{u} = (u_x, u_y, u_z) = u_x\mathbf{i} + u_y\mathbf{j} + u_z\mathbf{k} \tag{8}$$

is able to be represented by quaternion notation extending Euler's formula

$$\mathbf{q} = e^{\frac{\theta}{2}(u_x\mathbf{i}+u_y\mathbf{j}+u_z\mathbf{k})} = \cos\frac{\theta}{2} + (u_x\mathbf{i} + u_y\mathbf{j} + u_z\mathbf{k})\sin\frac{\theta}{2} \tag{9}$$

In addition, it can be shown that the rotation is able to be used with any three dimensional vector

$$\mathbf{p} = (p_x, p_y, p_z) = p_x\mathbf{i} + p_y\mathbf{j} + p_z\mathbf{k} \tag{10}$$

By evaluating the conjugation of $\mathbf{p}$ by $\mathbf{q}$

$$\mathbf{p}' = \mathbf{q}\mathbf{p}\mathbf{q}^{-1} \tag{11}$$

$\mathbf{p}$ forms a quaternion with real coordinate equal to zero. In equation 11 the Hamilton product was applied and p' is the new position vector after execution of rotation.

$\mathbf{q}$ is a unit quaternion and

$$\mathbf{q}^{-1} = e^{-\frac{\theta}{2}(u_x\mathbf{i}+u_y\mathbf{j}+u_z\mathbf{k})} = \cos\frac{\theta}{2} - (u_x\mathbf{i} + u_y\mathbf{j} + u_z\mathbf{k})\sin\frac{\theta}{2} \quad (12)$$

As a result the conjugation by the product of two quaternions is the composition of conjugations by these quaternions. Where follows that the rotation by $\mathbf{pq}$ is

$$\mathbf{pq}\vec{v}(\mathbf{pq})^{-1} = \mathbf{pq}\vec{v}\mathbf{q}^{-1}\mathbf{p}^{-1} = \mathbf{p}(\mathbf{q}\vec{v}\mathbf{q}^{-1})\mathbf{p}^{-1} \quad (13)$$

(with $\mathbf{p}$ and $\mathbf{q}$ as unit quaternions) showing the same result if rotating first $\mathbf{q}$ and afterwards $\mathbf{p}$ with a scalar component being zero necessarily.

Further, characteristics show that the inverse of a rotation is the opposite rotation, because $\mathbf{q}^{-1}(\mathbf{q}\vec{v}\mathbf{q}^{-1})\mathbf{q} = \vec{v}$. Moreover, rotating by twice the angle around the same axes corresponds to a square of a quaternion rotation which can be continued for any $n \in \mathbb{R}$ where q^n equals a rotation n-times around the same axis as $\mathbf{q}$. Also, the combination of two quaternion rotations to one rotation, according to the same rotations, can be done by

$$\mathbf{q}' = \mathbf{q}_2\mathbf{q}_1 \quad (14)$$

where follows, that any number of rotations can be combined to a single one [29].

Advantages

One advantage lies in the compactness of the notation by just four numbers, compared to the bigger Eulerian matrix representation. Additionally, it is possible to directly determine the rotation axis and angle with this notation, which is not the case in matrix representation, where it is normally very hard to observe those two parameters. Moreover it is the quaternion rotation which allows a smooth rotation in video games and other applications rather than a one step rotation. Most important, the phenomenon of the so called gimbal lock where one degree of freedom is lost, is avoided by the use of quaternions. However, one big disadvantage shows, that this method is only possible to perform rotations, which is a problem when other transform operations than rotations, like translations, are executed. A conversion from a matrix to quaternions requires only a square root and three divisions plus some additions in a worst case, thus being not very expensive in computational costs. On the other hand, a back transformation costs nine multiplications and 15 adds, which already states a more expensive transformation. [45]

3.2 Geometrical Background

3.2.1 Triangulation

Triangulation is a method that forms a triangle mesh using a cloud of vertices. Furthermore, this method is called a Delaunay-Triangulation [30] (in 2D) if the inner circle criterion is fulfilled. Therefore, the generated triangle's inner circle does not include other vertices than the edge vertices of the generated triangle. Thus, the smallest inner angle is maximized over all triangles, which is a preferred characteristic in Computer Graphics, since it minimizes the rounding error.

Generally spoken, the triangulated mesh is generated by connecting single vertices to triangles. In Figure 11 this process is visualized by performing a Delaunay-Triangulation with the use of random ordered vertices. This

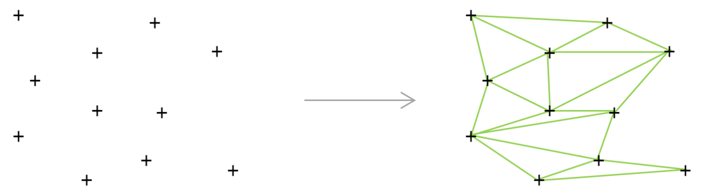


Figure 11: Triangulation process using a cloud of vertices (left) to generate a mesh consisting of triangles (right)

method provides an easy and precise way to form a mesh with surface information like used in winged edge meshes (described in the following section) by just using predefined points in the 3D space. An easy example is the formation of a cube model by just knowing the location of the four corner points. However, triangulating an unordered cloud of points has not only one unique solution rather then various different. Bern et al. [20] state in *Computing in Euclidean Geometry. chapter 2.2 Optimal Triangulation* the following characteristics of the triangles as quality measure for optimal triangulation:

- maximizing the minimum angle;
- no large angles;
- maximize the minimum height;
- conforming Delaunay-Triangulation.

3.2.2 Winged Edge Mesh - WEM

The winged edge mesh is a commonly used method to represent geometrical data in Computer Graphics based on the contribution of Baumgart in *A polyhedron representation for computer vision* [13]. Compared to other specifications of polygonal meshes like a node and element list, this one shows important improvements in accessing data.
WEMs consist of three different elements: vertices, edges and faces. Vertices are the node points where edges meet. Faces are surrounded by their edges, building the actual surface element. Therefore, depending on the mesh type a face holds various vertices and edges. Using a triangle mesh, for example, one face holds three edges and three nodes. Figure 12 gives a better understanding of this model. The theory of winged edge data structure is

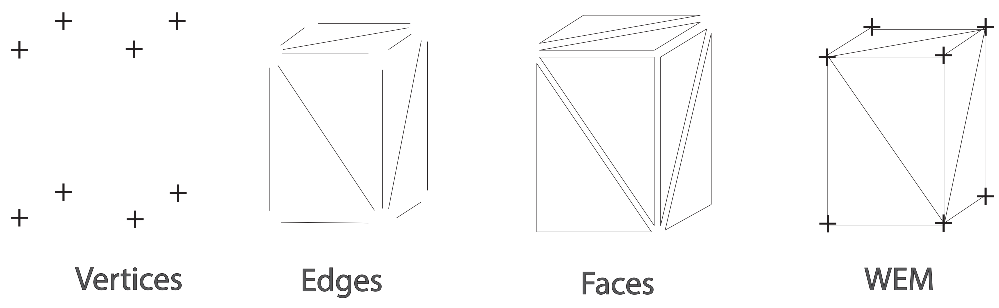


Figure 12: The Elements of the Winged Edge Mesh (WEM) data structure.

applicable when two or more faces come together sharing the same edge. Therefore, an edge table is generated, holding the following features of each edge:

- edge name;
- start and end vertex;
- left and right surface;
- left and right traverse.

Figure 13 together with Table 3 shows an general example.

Moreover, this data structures uses two more tables. The *vertex table* holding one entry for each vertex together with and edge incident on this one. Similar, the *face table* holding one entry for each face together with an boundary edge.

Type	**Number**	**%**
Lefort-I	4	0.7
LeFort-II	11	2.0
LeFort-III	9	1.7
Right zygoma	62	11.0
Left zygoma	60	10.9
Zygomatic arch, right	14	2.7
Zygomatic arch, left	11	2.0
Right orbital floor	60	10.9
Left orbital floor	70	13.1
Right orbit	8	1.6
Left orbit	9	1.8
Right upper jaw	13	2.5
Left upper jaw	11	2.1
Right upper alveolar	7	1.4
Left upper alveolar	9	1.8
Nose	32	6.1
Mandible symphyses	11	2.1
Right mandible	38	7.0
Left mandible	39	7.0
Right condylar neck	21	4.1
Left condylar neck	18	3.3
Right lower alveolar	9	1.8
Left lower alveolar	9	1.8
Condyle	5	0.8
Total	540	100

Table 2: The table shows the statistics of the collected data in Innsbruck, Austria of facial bone fractures caused by ski accidents.

Edge	Vertex		Face		Left-Traverse		Right-Traverse	
Name	Start	End	Left	Right	Pred.	Suc.	Pred.	Suc.
a	1	2	L	R	b	d	e	c

Table 3: Example entry of edge table according to figure 13

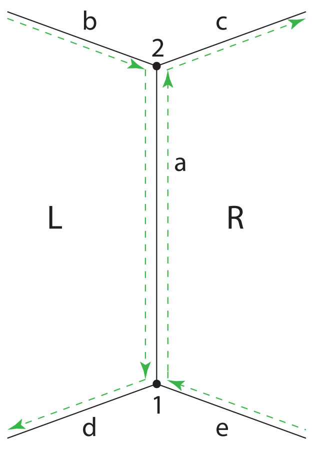


Figure 13: Two faces L and R sharing the same edge a building a winged edge.

Providing all this information, it is an easy task to access the desired elements of the data set. Further important for this contribution, is the introduction of normal vectors and face centroids. Figure 14 shows a single face, where **N**, the vector normal to the face's surface with it's origin C the face's central point (centroid), thus describing further characteristics used in my thesis. It is important to note, that mathematically, a face, like described here, has two normal vectors, one pointing outwards and the other one pointing inwards. However, the face is part of a model's outer surface, thus allowing to dismiss the vector pointing to the inside of the model, since it does not add useful information. Another part worth to mention and frequently used in this thesis, is the concept of patches. Under MeVisLab the WEM objects are represented in a special structure why it is important to introduce the WEM patch object. Therefore, a WEM object does not consist only of edges, faces and nodes, more this geometrics are first assigned to a patch. Further, one WEM object can hold several WEM patches. As a result, the hierarchical order is represented as follows:

$$object \rightarrow patch \rightarrow face \rightarrow edge \rightarrow node$$

3.3 STL-File Format and 3D Printing

This section is about the file format STL which stands for **S**tandard **T**eselation **L**anguage and is commonly used in CAD-systems for rapid prototyping, computer aided manufacturing and 3D printing [33].

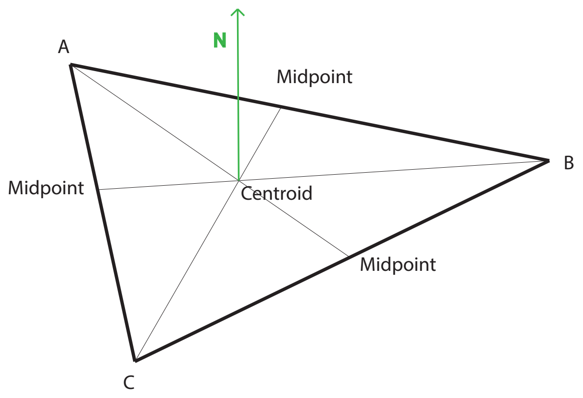


Figure 14: A WEM triangle face showing the centroid and surface's normal vector.

This format does not use any representation of texture, color or other attributes used in CAD formats. Further, it just describes the surface geometry represented through unstructured triangulated surfaces consisting of vertices of those triangles in a three-dimensional Cartesian coordinate system. Also, the STL file can be specified in two different ways, ASCII and binary format. Due to higher compactness, the second one is the more common one.

The 3D printing technology has gained lots of importance in the last years and still is fast developing. Since the STL-file format is a commonly used one in this still evolving technology, also the proposed methods are optimized for gaining a STL-file ouput data. More, there are three main steps in the process of gaining a final 3D physical model. Figure 15 shows those followed by a brief description [48].

Image Acquisition - This step is very important to the final quality of the resulting model, since the quality of the gained images influences directly the output quality. Many different imaging techniques like magnetic resonance imaging (MRI), CT, cone-beam CT, ultrasound (US), positron emission tomography (PET) or single-photon emission computed tomography (SPECT) are appropriate at this stage depending on the tissue of interest and it's location. The resulting images are frequently stored in DICOM (Digital Imaging and Communications in Medicine) [21] format.

Post-Processing - The 2D images serve as basis for this step, from which complex 3D models are generated using high performance computers

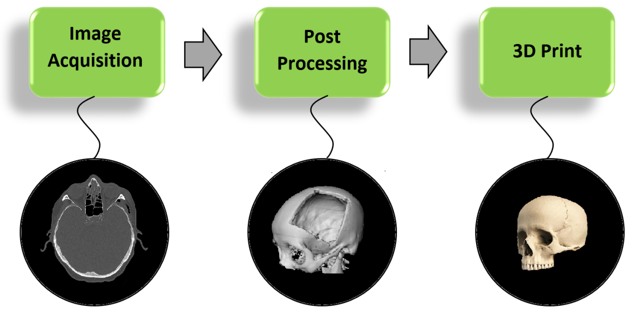


Figure 15: The figure shows the main steps in 3D printing together with an example each. Image Acquisition, for example by a CT scan, produces cross-section images in 2D (left). By further post-processing, a 3D computer model can be constructed (middle). Finally, the model is printed to gain the physical model (right).

and algorithms including segmentation, registration and visualization. The final model is saved in CAD format, whereby it is transformed to STL-file format to be able to be processed with a 3D printer. If this contribution should be categorized, it would fit best in the phase of post-processing since the 2D slices of the CT-scan are already provided but a 3D-printed, physical model is the next step.

3D Print - The final goal is to gain a physical 3D model based on the work gained in the post-processing step on the computer. Therefore, a range of different techniques have been developed, where additive fabrication is the most common one. By adding layer per layer, according to the model, the physical model is constructed piecewise. 3D printing has gained it's importance in the medical area [40] [39] due to the possibility of constructing nearly every geometry, even the most complex ones, very detailed and fast. This is why surgeons are very interested in developing this technology until it is usable in the daily routine thus reducing financial and time costs.

3.4 Computed Tomography

Nowadays, computed tomography is one of the most common methods used in medical imaging by showing increasing numbers of application in the last two decades in most of the countries [19]. By taking many x-ray absorption measures from different angles on one axes of the patient, these images are further processed forming the final cross-sectional image in 2D. The image aquisition happens by rotating a ray-sensor system around the patient who is fixed in the center of the rotation axes. A thrust in one direction of the patient allows to construct cross section images at different locations, referred to as the so called slices. By further combining a large series of slices, a 3D image of the patient, or parts, is possible to be reconstructed [22].

In CT images, the ray-sensor system casts a ray through the center of the rotation, spreading also through the patient to be able to be detected from the sensor placed on the opposite site. The sensor experiences diminution of the ray which is used for gray value calculation of each point. Further, each pixel is assigned to a gray value according to the tissue's specific ray attenuation. Using the so called *Houndsfield Units* (HU) [16] each tissue is assigned a gray value relative to the one of water ($HU_{water} = 0$). Therefore, being easier to compare. CT is everywhere applicable where the patient suffers from an issue causing deformations of the body structure. It is safely used by detecting bone fractures, issues with blood vessels, inflammation or bleeding detection. Another advantage is the relatively fast imaging procedure. By suffering from issues concerning soft tissue structures, MRI is preferred, even though the application costs are higher than the ones from CT the important advantage is the radiation free examination.

Even though, CT scans are very trustfully, they suffer from artifacts like beam hardening, motion artifacts, streak artifacts, partial volume effect or noise, to name a few [25]. However, increasing the ionization dose of the ray leads also to a better reconstruction of the inner body structure, why the most important issue is to lower the amount of radiation a patient is exposed to, by increasing or at least holding the image quality at the same time. Not only that high doses increase adverse side effects, also the risk of radiation induced cancer is elevated. Following there are listed some methods of how the radiation exposure can be reduced easily during CT scan examination wihtout noticeable loss of quality [12]:

- using new software technologies can decrease the exposure significantly;
- adapting the radiation dose to the tissue of interest;
- evaluate the most suitable examination method, e.g. full body scans are in most of the cases inappropriate.

3.5 MedArtis Miniplates

The implant models used in this thesis are so called miniplates rather than normal implant plates. Miniplates are reduced in their size and thus show several advantages. One of those is the effect of being easily bent due to the delicate shape of the plate allowing optimal adoption to the fractured bone's surface. Further, the use of cortical screws, able to be applied with this type of implant, ensures that nerve damage nearly never occurs. Of course, reduced forms like these, show some disadvantages like material damage due to too much bending in different directions or the loss of compression in interfragmental areas.

However, miniplates are commonly used, especially by the surgeons of the Oral and Maxillofacial departmant of MedUni Graz. Therefore, this thesis focuses on the three most implanted models by these surgeons, observable, in Figure 16. The three figures show the most used miniplates of MedUni

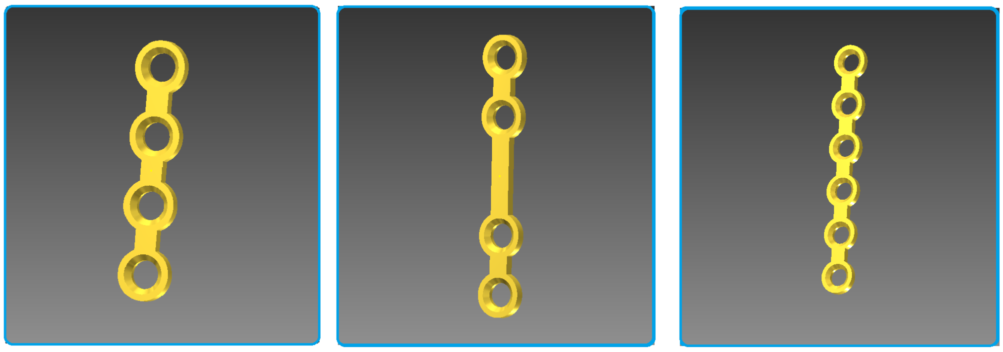


Figure 16: The three figures show the most used miniplates of MedUni Graz's Oral and Maxillofacial department. All models are of the Trauma 2.0 series and available in the MedArtis product catalog. The most left one with article number M-4138 showing an overall length of 23 mm consisting of two end ring elements and two middle ring elements. The model in the middle with article number M-4320, with the same properties as the left one but an extended bar of 9 mm and thus extended to 29 mm in overall length. Last, the most right one, with article number M-4322, with two additional ring elements and an overall length of 35 mm.

Graz's Oral and Maxillofacial department. All models are of the Trauma 2.0 series and available in the MedArtis product catalog [8]. The most left one with article number M-4138 showing an overall length of 23 mm consisting of two end ring elements and two middle ring elements. The model in the middle with article number M-4320, holds the same properties as the previous one but has an extended bar of 9 mm, consequently enlarging to 29 mm in overall length. The most right miniplate with is categorized with article number M-4322, added with two more ring elements and thus gaining an overall length of 35 mm. The implants also show the same properties in their plate thickness with a value of 1 mm for the whole series. For a full description of the implant, Figure 17 shows the miniplates' general dimensions.

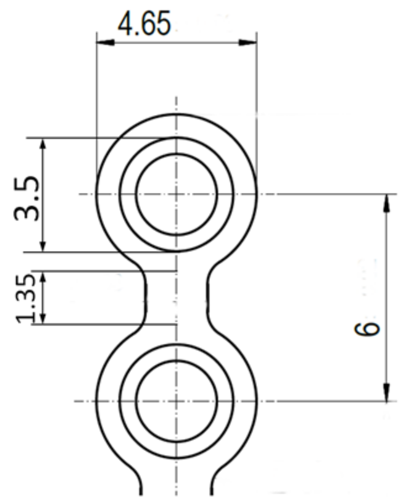


Figure 17: General dimensioning of miniplates of the MedArtis' Trauma 2.0 series [6].

Further, all the miniplates are made of pure titanium (ASTM F67, ISO 5832-2) which shows not only very good mechanical characteristics, but is also bio compatible.

3.6 MeVisLab

3.6.1 General

This section describes the medical imaging platform MeVisLab (Medical Visualization Laboratory) which's software development kit (SDK) is used for implementing the proposed methods. The information provided in this section is a recap of the most important chapters of MeVislab's *Getting Started Tutorial* [9] and *Reference Manual* [11]. MeVisLab provides basic and advanced algorithms for medical image processing and visualization where also an environment for individual module generation under C++ is included as well as an environment for implementing a user interface with MDL-script (MeVis Description Language). It is possible to construct networks using modules of different types for processing images.

3.6.2 Development

Under MeVisLab it is possible to develop under three different levels. First, via visual level programming by using pre-installed modules, individual image processing and visualization is able to be combined to complex networks using the graphical user interface. Another possibility is the usage of the scripting level making use of the python scripting language. Via the C++ programming level, it is possible to update or reprogram existing algorithms or even generate new ones in individually designed modules using the platform independent C++ class library. Additionally, with MDL script it is possible to design user interfaces hiding the underlying complexity thus avoiding confusion of the medical user and providing an easy to use surface for interaction. Figure 18 shows a very low level network with a minimum amount of modules by providing a module for loading the image data, the algorithm module, which applies a threshold function in this example, and the viewer module for visualization.

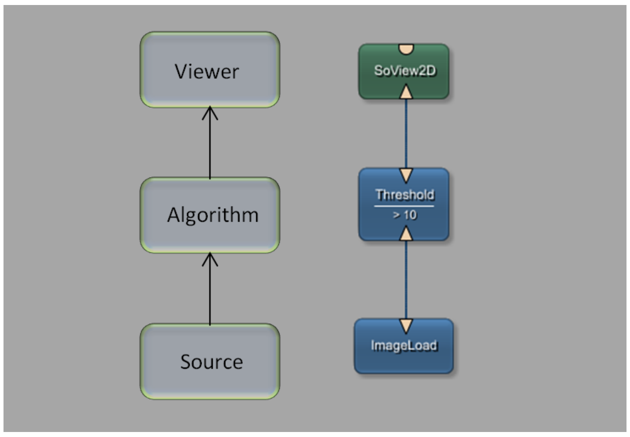


Figure 18: Basic module processing pipeline showing a network of three modules, an image source for loading the data set into the network, the algorithm module to process the data and finally a viewer module which enables visualization of the processed image.

3.6.3 Modules

In MeVisLab the network concept is based on a graphical representation of modules with their specific functions for image data processing. In the environment, three different types of modules are integrated, observable in Figure 19. The basic module types are ML modules (blue) which are page/-

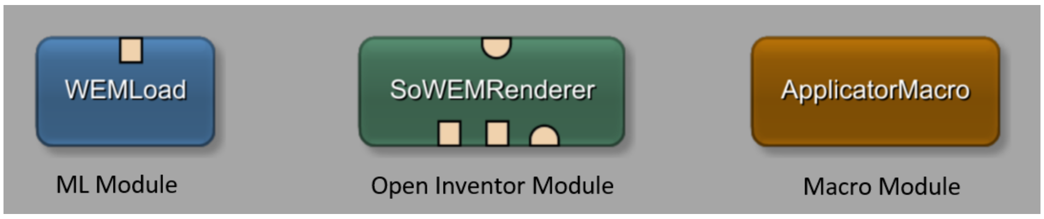


Figure 19: The basic module types are ML modules (blue) which are page/-patch based and demand-driven, Open Inventor modules (green), providing scene graphs in 3D, and Macro modules (brown), consisting of a combination of other modules.

patch based and demand-driven processing modules, Open Inventor modules (green), providing scene graphs in 3D, and Macro modules (brown), consisting of a combination of other modules. Further, not all but most modules, have connectors which are indicated either by a half circle, a triangle or a rectangle on the bottom (input) and the top (output) of a module. The three

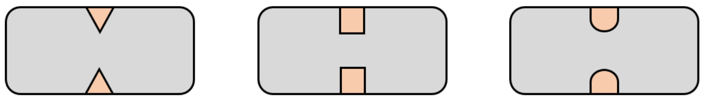


Figure 20: The three types define a connection for ML images (triangle), a base object connection indicating pointers to data structures (rectangle) and an inventor scene connection (half circle).

types define connections for ML images (triangle), a base object connection indicating pointers to data structures (rectangle) and an inventor scene connection (half circle). By connecting these connectors, the data transmission between modules is enabled. This connections are represented by a blue line between the modules, like shown in Figure 18. Beside this data connection a parameter connection is also defined enabling to connect single parameters between modules, indicated by a two sided arrow.

Type	Description
.mlab	network file holding all information about it's modules
.def	Module definition file
.script	MDL script file, includes the user interface
.mhelp	description of all used fields
.py	python file used for scripting
.dcm	DCM part of the DICOM file (if used)
.tiff	TIFF part of the DICOM file (if used)
.mlimage	6D image with DICOM tags (if used)
.gvr	Precomputed octree file for direct volume rendering

Table 4: The table shows the most important files used in the MeVisLab environment.

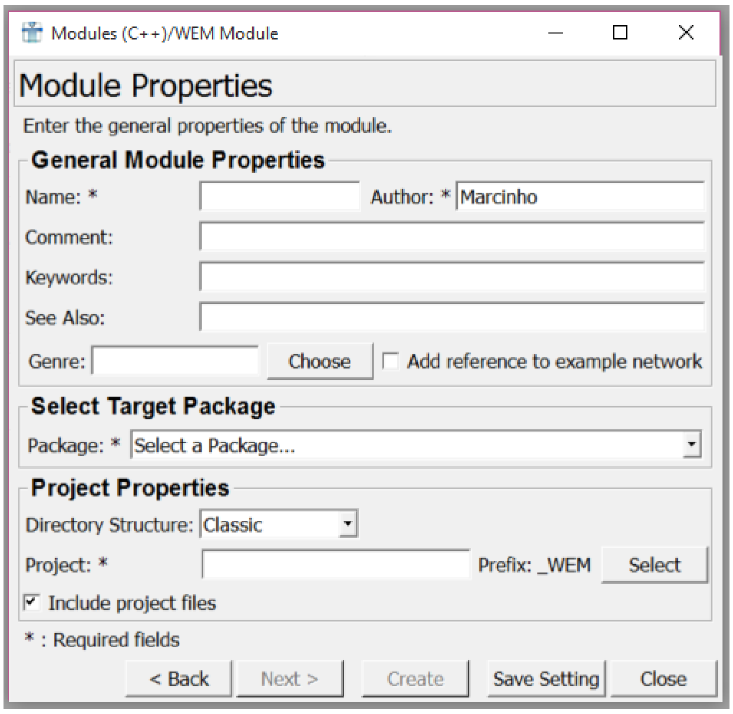
3.6.4 Important Files

Table 4 gives an overview of important files and their content used with the MeVisLab SDK.

3.6.5 Example Setup of a Module

In this section I show how to setup a module like used twice in the proposed methods. Therefore the integrated *Project Wizard* is used.

First, the wizard is started by selecting *Run Project Wizard* in the MeVis-Lab's file menu. If not yet created a new package has to be created, where the following generated modules are organized. Next, the *WEM Module* link for generating a WEM module is selected. The panel presented is shown in Figure 21. The wizard asks you to enter a unique name, as listed later in the database, an author name, an already created package and a project name which are mandatory before continuing the setup. Remaining fields are recommended to enter as well, since they will provide additional information for other parties to get a better understanding of the underlying process. After providing the required information, the *Next* button is enabled, leading to a panel where the module type has to be chosen. The WEM generator provides only one output since it's objective is to generate a new WEM using other types of inputs. Choosing the WEM processor, like done in this thesis, the input WEM is processed to form the output WEM, thus providing one input and one output. WEM inspector modules analyze properties of the input WEM, leading to only on input field. The following panel allows to include additional fields, which I prefer to setup directly in the C++ files, which is also explained in this section. By *creating* the module, the necessary files


Modules (C++)/WEM Module

Module Properties

Enter the general properties of the module.

General Module Properties

Name: *
Author: * Marcinho
Comment:
Keywords:
See Also:
Genre:
Choose
☐ Add reference to example network

Select Target Package

Package: * Select a Package...

Project Properties

Directory Structure: Classic
Project: *
Prefix: _WEM
Select
☑ Include project files

* : Required fields

< Back | Next > | Create | Save Setting | Close

Figure 21: This figure shows the project wizard's panel for setting up the modules properties.

are generated and included folders are shown to the user. Most important is the Visual Studio project file. But before opening that one, we have to make some adoptions to the .pro file. The line where the used projects are included has to be adjusted to

CONFIG+=dll ML MLBase MLWEM

to be able to use necessary commands for WEM processing. By running the .bat file the package is updated. The continuing step, is to open the Visual Studio project file (.vxcproj) to adjust the included files. The generated header file, for a module named *test*, is shown in the following code example Listing1.

```
#pragma once

// Local includes
#include "_WEMtestSystem.h"
#include <WEMBase/WEM.h>
#include <WEMBase/WEMModuleBase/WEMProcessor.h>

```

```
// ML includes
#include <mlModuleIncludes.h>

ML_START_NAMESPACE

//!
class _WEMTEST_EXPORT test : public WEMProcessor
{
  //! Implements interface for the runtime type system of the ↩
     ML.
  ML_MODULE_CLASS_HEADER(test)

public:

  //! Constructor.
  test (std::string type="test");

protected:

  //! Destructor.
  virtual ~test();

  //! Initialize module after loading.
  virtual void activateAttachments();

  //! Handle field changes of the field \c field.
  virtual void handleNotification (Field* field);

  //! _process()-routine for correct processing.
  virtual void _process();

private:

  //! The main processing routine. Here, the own mesh algorithm↩
       can be implemented.
  void myAlgorithm();
};

ML_END_NAMESPACE
```

Listing 1: Initial header file for test module.

The wizards sets up the header file already using the necessary includes, the constructor, the destructor, a method to initialize the module, the *handleNotification* method, a process routine and the *myAlgorithm* method.

Constructor This is where all the parameters and in-/output fields are initialized when opening a network.

activateAttachments This method is not used in this thesis, therefore it is deleted in all implementations.

handleNotification In this method, it is checked weather the processing method(s) are executed or not. Since most of the modules start their process on field changes, here it is checked if a field has changed.

process This method is called if the process is executed and guarantees correct processing.

myAlgorithm Called by the *process* method, the code developer implements his code in this section.

Since we defined a WEM processor module, one WEM input and one WEM ouput are already defined and accessible via the variables *_inWEM* and *_outWEM* as a *WEMPtr* variable thus pointing to a WEM object. Adding new inputs or outputs requires a definition of the appropriate fields in the header. If additional parameters want to be added to the module as well, also these have to be declared in the header first. The next code example Listing2 shows how this is done by adding an additional input for a marker list (*_inMarkerFld*) and an additional parameter for toggling a button on/off as a boolean (*_OnOffFld*).

```
  .
  .
  .
private:

  //! Field for additional parameters
  BoolField* _OnOffFld;

  //!Field additional input marker lists
  BaseField* _inMarkerFld;
  .
  .
  .
```

Listing 2: Declaration of in-/outputs and parameters.

As the variables are now declared, they have to be initialized by the constructor. The third code example Listing 3 shows how an initialization is performed using the declared variables from the previous example.

```
test::test(std::string type) : WEMProcessor(type)
{
  // Suppress calls of handleNotification on field changes to
  // avoid side effects during initialization phase.
  handleNotificationOff();

  // Add in-/output fields to the module and set their values.
  _inMarkerFld = addBase("inMarkerList", NULL);
  _inMarkerFld->addAllowedType<XMarkerList>();

  // Add parameter fields to the module and set their values.
  _OnOffFld = addBool("On/Off", false);

  // Reactivate calls of handleNotification on field changes.
  handleNotificationOn();
}
```

Listing 3: Variable initialization in the constructor.

First the method for handling the notifications is turned off during the execution of the constructor to avoid side effects by calling the method *handleNotificationOff()*. Finally the handling of notifications is activated again by using the *handleNotificationOn()* method. To the input marker list field *_inMarkerFld* a base is added by forwarding a name (inMarkerList) and a value (NULL) by the *addBase* method. Also the allowed connection is set, which is of a *XMarkerList* type. For the parameter variables only the definition of a name and an initial value has to be done, since no inputs nor outputs are included in such a variable type. As a result, the variables are now set up, initialized correctly and able to be used in further implementation.

4 Related Work

Surgeries of oral and maxillofacial defects use different pre-planning processes since on one hand, there are various fracture sites requiring different kinds of treatments and on the other hand there is a range of different treatment methods possible to use. However, centerline detections can be categorized in two main types. First, there are those using rapid prototyping to gain pre-bent implants and second, those using computer-aided pre planning software resulting in patient specific implants. However, the partners from MedUni Graz, and therefore I assume that also other medical institutions, do not use any of those two pre planning methods but bend the implant in a conventional way on the open cut patient which increases some risk factors like the longer lasting surgery due to frequent adjustments of the implant by bending. Anyway, this chapter gives a brief review on the two methods using the pre planning step, in comparison to the proposed one.

4.0.1 Pre-bent Implants using Rapid Prototyping

According to Bell [15] in his article *Computer Planning and Intraoperative Navigation in Cranio-Maxillofacial Surgery*, rapid prototyping (also named stereolithographic modeling) is a useful method to plan implants for complex facial reconstructions. Therefore, based on a CT-scan, a 3D physical model is generated, which serves as a bending guide for the implant or as a template for constructing a patient specific one [14]. This treatment also uses advanced software tools like Materialize's MIMICS [4] being able to reposition fractured bones to model the final outcome in a 3D computer and physical model [50]. Further, this method is also used in combination with a so called transfer key [34]. This key, serves as a guide for optimal implant positioning and is constructed to fit the healthy bone perfectly, thus allowing optimal placement of the actual, load bearing, implant, relative to the key. Even though this method is very precise, it holds disadvantages concerning financial and time expenses [50]. The used software is of commercial nature and also the training and application of the software is time consuming, thus expenses rise. Further, in some cases the generated models need to be adjusted during surgery, since the repositioning could not be performed the same way it was performed in the computer model due to inaccuracy or surrounding soft tissue, making the planned placement more complicated. The method proposed in my thesis is both, less time and cost consuming. By providing a software which is easy to handle and understand, the surgeons will not have the problem of spending lots of time on generating the implant. The CT-scan data, which is mandatory to derive for facial surgeries, can easily be loaded into the application and by additionally providing an easy to use and clear graphical user interface, the surgeon has to perform just a few clicks until gaining the desired, perfectly fitting implant. From here on, the 3D printer is setting the time limits, but resulting in less time consumption than the convencional method anyway since the printing process of the miniplate is less time consuming due to minor size than the big full skull prints. Additional, this properties, namely the use of less material and less consummated time, affect financial expenses directly. Another very important advantage of the method proposed in this thesis lies in the mechanical properties. Using a 3D physical model as a guidance, harms the implant's structure since the raw and straight material has to be bent for an optimal fitting shape [17]. Even though, this deforming process does not lead to difficulties in most of the cases, the bending holds a potential of failure which is avoided by directly generating the already bent implant. However, in both methods additional adjustments may have to be applied during surgery, since neither a physical nor a computer generated model catches the real life perfectly. Anyway, due to this fact, adjustments of directly printed implants need less absolute deformation, since the step of pre-bending is skipped.

4.0.2 Computer-aided Pre-planning

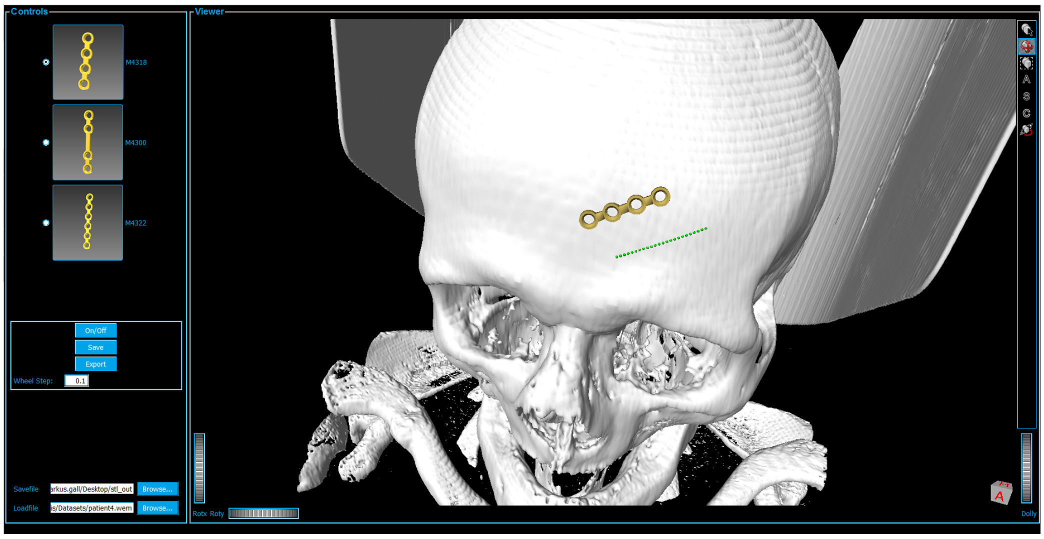
Like already mentioned in the previous section, there exists quiet a range on software tools for oral and maxillofacial surgery applications. Most of them focused on the repositioning of fractured bone tissue rather than implant generation like proposed in the article *A New System for Computer-Aided Preoperative Planning and Intraoperative Navigation During Corrective Jaw Surgery* proposed by Chapuis [32]. Further, a very powerful software tool is provided by Materialise [3] which has a broad range on application fields concerning software and services for medical 3D printing. The software tool PROPLAN CMF allows, like mentioned above, to reposition hard tissue parts and calculate the resulting outcome including soft tissue stresses due to the applied deformations, for surgical pre-planning (website: http://hospital.materialise.com/mimics-care-suite-cranio-maxillofacial-surgeons, last accessed on May, 2016 [4]). Additionally, the OBL software provides a tool for patient specific titanium mesh implant generation. Even though, with this packages a professional and comprehensive software kit is provided it lacks a few disadvantages. First, the commercial available software has to be purchased which leads to an elevation of financial costs. Second, surgeons, for example those of the LKH Graz, use miniplates like provided by MedArtis rather than personalized titanium mesh implants. Therefore, the software described in my thesis, is fitting better for the use of miniplates and reconstruction plates, since it is adopted for generating and placing MedArtis miniplates in an optimal manner. Upgrading to other types of implants is a question of not even one day of work, depending on the expertise level of the developer. Generally, commercial software is not able to be extended at all or does not provide interfaces for extensions. Again, the software provided in this thesis shows advantages, by being able to be rebuild using MeVisLab on a free available version for scientific research. Also, it enables the creation of customized extensions for anybody's needs, thus allowing for example the generation of other implant types or adding the additional feature of bone repositioning. Even better, that the software provided is editable by everyone to fit one's needs, independent if it is an update for new types of implants or even another area of application like vertebral disk or spine fixation.

5 Methods

5.1 Overview

This section gives the reader an introduction of how the proposed software is used to gain a general understanding and an basic idea of what the software is capable of and how underlying ideas are set into practice. Therefore, in Figure 22 the user interface is shown to be able to link a picture to the later mentioned interactions, calculations and objects, followed by Figure 23 showing the flow chart diagram illustrating the software's work flow to gain the desired outcome. Briefly summarized, the user first has to load the dataset into the application to be able to set an initial point defining the implant's center position, followed by setting up the direction and choosing the appropriate implant model necessary for calculating the baseline, which serves as an simplified implant model allowing to save computational time but still gives the user a clear and accurate response. If satisfaction is reached, the user visualizes the implant which is now able to be saved for parallel implant generation or exported for locally and permanently storage. Again, this section just shows an overview of the application but detailed information on the underlying processes including details concerning the user interface are provided in later sections referred to in the following more detailed lines.
Data Set Selection - orange
First thing to do, after opening the application, is to load the patient's data set by choosing a path to the locally stored WEM file by clicking on the *browse* button of the *Loadfile* field in the user interface. Having a look at Section 5.2, shows detailed information on this topic. The result of this user interaction already is visualized by the viewer. The viewer also has implemented some controls like options for zooming and rotating the camera.

Implant Setup - grey
Second, again it is the user who has to set up the implants properties. Therefore, he has to choose the center point of the implant by setting the initial point on the skull. This is done by just holding the *ALT* key and clicking on the desired position. The *Baseline Calculation* algorithm (detailed described in Section 5.5) will now start calculating the Baseline (green line) in an initial position depending on the mouse wheel value and the initially set point. This line shows the location, direction and the approximated length of the later generated implant including the underlying curvature and should serve as a



Wheel Step: 0.1

Savefile arkus.gall/Desktop/stl_out

Loadfile is/Datasets/patient4.wem

Figure 22: User interface, separated in *Controls* and *Viewer* boxes. The *Controls* box holds the implant model selection (currently three types available), a further box for controls like the *ON/Off* button for visualization, *Save* for saving the current implant and *Export* for saving the generated implants locally, together with the *wheel step* field for adjusting the accuracy of the mouse wheel. The two fields on the bottom of the *Controls box* let the user choose the path of the patients data set location (Loadfile) and the path for saving the outcome (Savefile) by using the *export* button. The *Viewer* box shows the data set (white skull) attached with a miniplate (gold) and the baseline (green) with further controls for a flexible and user friendly visualization.

simplified version of the resulting implant, thus giving an idea on how the implant will look like later. In Figure 22, which shows the user interface, the green dotted line is possible to be observed as well, placed on the patient's forehead. The length of this line depends on the selection of the implant model, by marking one of the radio buttons in the user interface's control box, since they vary in length, different models set the baseline's length to a different value according to the model's dimensions. Additionally, by turning the mouse wheel, the implant is able be rotated, thus the direction of the baseline is able to be altered to match the user's desired placement and orientation. The field *wheel step* lets the user choose the accuracy of this rotation for high precision positioning. Also the initial point, and thus the implant itself, can be readjusted by holding the *ALT* key (activates pick mode and deactivates view mode as long as held) and clicking on the new

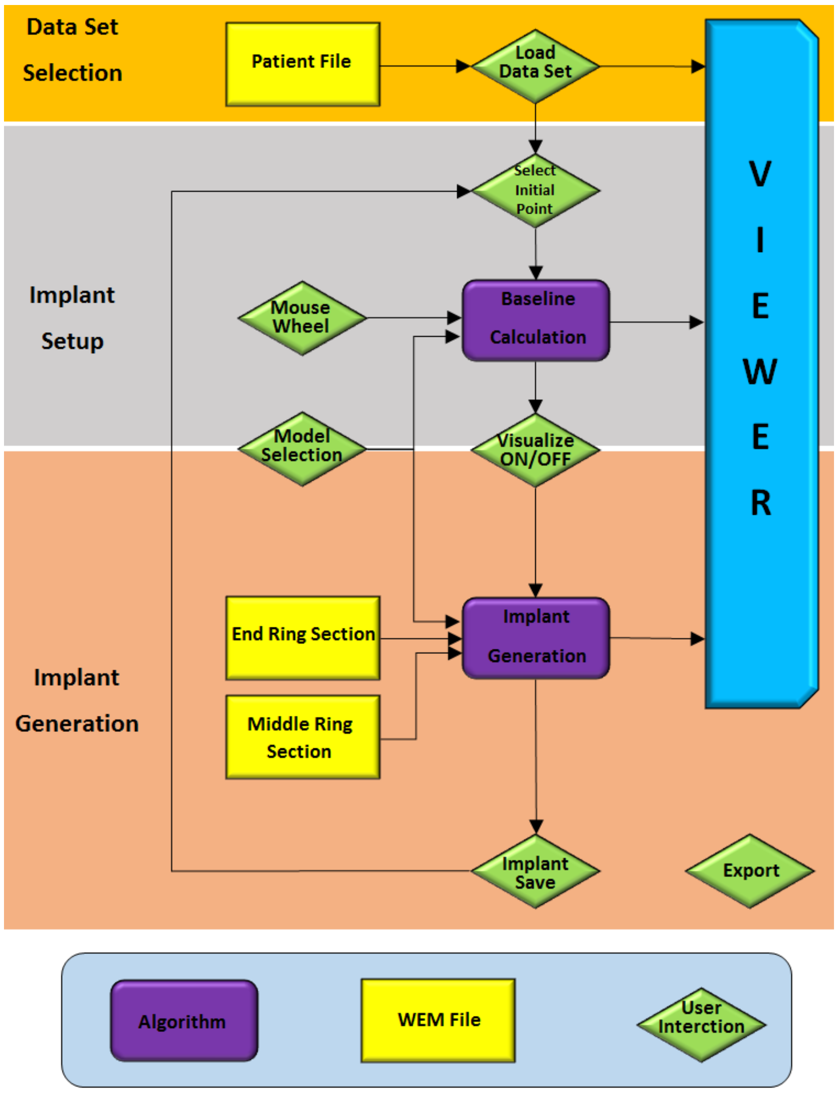


Figure 23: Overview of the implant generation. In the orange section, the user loads the patient's data set which is then visualized by the *viewer* block. Followed by the implant set up (grey section) where the user sets the initial point (implant center). Next,the baseline is to calculated which shows the user the current position and direction of the implant, depending on the mouse wheel value, the selected implant model and the initial set point is also visualized by the viewer. In the apricot section the implant is calculated (*Implant Generation* Block) by activating the *Visualize ON/Off* button. For the implant generation also the implants ring sections, middle and end parts, are required. The final implant is then visualized together with the baseline and the patient's data set. By using the block *Implant Save*, the user saves the current implant(s) for this session and starts setting up new ones. By exporting, the current visualized implants are stored as a STL file on a local path.

position on the data set's surface at any time. The step of first only visualizing the baseline rather than showing the implant directly is used to gain a more comfortable user experience since generating the model takes a few seconds to calculate, resulting in latency time which has to be avoided. As a result the compromise of adjusting the baseline in a fast manner followed by a latency time just during the finishing generation process of the implant, leads to a far better experience.

Implant Generation - apricot

After setting up the implant's position, direction and model type, the user has to click on the *ON/OFF* button in the user interface, which accords to the green, diamond shaped block in the overview graphic, named *Visualize ON/OFF*. Toggling this button, results in visualizing, or not visualizing, the current implant, depending on the baseline, performed by the *Implant Generation* algorithm. In addition to the properties of the baseline like position, direction and implant model, this algorithm also uses the implants ring sections, namely the end and middle elements for constructing the final implant. More details are provided in Section 5.6. Depending on the implant model's dimensions the positions and orientations of each ring element are than calculated followed by further generating the bridge sections, also according to the model's specifications. Important to mention is, that the ring elements have been constructed in Autodesk's Inventor and are permanently loaded into the application. Finally, the implant (gold) is visualized as shown in the user interface, Figure 22. Currently, for the proof of concept, the catalog of available implants is limited to three different models from the Medartis Modus 2.0 series which are characterized in section 3.5. Despite the fact that the implant's setup can be altered during every step, it has to kept in mind, that rotating the implant or adjusting the position in visualizing mode takes a lot of computational afford and thus may result in latencies. For optimal usage, it is therefore recommended to adjust these characteristics only during inactivated visualization. In the case of observing a satisfying result, the user can save the implant to continue setting up another implant while visualizing the previous generated and saved ones. This is done by using the *Save* button. Using this button does save the implant for this session only, thus allowing the generation of multiple implants being visualized next to each other. In other words, with this function it is possible to visualize more than one implants at the same time, e.g. for medical cases that require several implants like the LeFort fractures. Also, it is important to know the difference between the *Save* and *Export* buttons. Second, does export all implants, including the current (may not be saved yet) and all saved ones to the path, given in the *Savefile* field, as STL file. Again, export actually also means to save, but saves the implants locally on the computer in STL format and permanently, rather then the *Save* button which does save the currently processed implant for this session to the other already saved implants. Meaning, that as long as an implant is not saved, it is possible to be readjust.

Another option which is permanently able to be used, as long as the baseline is already generated, is to change the baseline's marker points in any arbitrary direction by just changing it's position by drag and drop. Since a three dimensional space is represented on a two dimensional plane (screen), it is sometimes not an easy task to perform.

5.2 Data sets

My thesis is optimized for working on real patient data why it is crucial to use data sets like those occurring in the surgeon's daily routine, suffering from different facial defects like LeFort fractures, mandible breaks or similar. Therefore a set including data of more than ten subjects is provided by the partners of MedUni Graz of the Oral- and Maxillofacial department and my supervisor Jan Egger of the Institute for Computer Graphics of the Graz University of Technology in DICOM or nrrd format.

As the software is built for STL-file input, it is necessary to convert the provided files first, for example by using the MeVisLab network shown in Figure 24. The Figure shows an example of converting the provided nrrd-files into STL-files by using a network constructed in MeVisLab. The lowest block named *itkImageFileReader* handles the loading of the image file, followed by the block for the actual conversion (middle, *WEMIsoSurface*) and finishing with the optional block for saving the generated WEM/STL-file named *WEMSave*. It is also possible to skip the saving process and link this network directly to the main network, thus, instead of inter saving the file and loading it again, one can spare the step of generating a new file. Moreover, it is important to choose the right parameter settings for the *WEMIsoSurface* module where the most important ones are described briefly in Figure 25. This figure shows an example set up for converting the provided nrrd-files into STL-files. The upper box sets up the used grey level values using the minimum and maximum levels of the provided file or using manually added ones. It is recommended to use the image's maximum value but manually choose the minimum value. The lower box provides an option for voxel sampling, differentiating between *Voxel Sampling* where voxels according to the shown number are sampled to one voxel, equal to the *Cell Extension* option

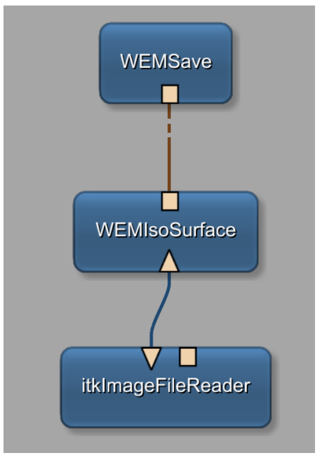
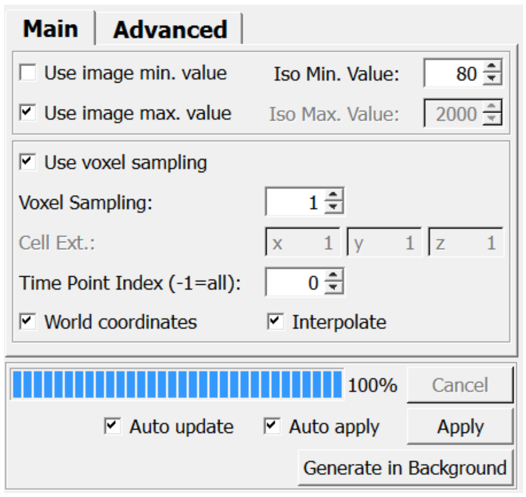


Figure 24: Example of converting the provided nrrd-files into STL-files by using a network constructed in MeVisLab. Lowest block handles the loading of the image file, followed by the block for the actual conversion (middle) and finishing with the optional block for saving the generated WEM/STL-file. It is also possible to skip the saving block and link this network directly to the main network instead of inter saving the file and loading it again in the main network.



☐ Use image min. value — Iso Min. Value: 80

☑ Use image max. value — Iso Max. Value: 2000

☑ Use voxel sampling

Voxel Sampling: 1

Cell Ext.: x 1 y 1 z 1

Time Point Index (-1=all): 0

☑ World coordinates ☑ Interpolate

100%

☑ Auto update ☑ Auto apply

Figure 25: Example set up for converting the provided nrrd-files into STL-files. The upper box sets up the used grey level values using the minimum and maximum levels of the provided file or using manually added ones. The lower box provides an option for voxel sampling differentiating between *Voxel Sampling* where voxels according to the shown number are sampled to one voxel. This also equals the *Cell Extension* option with three similar parameters for x, y and z

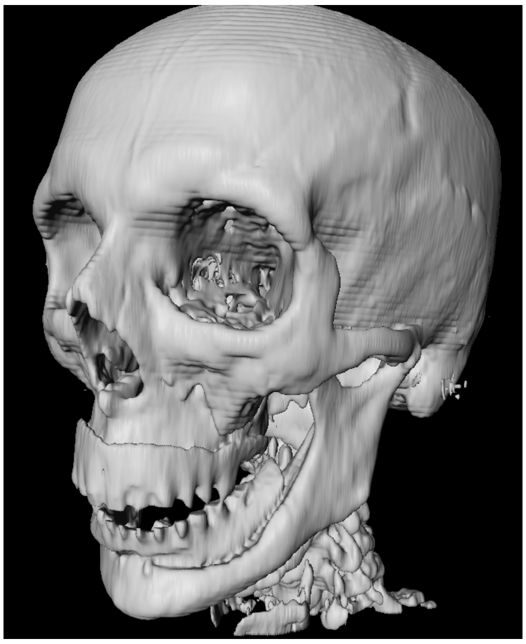


Figure 26: Dataset of healthy patient where no fractures are able to be observed, after conversion from nrrd to STL format.

with three similar parameters for x, y and z.

In the following Figures 26 and 27 two used datasets are shown, where the upper one suffers from a zygomatic bone fracture marked with a red circle and the second, lower one showing a healthy patient, without any fractures possible to observe.

5.3 Network

Opening the developed software, first the user interface pops up. By minimizing or closing this window one will only see the macro module as shown in Figure 28. The solution of providing only the macro module holds several benefits: First, this method enables the implementation of a user interface which is able to be linked easily with all underlying modules from the internal network. Also the wrapping of the network to only one module, gives protection to unintended interactions by the user, for example surgeons which are using the software, may click accidentally on a module's node resulting in disconnecting a link leading to failures, probably not able to be remedied by themselves. Consequently, this method provides a two level failure protection since the surgeon has to open first the internal network, which

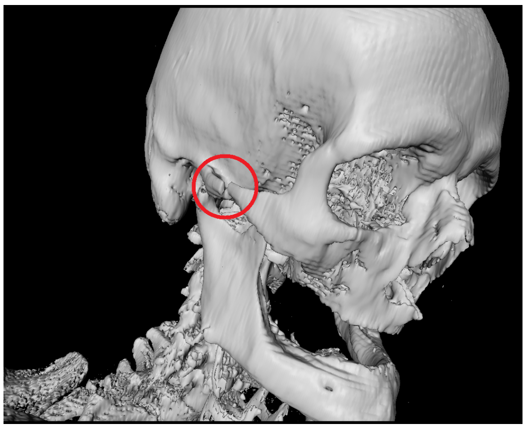


Figure 27: Dataset of patient suffering from zygomatic fracture on the right side, after conversion from nrrd to STL format.

Figure 28: The macro module which is observable after minimizing or closing the user interface, holding the underlying main network to protect it from unintentional user interaction.

is only accessible through the module's menu, until being able to harm the software. The main network is able to be accessed by right clicking on the macro module and choosing *Show internal network* in the menu. Afterwards, a new tab opens, presenting the internal structure as shown in Figure 29. The process of generating implants, starts first by loading the patient's dataset with the use of the *WEMLoad* module, permanently being rendered by the *SoWEMRenderer1* module and visualized in the *viewAll- SoCustomExaminerViewer* block. Next step is to choose the location of the initial point, marking the implants center, by clicking on the skulls surface during pressing the *ALT* key, for activating the pick mode. This interaction is detected and stored by the *So3DMarkerEditor*. It is also this module which provides the first set marker to the *CurvatureCulc* block, thus enabling the calculation of the baseline which determines the implant's curvature by using the patient's skull. As well the initially set point and the baseline are able to be visualized with the use of *So3DMarkerEditor* for the initial set point visualization and *So3DMarkerEditor1* for the handling of the baseline points. Afterwards, the baseline information is sent to the *ImplantGen* module which generates the implant, dependent on this information and further WEM files permanently

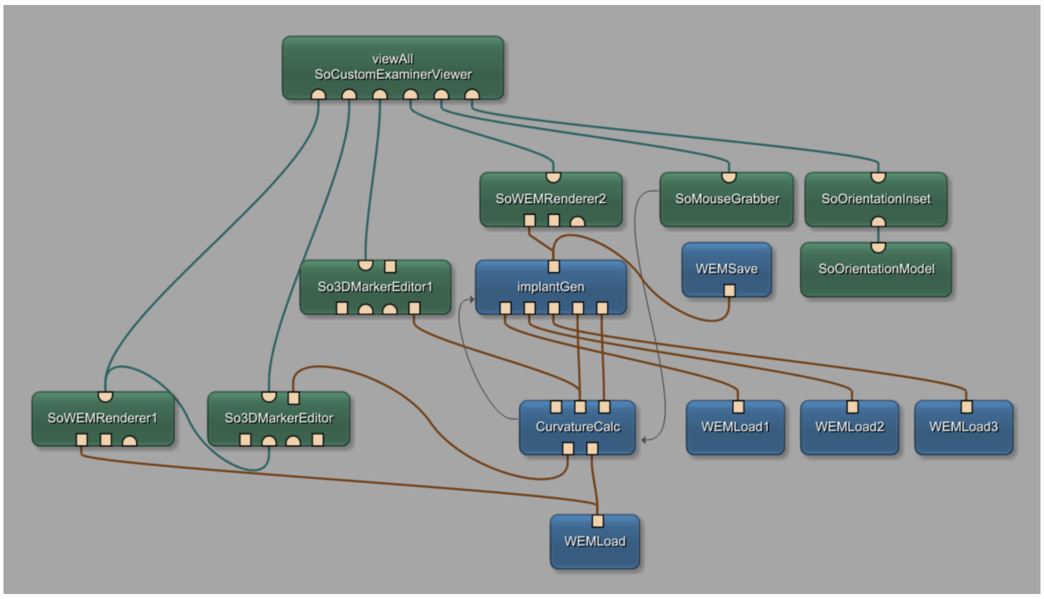
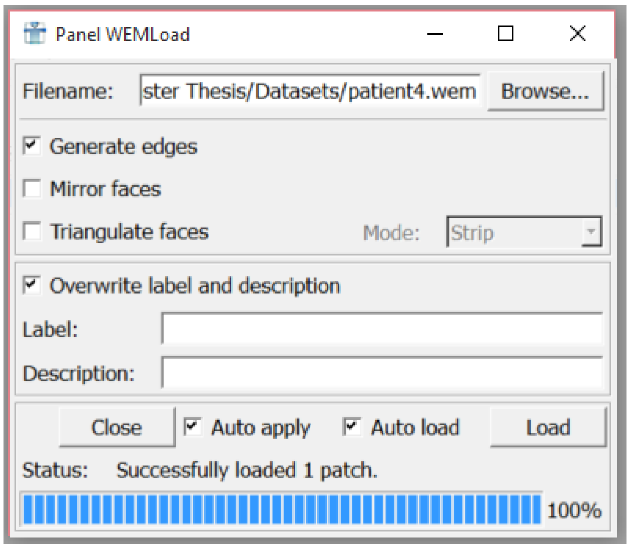


Figure 29: The main network observable by showing the internal network of the macro module.

stored in *WEMLoad1, WEMLoad2* and *WEMLoad3*. This three blocks hold the different ring elements of the final implant, where *WEMLoad1* holds the end elements, *WEMLoad2* the middle elements and *WEMLoad3* angle elements for implementing implants with a 90 degree angle. The generated implant is then able to be rendered with the *SoWEMRenderer2* module and visualized afterwards. Also, the final outcome is able to be stored permanently on the local drive by making use of the *WEMSave* module. The additional modules increase the handling and user experience, like the *SoMouseGrabber* enables rotating the basleline, and therefore the implant as well, by just turning the mouse wheel. Further, the *SoOrientationInset* and *SoOrientationModel* let the user choose from a range of different models visualized in the lower right corner of the viewer panel, serving as orientation help when rotating the viewer's camera.

5.4 Used Modules and Intermodular Linking

Following, each of the included modules is described more in detail to understand the function and the inter modular linking. This happens bottom up according to Figure 29 where modules with the same algorithm are grouped


Panel WEMLoad

Filename: ster Thesis/Datasets/patient4.wem Browse...

☑ Generate edges

☐ Mirror faces

☐ Triangulate faces Mode: Strip

☑ Overwrite label and description

Label:

Description:

Close ☑ Auto apply ☑ Auto load Load

Status: Successfully loaded 1 patch.

100%

Figure 30: Shows the panel of the WEMLoad module. The path gives the location of the patient's dataset. All other options are set to default.

together in one section since they fulfil the same task but with different connectors.

5.4.1 The WEMLoad Module

As the name already states, this module loads the initial WEM-file holding the patient's data which is also the module's only output in WEM-format. Figur 30 shows the panel of this module. The path able to be chosen in the panel marks the location of the patient's dataset. This field is linked with the user interface's *Loadfile* field. All other options are set to default.

Inputs:	None		
Outputs:	WEM-file	$\rightarrow$	CurvatureCalc, SoWEMRenderer1

Beside the *WEMLoad* module there exist three more of equal type named *WEMLoad1*, *WEMLoad2* and *WEMLoad3* providing the ring elements for generating the implant. This modules have the same setup as the *WEMLoad* module but differ in the files path location according to the implant element. Figure 31 shows the stored WEM-files for each of those three additional blocks.

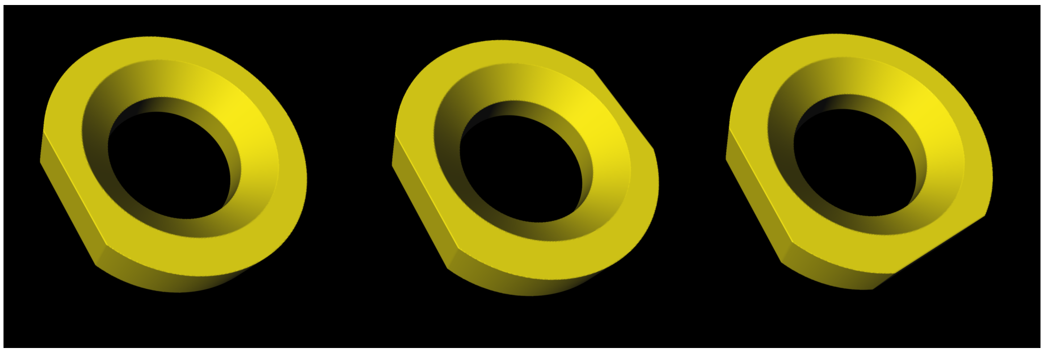


Figure 31: Shows the three ring elements of the implant provided by WEMLoad1 (left, end parts), WEMLoad2 (middle, center parts) and WEMLoad3 (right, angled part)

5.4.2 The SoWEMRenderer Module

This module is a standard one used in every network where WEM-objects have to be visualized, as the geometrical data needs to be rendered. This type of module is used twice in the main network using default settings.

SoWEMRenderer1
SoWEMRenderer1 renders the patient's input data, thus allowing it to be visualized. Further, it provides the skull surface for selecting the initial point using the connected marker editor module.

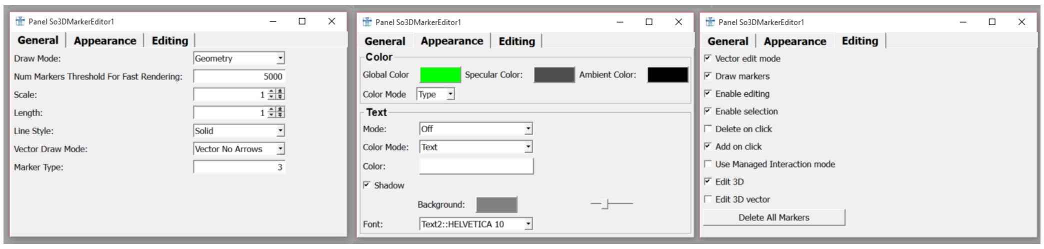
Inputs:	WEM-file	$\leftarrow$	WEMLoad
Outputs:	SoNode	$\rightarrow$	viewAll, So3DMarkerEditor

SoWEMRenderer2
This render module processes the geometrical data of the generated implant to be able to be visualized. Additionally, in the modules settings the diffuse color is set to titanium gold, leading to an implant representation according to it's real appearance.

Inputs:	WEM-file	$\leftarrow$	ImplantGen
Outputs:	SoNode	$\rightarrow$	viewAll

5.4.3 The So3DMarkerEditor Module

This module enables to add, select and edit markers. Therefore it needs a surface, serving as an input on which those markers can be placed on. Otherwise it would be difficult for the user to set a marker on the desired location,


Panel So3DMarkerEditor1

General | Appearance | Editing

Draw Mode:	Geometry
Num Markers Threshold For Fast Rendering:	5000
Scale:	1
Length:	1
Line Style:	Solid
Vector Draw Mode:	Vector No Arrows
Marker Type:	3

Panel So3DMarkerEditor1

General | Appearance | Editing

Color

Global Color	🟩
Specular Color:	⬛
Ambient Color:	⬛
Color Mode	Type

Text

Mode:	Off
Color Mode:	Text
Color:	
✓ Shadow	
Background:	⬛
Font:	Text2::HELVETICA 10

Panel So3DMarkerEditor1

General | Appearance | Editing

- ✓ Vector edit mode
- ✓ Draw markers
- ✓ Enable editing
- ✓ Enable selection
- ☐ Delete on click
- ✓ Add on click
- ☐ Use Managed Interaction mode
- ✓ Edit 3D
- ☐ Edit 3D vector

Delete All Markers

Figure 32: Shows the panel tabs of the So3DMarkerEditor. In the left tab it is possible to change the markers geometries like size and draw mode. The second (middle) one, enables appearance adjustments like changing the color. The right tab defines how markers are able to be edited like selecting, deleting, adding and so on.

since a three dimensional space is represented in two dimensions (screen). Linking with the *viewAll* module is mandatory for all of those modules to allow the user to set a marker based on the visualizing viewer panel. Figure 32 shows the panel tabs allowing to change appearances like size, color, draw mode and so on. Also, it is possible to change the editing behavior in the last tab.

So3DMarkerEditor1

This marker editor module handles the markers resulting from the baseline calculation. The settings are permanent and should not be altered by the users. Figure 32 shows the set up for this module where all options are set to default except those mentioned in the following lines. In the *General* tab only the scale is altered to a value of one, resulting in a better looking representation of the baseline. Setting the global color to green in the *Appearance* tab leads to a better distinction of baseline and other objects. Most important to mention is the *Editing* tab, since in default mode all options are unchecked. Following options have been chosen:

- **Vector edit mode** - enables editing of each marker by showing the normalized vectors of the main axis
- **Draw Markers**
- **Enable editing**
- **Enable selection**

- **Add on click** - this option is not necessary since the add on click option is used by the other marker editor module
- **Edit 3D** - for enabling editing in three dimensions

Inputs:	XMarkerList	←	CurvatureCalc
Outputs:	SoNode	→	viewAll

So3DMarkerEditor
This module handles the initially set marker, representing the center of the implant. Therefore, the surface structure on which the marker should be placed needs to be linked as well as the visualization panel to catch the user's interaction. The stored marker is further provided to the *CurvatureCalc* module. Since I did not want this single point to be shown in the user interface, the scaling was set to zero in the *General* tab. More, the color in the *Appearance* tab was set to default since it is not visualized anyway. In the *Editing* tab the following options have been chosen:

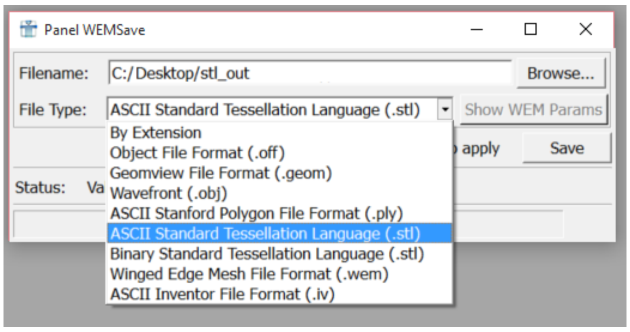
- **Vector edit mode** - enables editing of each marker by showing the normalized vectors of the main axis
- **Draw Markers**
- **Enable editing**
- **Enable selection**
- **Add on click** - this option is not necessary since the add on click option is used by the other marker editor module

All other settings where set to default.

Inputs:	SoNode	←	SoWEMRenderer1
Outputs:	SoNode	→	viewAll
	XMarker(List)	→	CurvatureCalc

5.4.4 The WEMSave Module

This module saves the generated implant to a given path in the given file format. A list of options is available where the ASCII Standart Tessellation Language with the ending *.stl* is the recommended one for further use with 3D print technology. Figure 33 shows the module's panel including the list of file types.


Panel WEMSave

Filename: C:/Desktop/stl_out

File Type: ASCII Standard Tessellation Language (.stl)

Browse...
Show WEM Params
By Extension
Object File Format (.off)
Geomview File Format (.geom)
Wavefront (.obj)
ASCII Stanford Polygon File Format (.ply)
ASCII Standard Tessellation Language (.stl)
Binary Standard Tessellation Language (.stl)
Winged Edge Mesh File Format (.wem)
ASCII Inventor File Format (.iv)
apply
Save
Status:

Figure 33: Shows the panel of the WEMSave module. The path of the *Filename* field gives the location where the resulting implant should be stored using the file type defined by the *File Type* field. A list of available options is shown, anyway the ASCII Standart Tessellation Language (.stl) is preferably used.

The *Filename* field is linked to the user interface with the field named *Savefile* thus enabling to change the saving path without accessing the internal network. Further, the *Save* button is linked with the user interface's *Export* button.

Inputs:	WEM-file	←	ImplantGen
Outputs:	None		

5.4.5 The CurvatureCalc Module

The module shown in Figure 34 is implemented by myself using C++. For a detailed description I recommend to have a look at Section 5.5 *Baseline Calculation*. It uses the initially set marker together with the surface geometry of the patient's skull to calculate, based on the input marker which represents the center, the baseline already including the skull's surface curvature. This is necessary to perfectly align the resulting implant on the patient's skull curvature for a perfect fit of the implant. The resulting baseline is represented as an array of markers, according to the chosen implant model. The additional output marker list stores the normal vectors for each corresponding marker of the baseline, used for further calculations in the *ImplantGen* module.

This thesis provides three different types of implants, possible to observe in Section 3.5, where the *CurvatureCalc* module adjusts the calculation according to the chosen implant. The parameter *article*, as an integer number ranging from zero to two, represents this three available implant models. In the user interface, one can select the implant type by checking the according

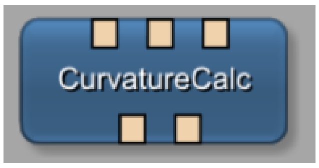


Figure 34: Shows the self implemented module named *CurvatureCalc* representing two inputs (left one for the initially set marker representing the center, right one for the surface geometry) and three outputs (left is for cutting plane visualization used for debugging, middle for the markers representing the baseline and right for the normal vectors of each baseline marker).

radio button. As also the module *ImplantGen* needs the information of the selected implant model, an additional connection of this parameter to the implant generating module is set up. Another connection is made between the mouse wheel value of the *SoMouseGrabber* module and the *RotationAngle* parameter of this module, thus enabling rotating the baseline by just turning the mouse wheel.

Inputs:	WEM-file	←	WEMLoad
	XMarker(List)	←	So3DMarkerEditor
Outputs:	XMarkerList	→	ImplantGen
		→	So3DMarkerEditor1
	XMarkerList	→	ImplantGen
Connections:	RotationAngle	←	Wheel (SoMouseGrabber)
	article	→	article (ImplantGen)

5.4.6 The ImplantGen Module

This module, shown in Figure 35, is also implemented by myself using C++. For a detailed description Section 5.6 *Implant Generation* provides further information. Based on the baseline markers and the corresponding normal vectors provided by *CurvatureCalc*, this module generates the final implants. According to the selected type of implant and its dimensions, the module first searches for the positions where the ring elements have to be placed to align them correctly with respect to the surface's shape. This is done for each ring element followed by generating the bridge sections according to Delaunay's triangulation theorem based on the curvature of the baseline and it's normal vectors to gain a well fitting implant which is finally visualized or saved by providing one single WEM output.

Since this thesis offers a selection of three different implant types, also this

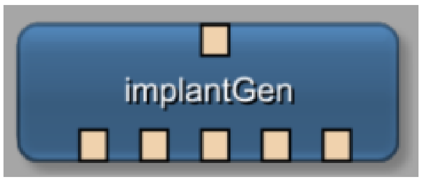


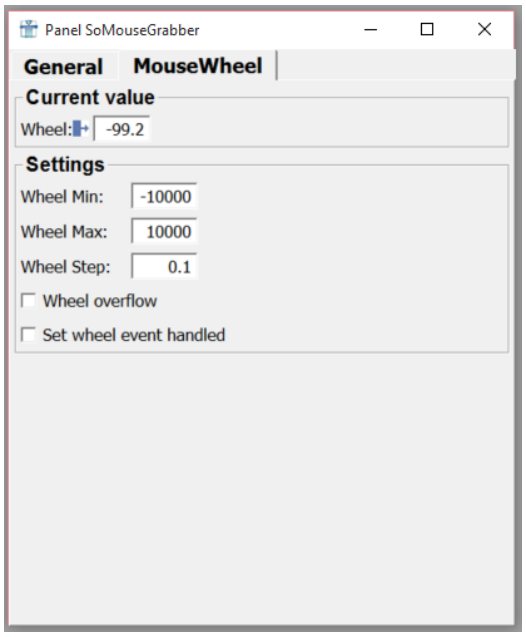
Figure 35: Shows the self implemented module named *ImplantGen* representing five inputs (first three ones provide the ring elements, fourth one provides the marker list representing the baseline and the most right one the corresponding normal vectors) and one output holding the generated implant in WEM-format.

module needs a parameter for the according selection. Therefore, a connection from the *CurvatureCalc* module's *article* parameter is established. Remember, that this parameter is linked to the user interface's radio buttons. Additional, it is the *ImplantGen* module providing an *On/Off* check box for switching on or off the visualization of the generated implant, thus increasing the user experience by decreasing latency times. This box is also linked to the user interface's button named *On/Off*.

Inputs:	XMarkerList (Baseline)	←	CurvatureCalc
	XMarkerList (normal vectors)	←	So3DMarkerEditor
	WEM-file (ring end element)	←	WEMLoad1
	WEM-file (ring middle element)	←	WEMLoad2
	WEM-file (ring angle element)	←	WEMLoad3
Outputs:	WEM-file	→	SoWEMRenderer2
		→	WEMSave
Connections:	article	←	article (CurvatureCalc)

5.4.7 The SoMouseGrabber Module

This module enables the user to gain a better experience of higher value since the module provides options for linking every interaction field of the computer mouse to a field in the MeVislab environment. I used this feature for linking the mouse wheel to the *CurvatureCalc's rotation angle*, which defines the direction where the baseline is aligned to. In other words, this link enables to rotate the implant around the normal vector of the implant's central point by only turning the mouse wheel. In Figure 36 the set up, like used in this thesis, is shown. As already mentioned the mouse wheel is linked to the *Wheel* field in the modules panel. Additionally, the *Wheel Step* field is further linked to the equally named field in the user interface resulting in



Current value

Wheel: -99.2

Settings

Wheel Min: -10000

Wheel Max: 10000

Wheel Step: 0.1

☐ Wheel overflow

☐ Set wheel event handled

Figure 36: Shows the panel of the SoMouseGrabber module with the tab for mouse wheel settings. In the *Wheel* field the current value for the mouse wheel position is shown. In the lower box the maximum and minimum values are defined together with the *Wheel Step*, defining the accuracy.

increased flexibility.

Inputs:	None		
Outputs:	SoNode	$\rightarrow$	viewAll

5.4.8 The SoOrientationInset and SoOrientationModel Modules

These two modules are used in combination with each other. By visualizing a model showing the axes' directions, on a desired location in the viewer panel, the user gains orientation support when rotating the viewers camera. The model can be chosen from a list where a cube (used in this thesis), a man, a skull, a torso or a skeleton is available. Further, the size and location, like upper-left corner are able to be chosen.

5.5 Baseline Calculation

5.5.1 Introduction

The baseline which is visualized as a very basic structure in form of markers, is calculated in the *CurvatureCalc* module of the network. The idea behind generating first a baseline instead of directly generating the final model, is to optimize calculation time resulting in a better user experience. Therefore the baseline is a simplified model of the final implant but already showing the most important characteristics like position, direction and curvature. Since the calculation of the baseline is less consuming in computational costs, the generation and adjustment of the baseline happens much faster than direct visualization. More, this method does not lack any characteristics a fully generated implant model would provide, as the baseline holds all the necessary information for optimal placement of the miniplates. In the following lines the idea behind the calculation process is described followed by a detailed explanation on how this idea was put into practice by implementing the *CurvatureCalc* module using the module wizard provided by MeVisLab.

First, I want to recall the calculation method of the ray-triangle intersection, according to Chapter 3.1.1 where the intersection location of a ray, cast from origin O in direction D, with a triangle is calculated. This method is used to determine the baseline where not only one ray but a cascade of parallel rays are cast, like shown in Figure 37. The object's surface, representing

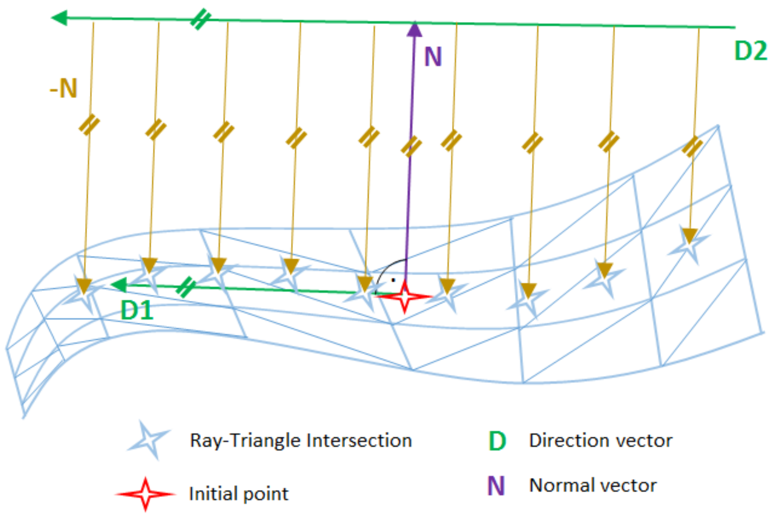


Figure 37: Illustration of the determination of the baseline. The direction-vector D (green) was set up parallel to the initial point's (red star) direction vector (green). The direction vector is normal to the initial point's normal vector (purple). The origins of the cast rays (brown) are translated along the direction vector and checked for intersection.

the patient's skull, is represented by the light blue triangulated mesh. The red star in the figure illustrates the initially set point as a starting value.

From this point, first the normal vector *N* (purple) according to the nearest triangle, is elicited. Next, the direction vector *D1* is determined by calculating by using the cross product of the normal vector *N* (which is the same for the implant and the baseline position) and the baselines direction vector *D*, which are not necessarily perpendicular to each other. The origin of *D2* is then calculated according to the implant model's measures, parallel to *D1*. Starting from this vector's origin, a cascade of rays is cast along vector *D2's* direction which means that the origins of the rays, used for ray-triangle intersection, are located along vector *D2*. Next step is to cast rays from the mentioned origins in direction of the negative normal vector -*N*. Where now, according to Chapter 3.1.1 the ray-triangle intersections are able to be located to set marker at those points (light blue stars), representing the final baseline. Important to mention is, that direction vector *D2* in this sketch describes the direction of the line along which the single rays, for ray-triangle intersection are cast, different from the direction vector *D* described initially for a single ray-triangle intersection, which describes the direction of the cast ray for intersection.

5.5.2 Input and Output Setup

Turning this idea into a module is done according to chapter 3.6, by using the module wizard from MeVisLab resulting in a C++ header and source file where only one output, a constructor, a destructor, the *handlenotification* method and an empty *process* method are defined. Since for generation of this module the option of a WEM-generator module was chosen, a single WEM output was already declared by the equally named class possible to be accessed by the *_outWEM* variable. The defined variables in the header file including their type can be observed in Table 5. Declared are two inputs, consisting of one WEM and one marker list connection, and three outputs, consisting of two marker lists and one WEM output. Further, WEMPtr defines a pointer to a WEM object and BaseField* a pointer to an in- or output field. Basefields for one WEM input (_inWEMFld) and one marker list input (_inPCAMarkerListFld, holding the initially set marker) are defined where a WEMPtr object (_inWEM) is defined to use the input WEM for internal operations. The ouput WEM (_outWEM) is defined by the WEM-gernerator class and not connected in the final version of this thesis since it just serves for debugging tasks. Two output marker lists are included, _outputXMarkerList (baseline markers) and _outputXMarkerListNorm (baseline normal vectors), where both need a XMarkerList for internal operations.

Type	Name	Description
BaseField*	_inPCAMarkerListFld;	marker input field
BaseField*	_inWEMFld	WEM input field
WEMPtr	_inWEM	Pointer to in-basefield
WEMPtr	_outWEM	WEM output
BaseField*	_outputXMarkerListFld;	baseline outputfield
XMarkerList	_outputXMarkerList	internal list for operations
Basefield*	_outputXMarkerListNormFld	marker output field
XMarkerList	_outputXMarkerListNorm	internal list for operations

Table 5: The table shows the declarations of the in- and outputs used for the CurvatureCalc module. WEMPtr defines a pointer to a WEM object and BaseField* a pointer to an in- or output field. Basefields for one WEM input (_inWEMFld) and one marker list input (_inPCAMarkerListFld) are defined where a WEMPtr object (_inWEM) is defined to use the input WEM for internal operations. The ouput WEM (_outWEM) is defined by the WEM-gernerator class. Two output marker lists are included, _outputXMarkerList (baseline markers) and _outputXMarkerListNorm (baseline normal vectors), where both need to be assigned to an XMarkerList for internal operations.

5.5.3 Algorithm

After setting up the module's inputs, outputs and parameters, the actual algorithm is implemented. An overview is given in the flow chart shown by Figure 38. **Inputs and Notification Handling**
Starting with the XMarkerList, holding the initially set marker, and the WEM, holding the patient's skull geometries, connected to the input, first the *handleNotification* method is applied. This method monitors weather one of the connected inputs or the defined module parameters have changed. If so, the algorithm continues in the *_process* method with a check for validity, otherwise nothing happens.

Validity Check
The validity check includes a verification of the connected components by checking for a permissible type of connection and an individual examination weather the WEM input holds at least one patch with at least one node, and if the marker list holds the initial marker, or not. If this survey is passed, pre-processing is applied.

Pre-Processing
First, this step includes the determination of the region of interest, which

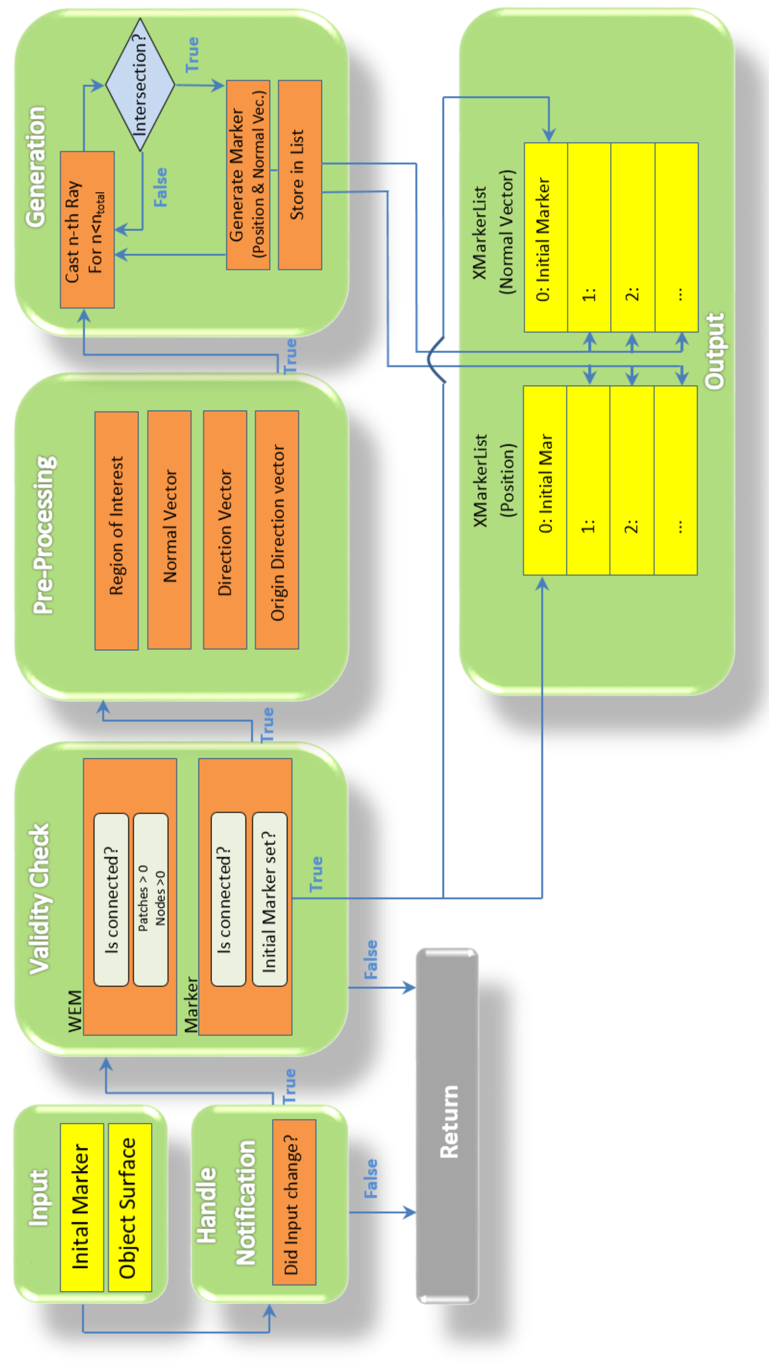


Figure 38: Flow chart of the baseline calculation algorithm. First, the input is checked for change with the handleNotification method. If so, the input WEM and marker list are checked for validity followed by a pre-processing step, if true. Finally, the baseline generation loop is applied which saves the markers (ray-triangle intersections) into a marker list together with the initial marker.

leads to runtime optimization during the generation loop, since the number of nodes to be checked are minimized within this step. Therefore, the distance between each face's centroid of the WEM object and the initial marker is checked for the distance to be smaller than a model dependent pre-defined value. If so, the face is stored in the *faces* vector used for further calculations. The distance is calculated using the L2-norm according to Formula 15, where d describes the Euclidean distance between M the initial marker's position and c_n, the centroids of the WEM-patche's faces:

$$d = \sqrt{\sum_{n=1}^{N}(M - c_n)^2} \tag{15}$$

The pre-processing also includes the calculation of the initial marker's normal vector. This is done by searching for the closest, in the WEM object stored, node, again by using the L2-norm shown in Equation 15. Since the nodes of the surface geometry already include a normal vector, the closest node's vector is used for the initial marker's one. The position of the initial set marker and it's normal vector are stored to position zero in the XMarkerLists. To *_outputXMarkerList* for the position in the 3D space and to*_outputXMarkerListNorm* for the corresponding normal vector.

Calculating the vector on which the single rays are cast from, is the next task of this intermediate step, where the direction and the origin have to be determined. Starting with the direction of this vector, which is computed by building the cross product of the initial set point's normal vector and the unit vector along the z axis, resulting in a vector, perpendicular to the normal vector of the initial marker thus describing the direction vector. For this purpose a macro is implemented computing the cross product shown in the following code example Listing 4.

```
#define CROSS(p, v1, v2)\
   p[0] = v1[1]*v2[2] - v1[2]*v2[1];\
   p[1] = v1[2]*v2[0] - v1[0]*v2[2];\
   p[2] = v1[0]*v2[1] - v1[1]*v2[0];
```

Listing 4: Macro for cross product calculation.

P denotes the outcome vector as a three dimensional vector using the three dimensional input vectors *v1* and *v2* of which the cross product is calculated. With this elements it is now possible to determine the origin of the ray casting ray in the code defined as *rayOrigin*. In order to do this, starting at the position of the initial marker, a value of 100 in direction of the normal vector is added, to further add an implant model dependent value in the negative direction of the direction vector. This value is built using the number of rays and their accuracy, which are also specified individually for each model type. Accuracy in this case is defined as the distance between each of the origins of the rays which are checked for intersection with the surface geometry. The value of this parameter is set to a value of 1 mm in standard settings. As a result the direction vector is translated along the normal vector and is cast perpendicular from it's origin to this normal vector.

Baseline Generation and Storage

With this set up it is possible to cast a defined number of rays (*numRay*), which are checked for intersection with the triangles, having their origins along the set up direction vector with defined distances (*accuracy*) between each other. The rays are cast one after another and first checked for intersection before casting the next one. The area, tested for intersection covers the full length of the implant including a few counters for certainty. For the ray-triangle intersection, each cast ray is tested against each face stored in the region of interest list. For the calculation of the intersection position, according to Section 3.1.1, all three nodes of the tested triangle as well as the origin of the current ray and it's direction (which is negative to the initial markers normal vector) are necessary. If the intersection is true, the exact position is calculated by determining the distance, along the ray, from the origin to the intersection point. Finally, the marker with the position of the intersection is stored in the marker list *_outputXMarkerList* and the current faces normal vector to the marker list *_outputXMarkerListNorm*. Intersection is tested with all triangles in the region of interest by each ray. If more than one triangle is intersected by the same ray, the one with the closest distance to the origin is stored, since the other ones are part of some not visible, inner surface geometries rather than outer ones. After one ray is tested against all triangles, the following one is checked for intersection until the predefined number of rays is reached. Since in this calculation the vector operations of building the cross and dot product as well as the substitution of two vectors is used frequently, those three operations are implemented as a macro, shown in listing 4 and 5, to improve performance.

```
#define DOT(v1,v2) (v1[0]*v2[0]+v1[1]*v2[1]+v1[2]*v2[2])
#define SUB(r, v1, v2)\
    r[0] = v1[0] - v2[0];\
    r[1] = v1[1] - v2[1];\
    r[2] = v1[2] - v2[2];
```

Listing 5: Macros for dot product and substitution calculation

The dot product does not use a variable for storing the result since the return value of the macro is the calculated result. On the other hand, the substitution calculation needs a variable r as a three dimensional vector where the result is stored to.

5.6 Implant Generation

5.6.1 Introduction

The task of generating the final implant(s) is processed in the *ImplantGen* module. Using five inputs, the baseline, it's normal vectors and the three different ring elements of the miniplate, this module generates the final output implant(s) in a piecewise manner where first two ring sections are brought into position followed by building the connecting bridge rather than building the first bridge element after the first ring element. The generation process is executed with respect to the baseline and it's normal vectors, where the implant model specific distance from the center (initial marker position) along the baseline is calculated for exact placement of the ring element. This point serves then as starting point where the first ring is set to. Following ring sections are placed again along the baseline but in opposing direction where the previous set element is used as starting point for distance calculation. Correct alignment of orientation and direction of each section is ensured by using the normal vector marker list, where the ring element's normal vector N is adjusted to align the baseline's normal vector$N$ at the position of placement. Additionally, the direction vectors D of both, the ring element and the current baseline position are aligned. The direction vector of the baseline shows the direction to the next marker. The implant's direction vector describes the connecting end of the ring. Figure 39 illustrates this processes by showing the baseline (green) on the patient's skull, together with a ring element and described vectors N and D.

5.6.2 Input and Output Setup

Again, MeVisLab's module wizard was an appropriate choice to set up a module of the type WEM processor, resulting in a C++ header and source file, where one WEM input, accessible by the *_inWEM* variable, a WEM output, represented by the *_outWEM* variable, a constructor, a destructor, the *handleNotification* method and an empty method named *process* are defined. The *handleNotification* method is skipped in this thesis since implant

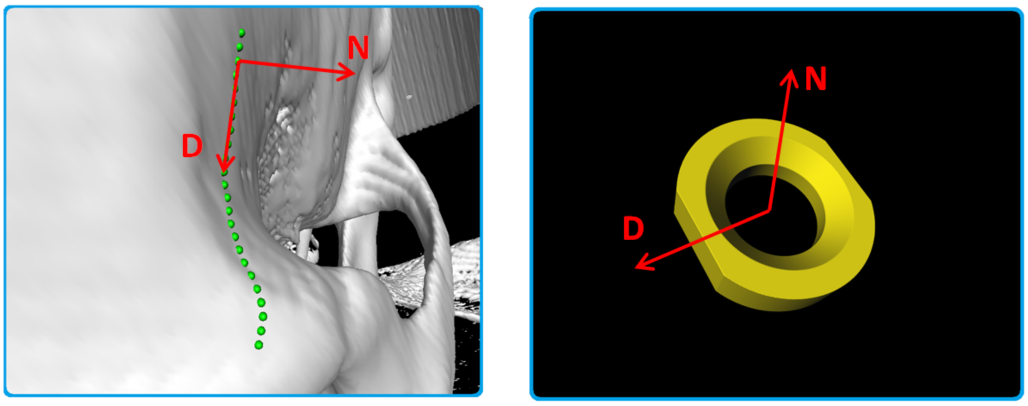


Figure 39: The patient's skull (white) with an applied baseline (green) are shown on the left side, where the normal vector N and the direction vector D are shown for the ring elements point of placement. The ring element itself is shown in the right figure, where the normal vector N and direction vector D (red), which have to be aligned with the vectors from the baseline, are illustrated.

generation should happen by enabling the corresponding button. As this module has to process the data provided by the baseline calculation module and also has to include the ring elements as a WEM file for generating the final implant, some additional inputs have to be declared in the header, shown in Table 6.

Summarized, the module uses two inputs of the type XMarkerList forwarded from the baseline calculation module, which are declared by the input basefields *_inMarkerFld* and *_inNormalsFld*, where first holds the marker positions in the three dimensional space representing the baseline and second the corresponding normal vectors. More, three inputs for the different ring elements (end, middle and angle parts) are necessary to be loaded into the module for implant generation. Consequently, to the already set up WEM input *_inWEM*, declared by the WEM processor class two input fields (*_inWEM2Fld* and *_inWEM3Fld*) are defined. *_inWEM3Fld* is not used in this thesis but serves for future software updates including a broader band of implant models. The variables defined as *WEMPtr* point to the appropriate WEM input basefield for internal computations. As a result, the final implant is stored in the *_outWEM* variable.

Type	Name	Description
BaseField*	_inMarkerFld	Baseline input field
BaseField*	_inNormalsFld	Normals input field
WEMPtr	_inWEM	Pointer to first input WEM
BaseField*	_inWEM2Fld	Second WEM input field
BaseField*	_inWEM3Fld	Third WEM input field
WEMPtr	_inWEM2	Pointer to second input WEM
WEMPtr	_inWEM3	Pointer to third inputWEM
WEMPtr	_outWEM	Pointer to ouput WEM

Table 6: The table shows the declarations of the in- and outputs used for the ImplantGen module. WEMPtr defines a pointer to a WEM object and BaseField* a pointer to an input field. _inWEM and _outWEM are declared by the WEM processor class. Additional inputs for processing the marker lists of the CurvatureCalc module are defined, _inMarkerFld and _inNormField as well as two more input fields for the additional ring elements, _inWEM2Fld and _inWEM3Fld with the corresponding pointer _inWEM2 and _inWEM3 for internal operation.

5.6.3 Algorithm

Finishing the set up of inputs, outputs and additional parameters the implementation of the algorithm is ready to be started. First, Figure 40 gives the reader an overview of how the implant is generated, followed by a detailed description of each part.

Input and Visualization Button

Important to know is, that by setting up a module of the type WEM processor, a WEM input as well as an output are already defined, where the object from the input is piped to the output. In other words, this means, that the input and the output WEM are the same where changes are made on the input WEM to gain the desired output. However, this characteristic is not desired in this work why it is necessary to add the lines shown in code example Listing 6 leading to a break of this pipeline.

```
_copyInputWEMsFld->setBoolValue(false);
_useInputWEMToCreateOutputWEMFld->setBoolValue(false);
```

Listing 6: C++ Code for unpiped in- output.

As already mentioned the *handleNotification* method is skipped since the generation of the implant depends on the status of the *On/Off* button

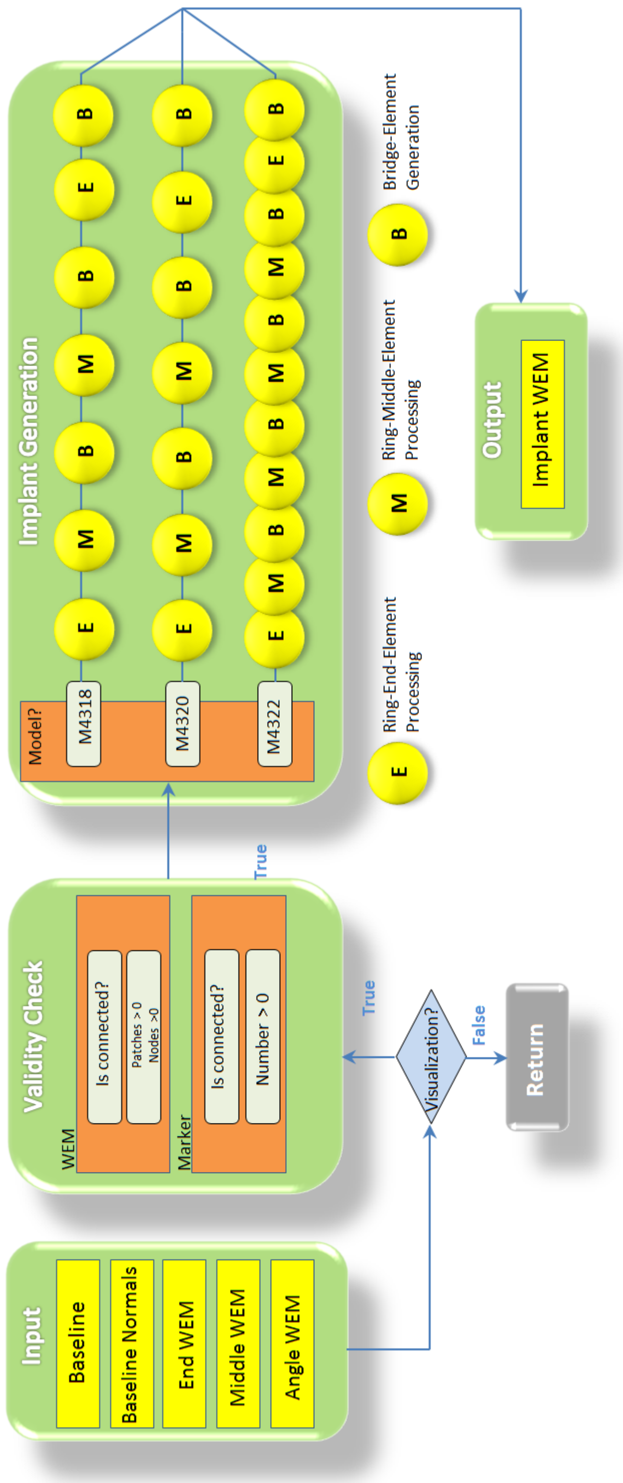


Figure 40: Flow chart of the implant generation algorithm. First, check of active visualization button is performed before piping the input to the validity check. This method checks the validity of the input WEMs and marker lists. If valid inputs are confirmed, the implant is generated depending on the chosen model by positioning end and middle ring elements linked by generated bridge sections. As a result the final implant is stored in the output WEM.

(*_OnOffFld*) from the user interface. In inactive mode, the algorithm is not applied by removing all patches of the *_outWEM*. Otherwise, in active state, the *_generator()* method is called starting with a check for valid input data.

Validity Check
First, the check includes the verification of connected components, thus all inputs have to be connected. Second, the WEM connections are only valid, if they hold just one patch with more than one node, since the application is not designed for files with multiple patches. Finally, the marker list inputs are checked for holding any markers, since calculations with empty lists do not lead to satisfying results. If valid inputs are confirmed, the following section generates the implant, while corrupted input data leads to cancellation of the generating process.

Implant Generation
Since three different implant models are possible to be chosen, first the user's selection is determined with the *article* variable. The name article is based on the fact that the MeVisLab catalogue defines the different models by article numbers. According to the model, the implant is generated sequentially like shown in Figure 40 in the *Implant Generation* block. It is important to keep in mind, that before the first bridge element is generated, two ring sections are positioned. Also, each placement of a ring section happens equally to the others, by first determining the position where the element has to be places, followed by translating all nodes to the correct position and finally rotating the ring accordingly for correct orientation, described in detail in the following lines.

First, the ring element's position on the baseline is determined by using the **_position** method. To work properly, the implant model (*article)* and the current ring element (*type*) are forwarded to the method. The current ring element is described by integer numbers starting by zero. With this information, the algorithm selects the model specific measurements on which the further calculation is based on. Starting at the position of the initially set marker, using the known distance from this central point to the point where the first element has to be placed, the algorithm runs from marker to marker, in one direction, along the baseline summing up the distances until reaching the final position. In most cases, this position is located between two markers. For exact placement, the direction vector between those markers is stored to add the missing length along this vector. Thus allowing to determine the exact position. For a better understand, Figure 41 illustrates an example of the described position determining process. Finally the direction

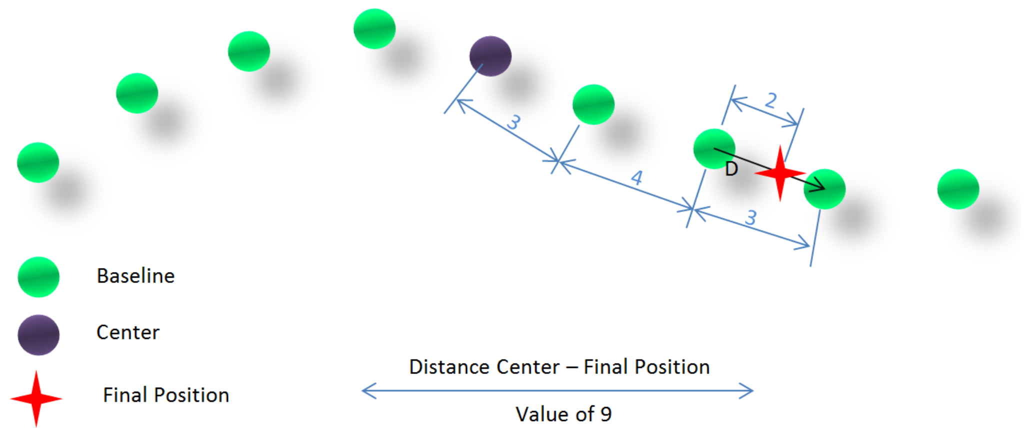


Figure 41: This Example shows how the ring element is placed exactly. The specific distance from the center (purple point) to the center of the ring on a straight line is given by a value of 9 in this example. What the algorithm does now, is to sum up the values along the baseline (green points) in one direction until gaining a Value higher or equal to the desired distance, in this case a value of 10 is achieved. Since the final point (red star) is located in most of the cases between two markers, the direction vector D (black arrow) between those two markers is used to add the correct value of two to the before added marker along the correct direction. Resulting in a properly determined distance of 9 along the baseline for this example

vector, the final marker position and the reference point are used for further calculation enabled by forwarding these variables by reference, to the method.

As with this step the element's final position is known, translation is performed by accessing each node of the WEM object's first patch, representing the ring element, and translating it by exactly this position's value, resulting in the correct position, since initially the object centres are located at the coordinate's origin thus enabling an easy calculation.

However, the implant is not aligned perfectly to the surface yet, since only positioning has been performed but the implant remained in its initial orientation with the normal vector aligned to the y axis. Therefore, the **_rotation** method is accessed where the center of the implant, the reference position, the current WEM patch, the direction along the baseline form the implants center and the implants normal vector are forwarded. In this method, first the implant's normal and direction vector are defined, which always point in the same directions, y-axes for the normal and -z for the direction vector, with a value of one. Together with the normal and direction vector form the center of the determined position, all vectors used for this calculation, like shown in Figure 39, are defined. Next, using quaternion rotation, described in Chapter 3.1.2, the normal vectors are aligned resulting in an implant already placed perfectly on the surface, but still with the side for bridge connection not being adjusted correctly yet. Now it is important to understand, that the normal vector at the determined position is not necessarily perpendicular to it's direction vector of the baseline, but the implant's vectors are. To achieve a correct orientation, the idea of spanning a plane by each normal vector and the corresponding direction vector of the ring and the baseline position, and using theirs normal vector for alignment, is used. The planes normal vectors are calculated by determining the cross product of each normal and direction vector, thus obtaining the perpendicular, to the spanned planes normal, vectors, resulting by two vectors in the same plane, which is perpendicular two the previous ones. In the following step, the rotation to align these two plane normals is applied on the implant to determine the correct orientation. Since this thoughts are not easy to understand by describing this issue only in words, Figure 42 illustrates this method. Knowing the correct rotation, the implant is rotated a second time. Due to this method the first rotation, which aligned the normal vectors, is not altered and the normal vectors still point in the same direction, perpendicular to the surface.

Finishing this tasks, which is equal to all ring sections, independent of the type and position, the element is now positioned and oriented perfectly with respect to the surface's shape and the chosen model type. However, the bridge sections are still missing, but described in the following section.

Having a look at the flow chart of Figure 40, again, one can observe, that before generating a bridge element, first two of the ring elements have to be generated which is equally implemented in the algorithm. Anyway, building these connections is identical by generating a connection consisting of at least two, triangulated blocks, connected to each other by their four corners. Thus, the diameter shows a rectangular shape: First, the method **_bridge** is called and attributes describing the two ring elements, between which the bridge has to be built, are forwarded. It is now another method, named **_connection** determining the four corner positions of where the bridge element has to be connected to. Since the ring's geometries are known, it is sufficient to forward the rings center position as well as normal and direc-

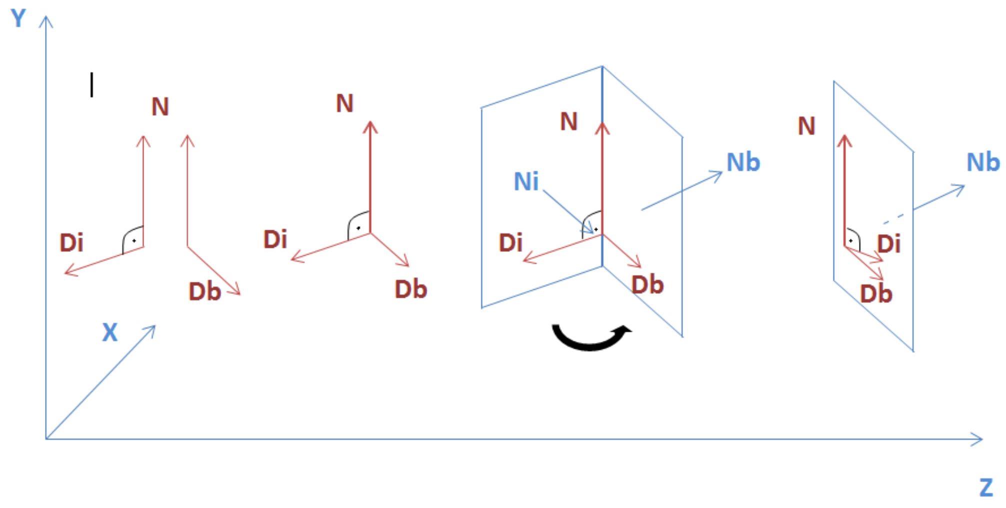


Figure 42: Method of orientation adjustment. The normal vectors N are already aligned. Di is the implant's and Db the baselines directions vector (left and middle-left), where the ring vectors are perpendicular which is not given for the baseline's vectors N and Db. Two planes and their normal vectors Ni and Nb are constructed each (middle-right), which appear in one plane as well, to be further at Nb's position (right). Resulting in Di and Db in the same plane, giving the correct direction, where Di is still perpendicular to N but adjusted according to Db.

tion vector each. The correct position of these nodes is found by distance measurements, using again the L2-norm, like described in Equation 15. By applying this method twice, the four corners are stored in two *WEMNode* arrays, *first* describing the array from where the bridge element is starting, and *last* describing the end of the generated element. Next, at each baseline point between those two rectangles, an additional rectangle is constructed based on the normal vector's direction. As a result the basis for finalizing the bridge element using triangulation is built, Figure 43 shows the current status. In a next step, the starting, ending and the corner positions of the rectangles are used for the triangulation task. To do so, first, a surface consisting of four nodes has to be defined. As an example one uses two neighbouring nodes from the starting rectangle (red crosses) and the opposing ones of the first constructed rectangle. In the implemented code, all those nodes are stored in different lists, thus allowing an easy separation of the different objects. Next, a triangle is defined between three of the chosen points to finish the task by

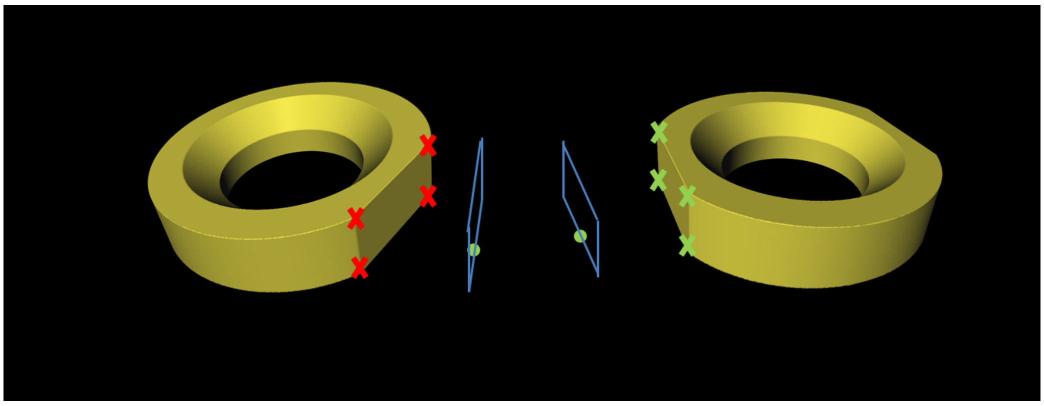


Figure 43: Basis for bridge building with triangulation. Red crosses mark the corner of the starting position, green crosses mark the corners of the ending position. To each baseline point (green points) a rectangle (light blue lines) is constructed using the normal vector, resulting in building the basis for generating a triangulated mesh with rectangle corners.

adding a face to this triangle. This is also done a second time with the other triangle, possible to be constructed with these four nodes. As a result, one gains rectangles, triangulated by two triangles. The step of generating the triangulated surface is illustrated in Figure 44 for an example on one surface part. This step is then repeated four times for each pair of rectangles until one rectangle is the one including the corners marking the end on the ring element. Finally, if the last bridge element is generated between the forelast and the last ring element, the generation of a perfectly aligned and orientated miniplate, depending on the surface geometries, the initial set point and the chosen implant model can be observed. However, the proposed software includes the additional feature of generating multiple implants in the same session, which's implementation is not of a trivial kind. Consequently, some explanation has to be done:

As the _**generator** method is finished the _**process** method continues where the handling of the output is now executed. This means in detail, if the first implant was generated, without saving any other before in this session, and the save button is disabled, only the current implant is forwarded to the output and the generation is finalized. In case, the *save* button returns a true value (_saveFld->getBoolValue == TRUE)), for the first time, the current generated implant is stored to the variable named _*finalImplants* which holds all the saved implants. Also in this case, a flag is set, remembering

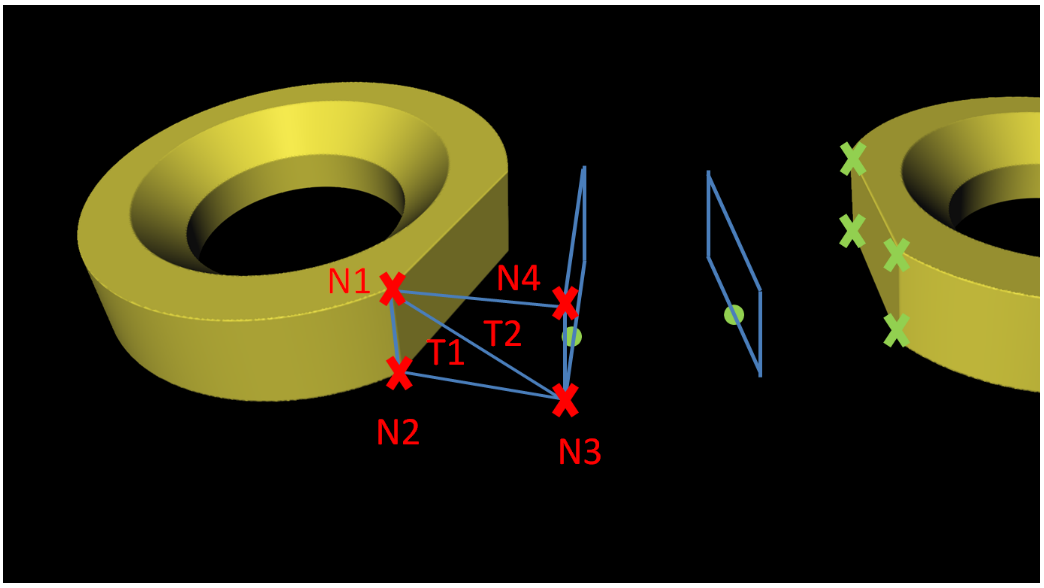


Figure 44: Example of generating the triangulation on a rectangle spanned by the two starting nodes (N1 and N2) and two opposing nodes on the first constructed rectangle (N3 and N4). Node N1, N3 and N4 build a triangle T2, and Nodes N1, N2 and N3 build the second triangle T1.

that at least one implant is already saved. In case of saving an implant to already existing ones, this is done by adding the current one to *_finalImplants*. Summarized, this means, that by saving the current implant, it is added to *_finalImplants* storing all previous saved implants, whereby the output of this module consists of only one WEM patch, including all saved implants together with the currently generated one.

5.7 User Interface

Even though the user interface is already mentioned various times in previous sections, in this one I give in this section a more detailed but still compact description of the interaction possibilities, which objects are included and how the implementation works. Therefore, the panel is initially shown and elucidated with all including fields, boxes and buttons together with an explanation on how those are linked to the network. Then, the MDL script, working in the background is explained.

5.7.1 Panel, Components and their Function

The full user interface is already shown in Figure 22 of Section 5.1 *Overview*. As one can see, the interface is separated in two main parts, the *Viewers* box, possible to observe in Figure 45, and the *Controls* box shown in Figure 46. Both assign an appropriate name to each of the included elements thus possible to be described properly in the following lines:

Viewer Box Visualizing the outcome, like the finally generated implant,

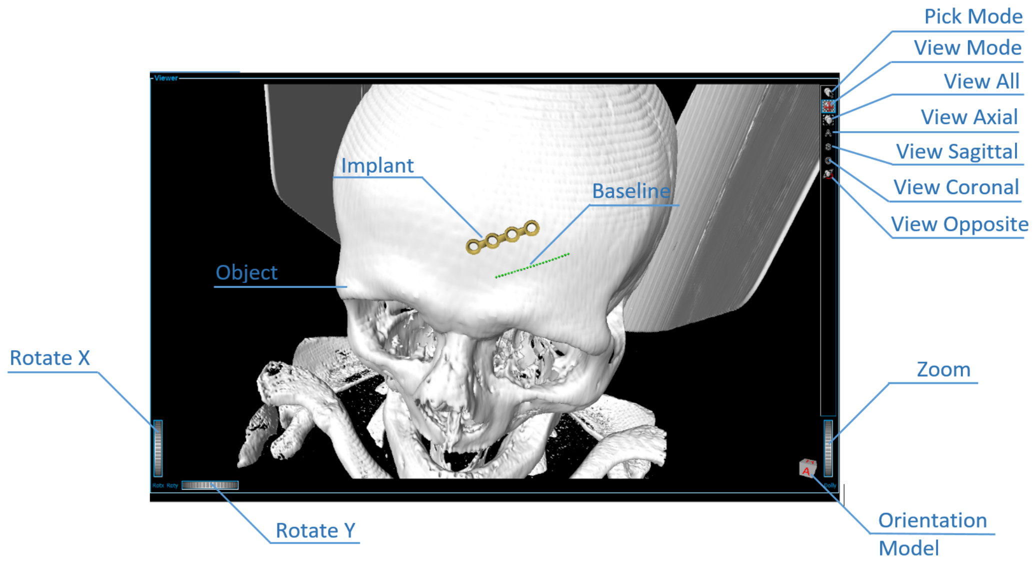


Figure 45: The user interface's Viewer box with name assignments to each element. This box visualizes the outcome and several intermediate data. It also includes interaction elements for a user friendly surface enabling flexible visualization commands.

and intermediate steps, like the baseline or the skull, is very crucial for a good user experience since this is the part where the result and it's relative position to the defect can be observed. Also, with this element it is possible to let the user perform interactions on the object's surface using the mouse pointer. Additionally, commands included as buttons and wheels further enhance the handling and flexibility. Following the elements are listed and briefly explained.

Object - Representing the patient's skull, this element is one of the most important ones since it represents the basic informations used for this tool.

Baseline - This geometrical objects is represented by a line of dots and serves as a guideline for the user by representing the implants orientation and curvature. Using the users first interaction by clicking on a surface location, marking the baseline's center with this step, this line is generated automatically. Depending on the selected implant model the length is adjusted automatically as well. By further turning the mouse wheel, the direction of the baseline may be changed.

Implant - The final outcome, based on the baseline's setup. The possibility of generating and visualizing more than one implant is given.

Orientation Model - Fixed to the lower-right corner, this variable model gives the user a orientation of the object by providing visual guidance of the current camera position, since it is possible to lose orientation especially when using a high zoom.

Pick Mode - Alternatively this mode is able to be selected by pressing and holding the *ALT* key, thus allowing to set a marker by clicking on a surface indicated by a cross shaped cursor. If inactive, *View Mode* is active.

View Mode - If active, the cursor appears shaped as a hand thus allowing to rotate the geometries in any desired direction.

View All - Sets the frame of the visualized field in a way all objects are included using the camera's current line of sight.

View Axial - Sets the camera to view the axial side.

View Sagittal - Sets the camera to view the sagittal side.

View Coronal - Sets the camera to view the coronal side.

View Opposite - This option changes the camera position to the opposing one, thus visualizing the object from the other side.

Rotate X - By turning this wheel, using the mouse's cursor, the camera is rotated around the x-axes.

Rotate Y - By turning this wheel, using the mouse's cursor, the camera is rotated around the y-axes.

Zoom - This wheel allows to zoom in and out along the camera's current line of sight.

Further, this panel provides user interactions using the mouse cursor. Therefore, one has to distinguish between the *Pick Mode* and *View Mode* like described above. First, by clicking on the skull's surface a user can set the initial point. By clicking an additional time, the before set point is removed and the new position is marked as center for the baseline. This change of position can be executed any time thus changing the position of all dependent geometries which means in specific, that this interaction alters the position of the initially set point, the baseline (if at least one marker was set before) and the generated implant (if already visualized). However, implants which are saved by using the *save* button are not altered in their position. In *View Mode* the mouse cursor is able to rotate the visualized geometries in any direction by clicking and holding the left mouse button followed by sliding the mouse in the desired direction. Also the rotation and zooming wheels are accessed with the click and slide operation.

All the mentioned elements including the tasks involving the mouse are already implemented by using the *So3DExamineViewer* which only has to be linked with the user interface via the scripting file.

Controls Box Including the *Controls* box enables the user interface to gain it's full power by providing all control elements necessary for planning a facial surgery using miniplates. Following, the single elements are explained according to the flow chart shown in the overview Section 5.1 in Figure 23.

Dataset Location - First thing to do, is to load the patient's dataset into the system, by clicking on the *browse* button next to the *Loadfile* field, named *Dataset Location Field*. Another window opens where the path to the patient's CT-scan has to be inserted.

Model Selection - Second, the desired implant model has to be selected by marking one of the proposed figures' radio button. The model's number, according to the MedArtis product catalog, is located to the right. This thesis includes the three most used implants in facial surgeries.

Wheel Accuracy - With this field, the accuracy of the mouse wheel is able to be adjusted, thus enabling precise orientation adjustment of the baseline and the implant. In other words, the user experience is increased by having an adjustable accuracy. By using a high starting value the implant can be placed roughly and fast. By further diminishing the

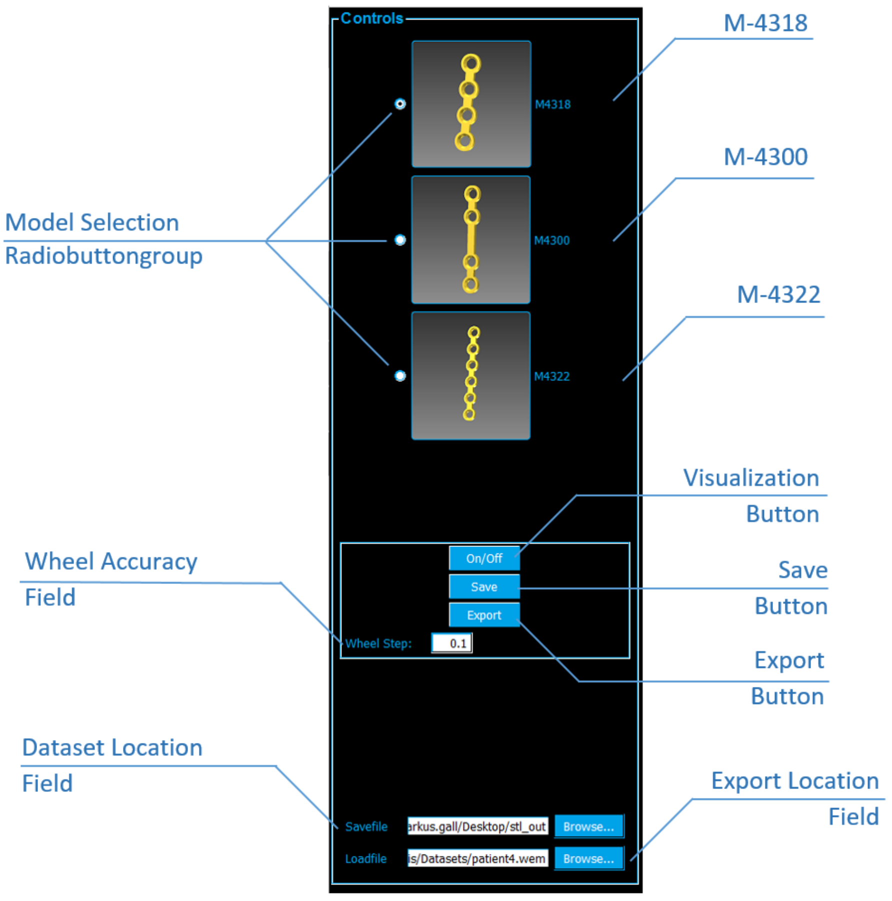


Figure 46: The user interface's Controls box with name assignments to each element. This box holds all the important elements for calculating, generating, saving and exporting the desired implant.

mouse wheel value the implant is able to be placed very accurate using the desired orientation.

Visualization - After the user is satisfied with the placement and the orientation of the implant's baseline, he can visualize the miniplate by clicking the *On/Off* button, also named *Visualization* button in the demonstrating figure. As a result, the final implant is generated and visualized in the *Viewer* box, using the chosen setup.

Save - The generated implant can still be altered in position, orientation and even in it's model type until the *Save* button is applied, resulting in saving the currently processed implant on it's position for the current session. Pressing this button, the current implant is not able to be altered anymore but a new one may be set up afterwards.

Export Location - To finally store the generated implant(s), first the location path of where the final outcome should be stored, has to be provided. This is done by clicking on the *browse* button next to the *Export Location Field* leading to open a new window where the exporting path is able to be inserted.

Export - The final step in the pre-planning process is to save the generated implants locally on the user's computer. By pressing the export button, all implants, including the currently processed one (which may be not be saved in the current session yet), are stored locally according to the *Savefile* field's path in STL-file format, thus enabling to use the outcome for further processing like 3D printing.

5.7.2 Scripting

This subsection gives the reader a detailed understanding on how the user interface's elements are generated and linked. Therefore, I give an example implementation including all important requirements and set ups followed by a description of the code used for the main user interface of the macro module.

Example Implementation of the On/Off Visualization Button

The *On/Off* button's function is already described in the previous Section 5.7.1 under *Controls Box* and is implemented in the *ImplantGen* module.

First, the desired button has to be defined in the module's header file. This is done by typing:

```
Boolfield *_OnOffFld;
```

The field is now defined as boolean variable, thus having just two states, an active and an inactive one. Next thing to do, is to assign a name and an initial value to the defined variable. This is done by entering the following line (line 53, mlImplantGen.cpp) in the module's constructor.

```
_OnOffFld ≡ addBool("On/Off", false);
```

where the method *addBool* sets up the defined variable *_OnOffFld* using the given name (red) and the value (blue). As a result, in the module's parameter panel one can now observe a new parameter named *On/Off* as a boolean value initially set to false.

The variable is now possible to be used in the further C++ code referred to as *_OnOffFld* and is also able to be accessed, by using the module's parameter panel. Additionally, the variable can be altered by clicking the according button in the user interface. To gain this function the script file of the macro module has to be adjusted correctly by defining a button and adding desired parameter options. Since this part's focus lies on giving an example of setting up a button, having a look at Listing 7 is sufficient for a general understanding.

```
1  Button implantGen.On/Off {
2          x = 0
3          y = 2
4          border = On
5          alignX = Center
6          title = On/Off
7        }
```

Listing 7: ON/OFF button script implementation.

The MDL code example shows how a button, for the usage in the user interface, is implemented. First, the object type has to be set up by defining the *Button* in line one. Since the variable type assigned to this element is boolean, the choice of a button is appropriate. In the same line, the linkage between the object and the module's parameter is set up (*implantGen.On/Off*). The x and y parameter define the alignment in the used grid where depending on the used variables x defines the position in horizontal direction and y the position in vertical direction. *Border* adds a border to the button object, for a better differentiation between background and object. *AlignX* describes the position in the defined grid block, where in this example the button is centred. The *title* parameter defines the title of the button, whereby the default value is the one of the linked parameter like used in the module's parameter list.

Many other options may be added to the button for altering the design, thus enabling a flexible design to fit the needs of the application. Even color schemes can be designed which are able to be selected by the end user.

Script File Structure

This part explains the structure of the script file used for generating the user interface. Therefore a simplified version of the code is shown in Listing 8 to get a better understanding.

```
Window {
 maximized = True
 style {
  colors {
    fg              = #00A2E8
    bg              = black
    button          = #00A2E8
    buttonText      = white
    light           = white
    midlight        = #00A2E8
    ...
  }
 }
horizontal {
    Box Controls {
    h = 900
    layout = Grid
    ...
    RadioButtonGroup CurvatureCalc.article{
      orientation = Vertical
      x = 0
      y = 0
      ...
      items {
        item 0 {image = $(LOCAL)/images/M4318.PNG}
        item 1 {...}
        item 2 {...}
      }
    }
    Box  {
     ...
      Button implantGen.On/Off {...}
      Button implantGen.save {...}
      Button WEMSave.apply {...}
      Field SoMouseGrabber.wheelStep {...}
    }
    Empty {...}
    Box  {...
      Field WEMSave.filename {
        title        = "Savefile"
41           browseMode = save
42           browseButton = ON
43           ...
44         }
45         Field WEMLoad.filename {...}
46       }
47     }
48     Box Viewer {
49       expandX = True
50       expandY = True
51       Viewer viewAll.self {}
52     }
53   }
54 }
```

Listing 8: Script structure of the user interface.

The code example shows a simplified version of the code generated in the script file. Thus, three dots represent object parameter settings already mentioned before, like the definition of colors, positions, expanding options or even the binding of a figure.

Since there are no inputs nor outputs used, the interface definition is skipped. That is why the example starts by defining the window in line one and adding the parameter *maximized = true*, thus expanding the user interface to full size when opened. The frequently used parameters for *color*, which uses a so called *style* environment, defines the colors for different elements by using the hex color code or pre defined words. In line 14 the horizontal alignment of the following elements is defined, forcing the boxes named *Controls* and *Viewer* to be aligned beside each other rather than being aligned to one's top or bottom. Boxes are able to be applied with a height h and a layout type. In this code, the layout type is set to grid, enabling to align each defined object to a position in the grid using the parameters x and y like in line 22 and 23. Also boxes are possible to be adjusted in color and other option indicated by three dots in line 19. To set up the implant model selection, a radio button group is defined and linked with the corresponding parameter *article* of the *CurvatureCalc* module. Additionally, the positioning on the grid position zero-zero is defined causing the group to be aligned in the upper left part. By defining further items, the options for selecting the implant models are generated, including an image for a better overview and design. The following lines define another box which includes the buttons for the user interaction. Again, the dots replace possible parameter setting like already used in upper parts. The conclusive box, holding the fields for path selection, is separated from the box above by an empty space. Closing the brake in line 48, enables to define the second, horizontally aligned *Viewer* box.

6 Results and Evaluation

The results presented in this section are accomplished on a system with an Intel©Core i7 CPU at 2.80 GHz and a memory (RAM) of 24.0 GB running on Windows 8.1 Enterprise 64-bit. Additional, the system is equipped with a NVIDIA GeForce GT 630 graphics card with 2GB VRAM.

Since the focus of this thesis lies in pre planning the positioning of miniplates for facial reconstructions, in the following, the results during the most important intermediate stages are shown. Starting by opening the network, consisting of only one macro module, the user interface expanses itself to full screen with the initial state as shown in Figure 47. Since no loading path

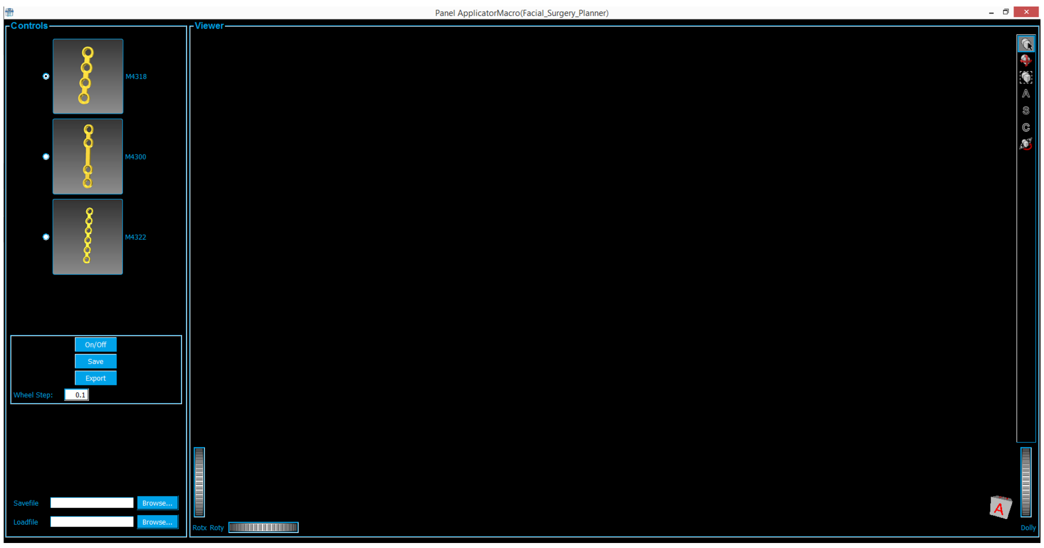
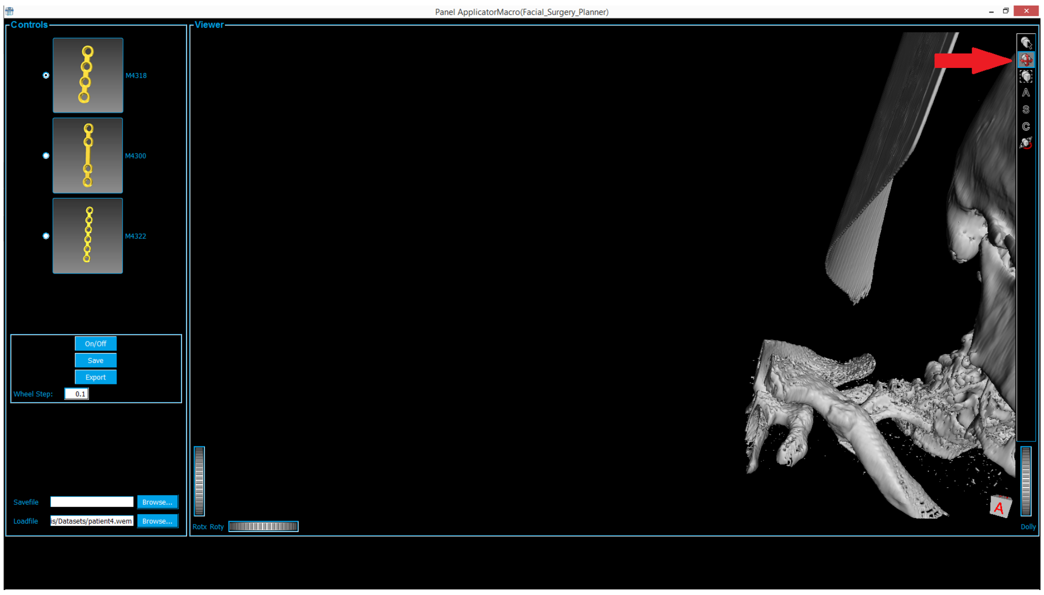


Figure 47: The figure shows the user interface at it's initial state where no data set has been selected, resulting in an empty viewer box.

is pointing to a dataset, the *Viewers* box is empty. The *Save*, *Visualization* and *Export* button are inactive and have not been activated yet. The *Wheel step* is initially set to a value of 0.1 and the selected miniplate model is set to M4318.

Next, a data set of the type WEM or STL is added by clicking on the browse button next to the *Loadfile* field, resulting in Figure 48. The figure shows the



Wheel Step: 0.1

Savefile

Loadfile s/Datasets/patient4.wem

Figure 48: The figure shows the user interface where a dataset has been selected already. Due to swiping over the Viewer box in view mode, the camera is translated. The red arrow indicates the selection of the view mode.

loaded data set which is translated by rotating the camera due to swiping over the box and holding the left mouse button. Initially the panel is set to *View mode*, indicated by the red arrow. To set the skull to the center of the box, the *View all* button has to be activated, thus adjusting the camera's field of view to focus on the visualized model. Figure 49 shows the result of this step where the red arrow points to the *View all* button.

It is now possible to rotate, zoom or change the camera position with the help of the wheels and buttons in the *Viewer* box to gain a perfect sight on the area of interest. If one has reached a satisfying field of view, next step is to set the baseline. Therefore, the mode is changed from *View* to *Pick* mode, by either pressing the *Pick mode* button, or toggling between those two modes by holding the *ALT* key. Now the initial point is selected by clicking on the skull's surface followed instantly by the calculation of the baseline, shown in Figure 50, where the green line indicates the baseline and the red arrow the button for activating *Pick mode*. By using the mouse wheel the baseline now is possible to be rotated to gain the desired orientation. Also by selecting another initial point, the center of the baseline is translated to the new position. When satisfaction is reached, the model is able to be generated by a click on the button named *On/Off*, thus toggling between active

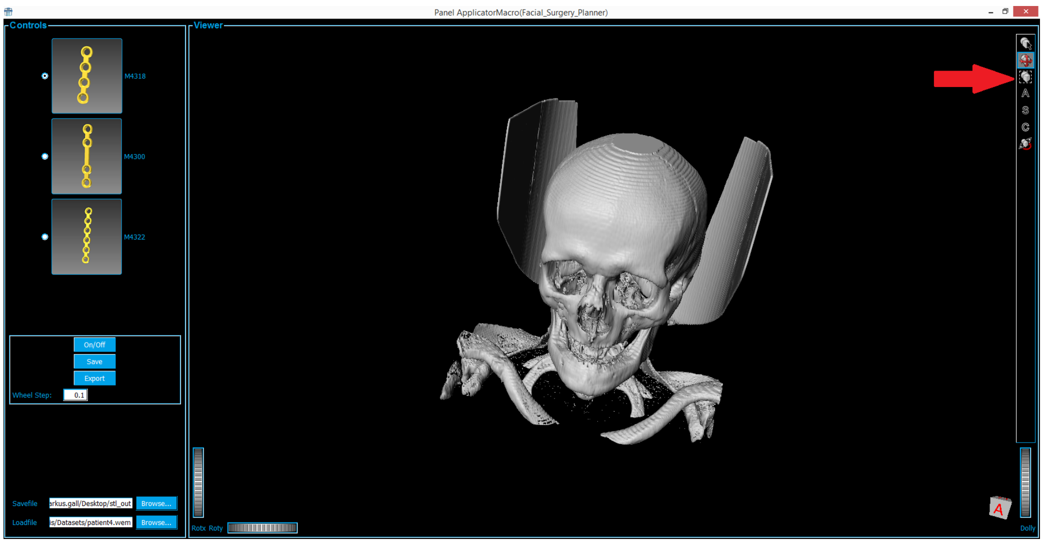


Figure 49: The figure shows the user interface where a dataset has been selected and already centred by clicking on the View all button, indicated by the red arrow.

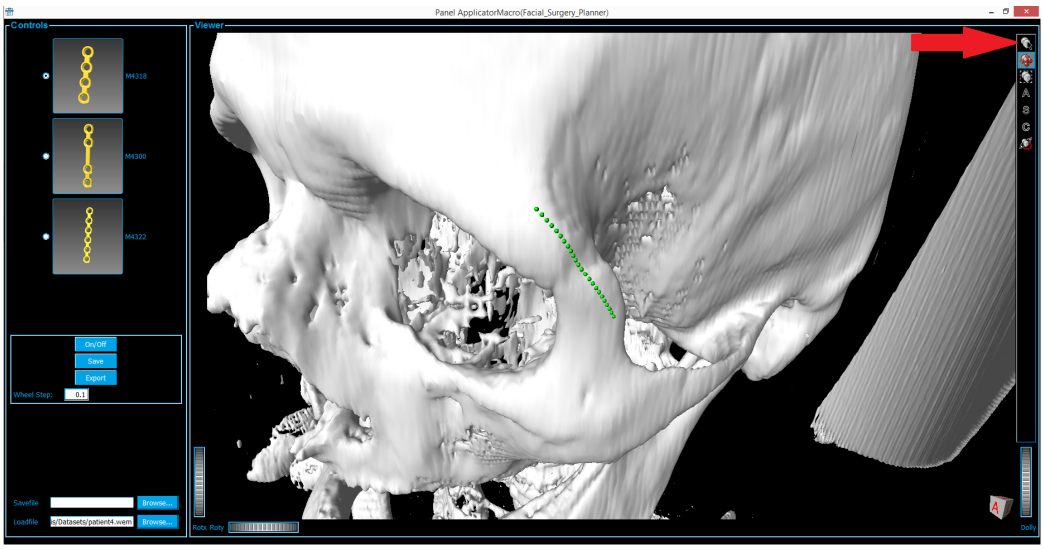


Figure 50: The figure shows the user interface where the initial point has been set and the baseline (green) has been calculated. The red arrow shows the button for active Pick mode.

and inactive visualization mode of the selected miniplate (active state is indicated by a grey background color). Figure 51 shows the outcome of this process. Important to mention is, that implant adjustments should be exe-

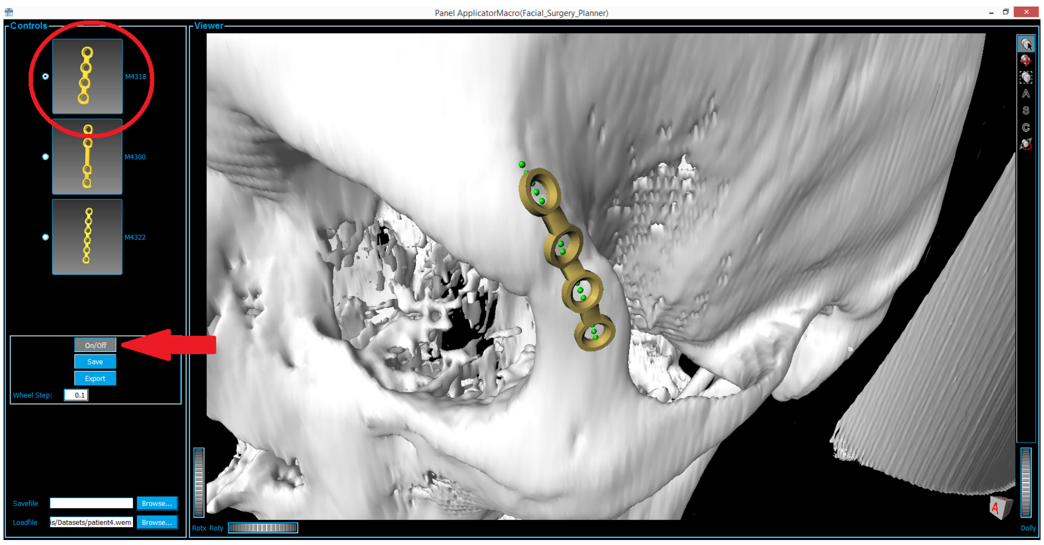


Figure 51: The figure shows the user interface where the miniplate model M4138 has been placed on the zygomatic upper bone with active visualization.

cuted only in inactive visualization mode (indicated by blue *On/Off* button), thus decreasing latency times and increasing the user experience. Anyway, adjustments of all kinds, are able to be performed at any time, until the implant is saved.

It is also possible to toggle between the implants, where, depending on the state of the visualization mode, the generated implant or the baseline are adjusted. Figure 52 shows the same user interface as shown in the previous figure, but changed the selected model from M4138 to model M4300. Observable is, that not only the generated implant changed, also the baseline adjusted it's length. To store the implant for this session, the *Save* button has to be activated, resulting in storing the generated implant in it's current state for this session. Thus, adjustments are not possible to be done any more, but a second implant may be generated by setting a new initial point. Figure 53 shows a second baseline, placed on the lower zygomatic bone, with a previously saved implant.

As already performed in a previous step, the implant is again generated by activating the visualization button leading to a second implant observable

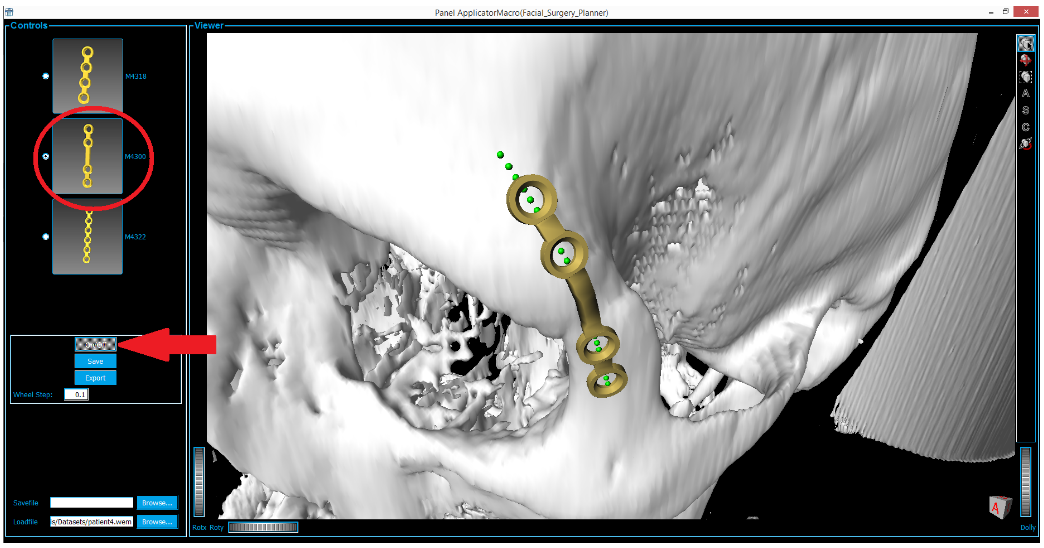


Figure 52: The figure shows the user interface where the miniplate model M4300 has been placed on the zygomatic upper bone with active visualization.

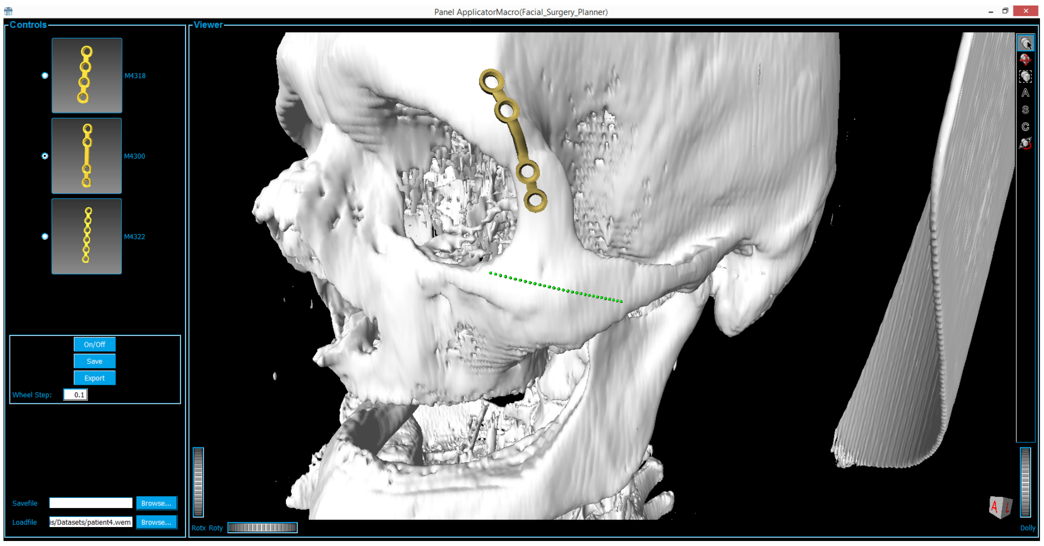
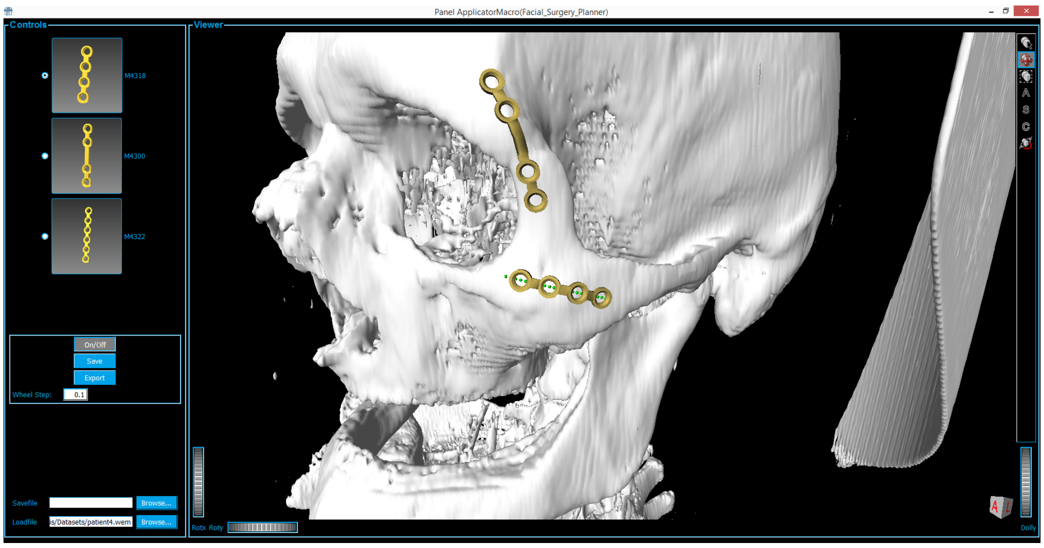


Figure 53: The figure shows the user interface where the saved miniplate model M4300 has been placed on the zygomatic upper bone and a baseline for a second implant generation is placed on the zygomatic lower bone.

in Figure 54, where the models M4138 and M4300 are visualized next to each other. To finish the generation process, the generated implants are able to



Wheel Step: 0.1

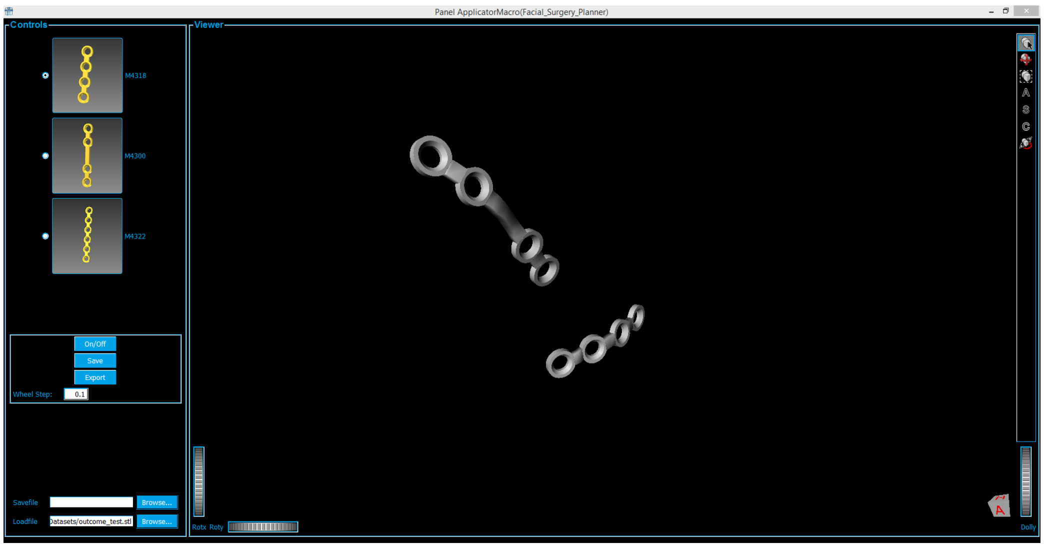
Savefile

Loadfile is/Datasets/patient4.wem

Figure 54: The figure shows the user interface where the saved miniplate model M4300 has been placed on the zygomatic upper bone and M4138 miniplate is placed on the zygomatic lower bone.

be exported, which means in other words to save them on the system's local drive. Therefore, the current implant does not have to be saved, since all implants visualized in the *Viewer* box are exported. Consequently, it is necessary to define a *Savefile* location, by clicking on the field's browse button. Followed by performing an activation of the *Export* button, the implants are saved on the local drive in STL file format. The exported STL file is possible to be viewed either by another network constructed in MeVisLab or by choosing it as a dataset. Even though the tool is not developed for visualizing the generated implants seperated from the skull, it is possible to do so for controlling purposes like shown in Figure 55. The exported implants are represented in gray rather then in their titanium color since in this case only for representation purposes, the implants are visualized as a dataset, where the diffuse color in the renderer module is set to grey. For rendering an implant, the renderer's diffuse color is set to titanium in the network. Generating additional implants is done in the same way as described by this example set up of two miniplates in the zygomatic bone area.

Before showing results of the baseline and the generated implants, Figure



Wheel Step: 0.1
Savefile
Loadfile Datasets/outcome_test.stl

Figure 55: The figure shows the user interface with exported miniplate model M4300 and M4138 without visualization of the skull or baseline.

56 shows the region of interest for the model M4138. All face centroids, included in the ROI, are marked with a green dot and the initial set point is represented by the purple cross. As a result, only faces inside the green doted region are utilized for the calculations.

Following results focus on the outcome, first on the baseline adoption, and second on the final miniplate outcome, rather than on the work flow of the generation by using the user interface. Hence, Figure 57 compares the baselines of the three different models. The centres of the model's different baselines are found at the same position, only the length between each of them varies depending on the model's dimensions.

Moreover, the baselines are designed to capture the curvature of the skull's surface independent of the surface form. Figure 58 shows, that the baseline does adjust itself even on the most difficult surfaces like a wave form or 90 degree angles.

In Figure 59 an example result containing all three available implants, each located on a position where they generally are applied (left). For better illustration, the single implants are extended portrayed on the right as well.

Additional, Figure 60 shows the same outcome from it's sagittal and coronal perspective. Beside the presented results, an evaluation has been performed to check the proposed software for weaknesses and to get ideas which

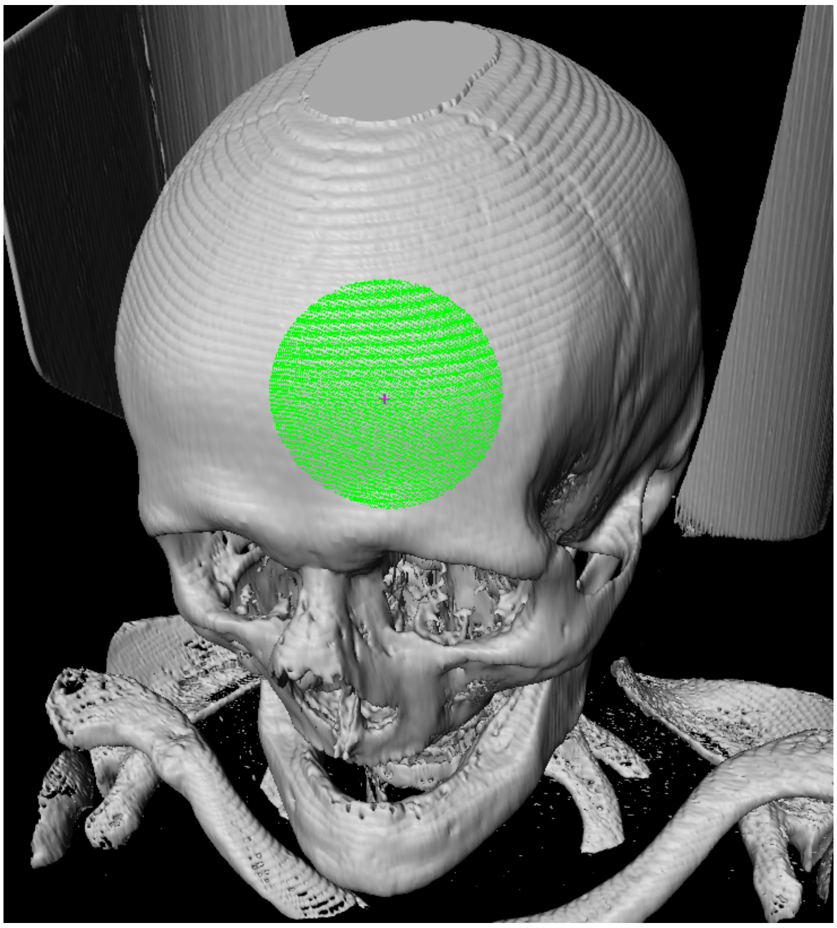


Figure 56: The figure shows the region of interest for the model M4138. All face centroids, included in the ROI, are marked with a green dot and the initial set point is represented by the purple cross.

characteristics are important to the end user who's expertise will generally be of a surgeon of the oral and maxillofacial area. Consequently, two surgeons, of the Oral and Maxillofacial department of MedUni Graz are used as subjects in addition with one PHD student and a master student from the Institute for Computer Graphics and Visions (ICG) who work on medical data, to get a rating from another perspective as well, thus determining weaknesses from the developing point of view. No one of them had any experience with the presented software so far, which was first introduced by showing them all functions and possibilities followed by an short example performed by my-self. Afterwards, it was their turn to fulfil a predefined exercise, euqally to all subjects and similar to a case occurring in the daily routine of a surgeon.

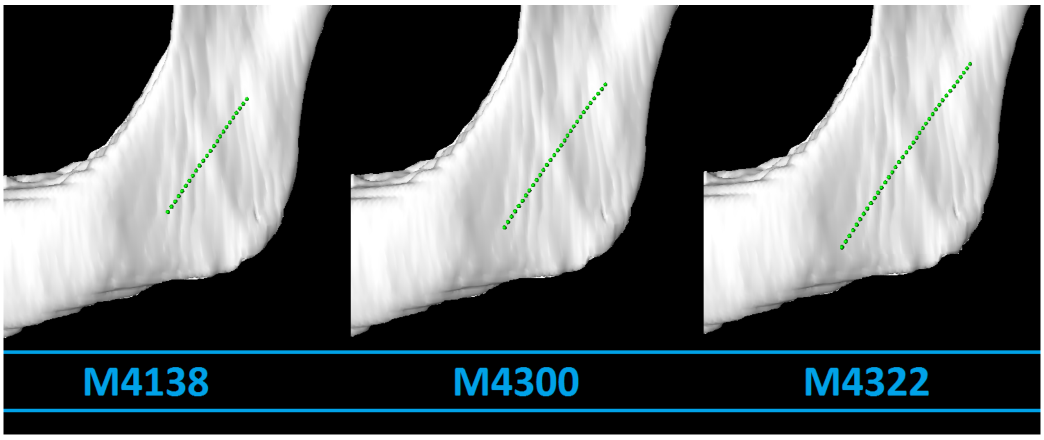


Figure 57: The figure shows the baseline for each of the three selectable models, where the center and the orientation of the line equal each other.

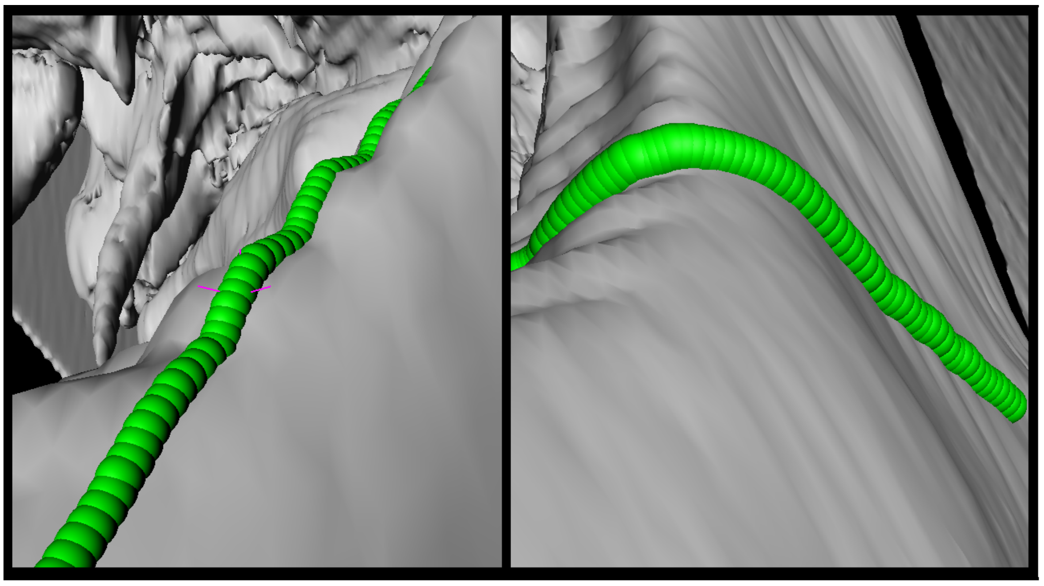


Figure 58: The figure shows the baseline (green) for difficult surfaces. On the left image a wave like surface is shown, and on the right side the baseline adjusts to a 90 degree angel.

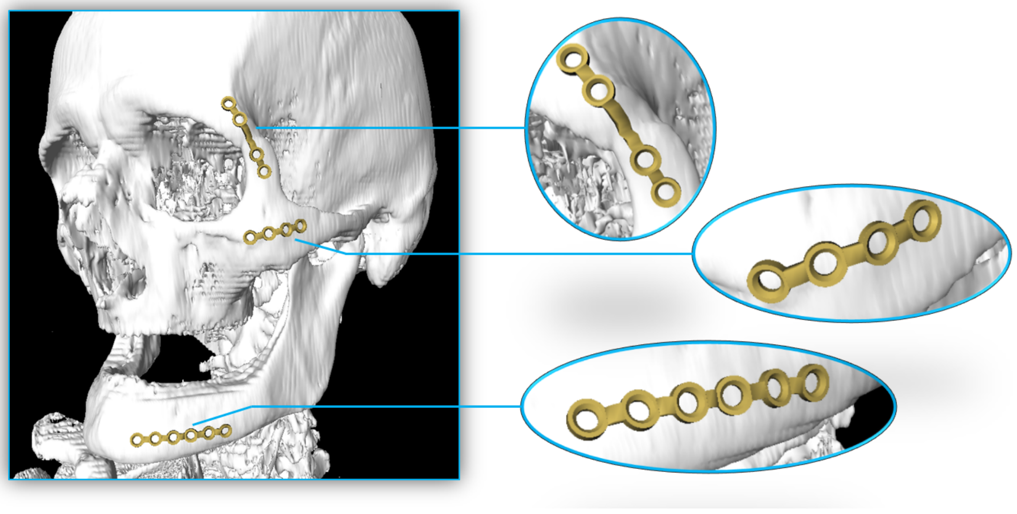


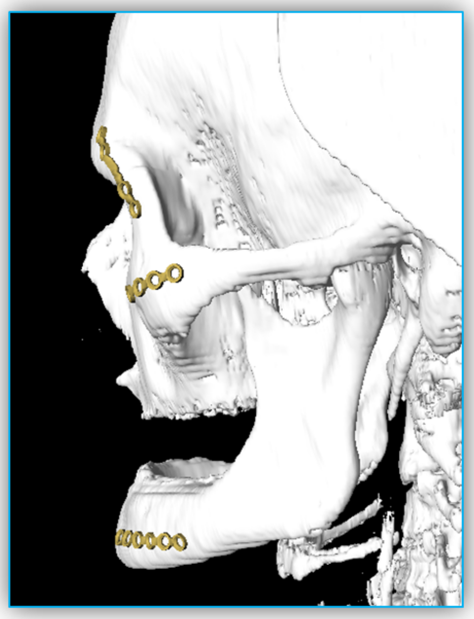
Figure 59: The Figure shows an example result containing all three available implants each located on a position where they generally are applied (left). For better illustration, the single implants are extended portrayed as well on the right.

During working on this exercise the time was measured to get an idea of how long it does take a person to fulfil this task. Subsequently, the testers are asked to answer a questionnaire containing 11 questions which were to answer on a six point Likert-scale but one (question 11) has to be answered with yes or no. The questions are listed in table 7 and are adopted from the software evaluation questionnaire of ISO Norm 9241/110 [2].

The exercise to be accomplished is given as:

Load the data set given for Patient 4 and perform the placement of a mandibular media fracture (imagine the fracture in the center) using at least two implants of different models and export the data to the desktop, if satisfaction is reached.

The result of this questionnaire is shown in Figure 61. The x axes represents the questions from one to eleven (Q1 - Q11) and the measured time T. The y axes represents the median score for Q1 to Q10. Also the median value for Q11 is shown but represents a yes or no answer, where yes equals a value of six points and no equals a value of zero points. The time is represented in minutes. To all data the superimposed standard error is represented by the white lines. In Table 8 the underlying data is represented, where subject two and three represent the surgeons' results. The table shows the resulting data



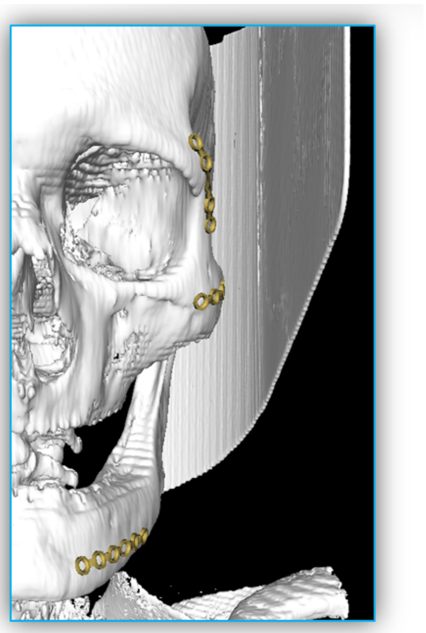


Figure 60: The Figure shows the result from Figure 59 from it's sagital (left) and coronal (right) perspective.

Nr.	Question
Q1:	The software does not need a lot of training time.
Q2:	The software is adjusted well to achieve a satisfying result.
Q3:	The software provides all necessary functions to achieve the goal.
Q4:	The software is not complicated to use.
Q5:	How satisfied are you with the UI surface? (arrangement, style, clarity)
Q6:	How accurate was the placement of the implant?
Q7:	How satisfied are you with the presented result?
Q8:	Was it easy to adjust the implant? (position, orientation, model,...)
Q9:	How satisfied have you been with the time consumed? (no training)
Q10:	How is your overall impression?
Q11:	Would you use the software in a daily routine? (assumed that the 3D printed implant would fit with just a few adjustments)

Table 7: The table shows the questions asked after fulfilling a predefined task.

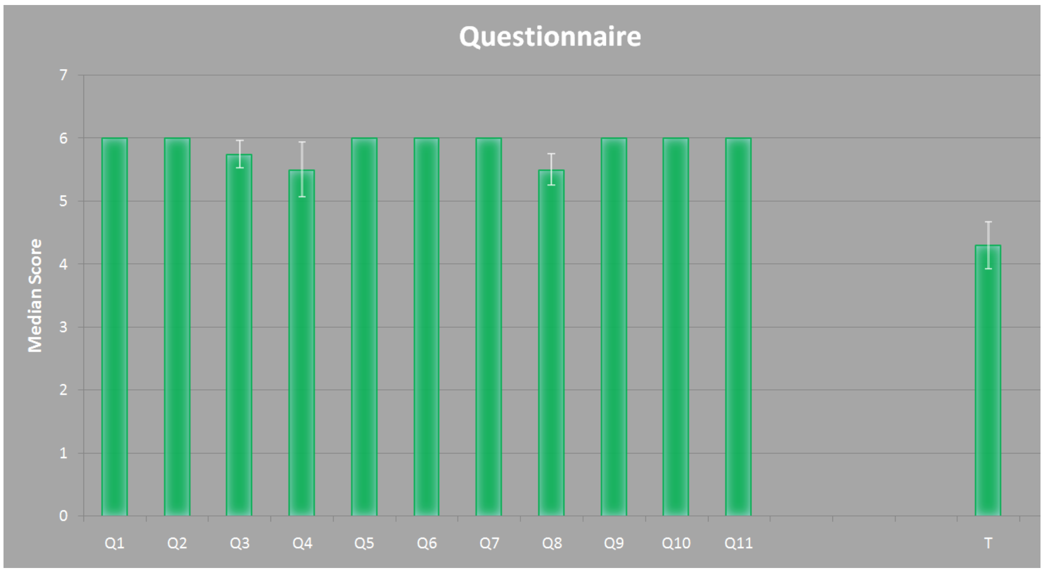


Figure 61: The x axes represents the questions from one to eleven (Q1 - Q11) and the measured time T. The y axes represents the median score for Q1 to Q10. Also the median value for Q11 is shown but represents a yes or no answer, where yes equals a value of six points and no equals a value of zero points. The time is represented in minutes. To all data the superimposed standard error is represented by the white lines.

from the evaluation questionnaire for each subject. Subject one and four do not have their expertise in the medical are, where subject two's and three's is. Median is the median value of the four subjects and error describes the superimposed standard error. Q1 to Q11 are possible to be answered with the use of a six point Likert-scale but Q11 is a yes or no question where no equals a value of zero and yes equals a value of 6. The time T is measured in minutes.

More, some initial results have already been presented and discussed as a late breaking research poster at the 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC) in Orlando (Florida, USA). However, at the EMBC I present only a one page summarized description of the methods together with a poster, possible to be observed in the Appendix section A. Additionally a paper at the SPIE medical imaging conference in Orlando (Florida, USA) was submitted followed by the idea of writing a journal article.

	Subject					
Nr.	1	2	3	4	Median	Error
Q1:	6	6	6	6	6,00	0
Q2:	6	6	6	6	6.00	0
Q3:	6	5	6	6	5.75	0.21
Q4:	4	6	6	6	5.50	0.43
Q5:	6	6	6	6	6.00	0
Q6:	6	6	6	6	6.00	0
Q7:	6	6	6	6	6.00	0.25
Q8:	5	6	6	5	5.50	0
Q9:	6	6	6	6	6.00	0
Q10:	6	6	6	6	6.00	0
Q11:	6	6	6	6	6.00	0
T:	4	5.5	3.5	4.2	4.00	4.3

Table 8: The table shows the resulting data from the evaluation questionnaire for each subject using a six point Likert rating. Median is the median value of the four subjects and error describes the superimposed standard error. Q11 is a yes no question where no equals a value of zero and yes equals a value of 6. The time T is measured in minutes.

7 Discussion and Future Outlook

In the proposed work, a software tool for surgery planning of facial reconstructions using miniplates was implemented to support surgeons in their daily routine. Therefore, with the medical imaging platform MeVisLab a modular network was constructed able to load a patient's CT data set and plan the positioning of the three most used miniplates. To fit the surface perfectly, it was necessary to generate two modules using C++ with the provided wizard for module generation.

First one was implemented to catch the surface curvature depending on an initially set point by calculating a baseline which serves as a simplified implant model, showing the user all necessary properties to adjust the implant properly. The calculation of this baseline is represented by a list of markers, where each marker describes the position of a ray-triangle intersection, where the triangles are those from the data set surface geometry and the rays are cast along a defined line, dependent on the initial set point and it's normal vector.

The second module uses the marker list, representing the baseline, another list holding the corresponding normal vectors to each baseline point and the pre-built ring sections as a WEM or STL object. With this inputs, the module is able to generate, depending on the selected implant model, the final outcome. To do so, the ring sections have to be aligned with the corresponding point on the baseline, possible to be determined by the model's dimensions. Next, the ring elements were oriented, by aligning the ring's and the baseline point's position normal and direction vectors. This was done with all ring sections equally. To gain a final implant, the bridge elements were generated as well by building them step by step, from one ring end to the other, by setting up a rectangular skeleton at each baseline point which are then triangulated, according to Delaunay, in the following step, thus building the final implant in STL format, possible to be stored on the local drive and also possible to be printed in 3D.

As described in the results section, the generating process is easy to handle and since just a few user interactions are mandatory, it is not of a big difficulty to remember these steps. Analyzing the evaluation results shows, that subject one was not perfectly satisfied with question four, asking if the software is complicated to use where the subject stated, that the use of the visualization button and an additional export button, made it feel complicated to him. Also, I recognized, that especially the surgeons had difficulties using the viewer panel first. However, this difficulties disappear after getting to know a visualization tool like this, since the handling is new, once comfortable with using it, it is an easy task.

More, the tool is very flexible, concerning implant position and number of implants possible to be applied, since there are no limitations from the coding part. However, it depends on the system used for running the proposed software, since the implants are generated with very high quality, the working memory reaches it's limitations fast. With the tested system of 24Gb RAM it is possible to generate up to four implants of the M4322 model, which is the most expensive one.

As also shown in the results section, a region of interest optimizer was implemented which is an easy way to decrease computational costs but does not influence the visualized result in any way, since the end user will observe the same outcome. The advantage lies in actually including only the faces of the ROI for calculations, thus reducing the data size significantly, resulting in faster computations but still being able to present the full skull to the user.

Having a closer look on the results of the baseline, one can clearly see, that it fits perfectly on any surface curvature. Anyway, it is necessary to choose a dataset and a position where the surface is closed, since the generation of the baseline will stop, if one ray is cast through a hole and intersecting with another surface, very distant from the other points. More, in Figure 58, the baseline on a 90 degree surface is shown. In a case like this it is important to set the center of the baseline on the edge, since baseline calculation is based on this point's normal vector. Otherwise, cast rays will not intersect at both sides (upper and lower from the center point) with the surface, as a result of parallel cast rays to the surface. Figure 62 illustrates this case. However,

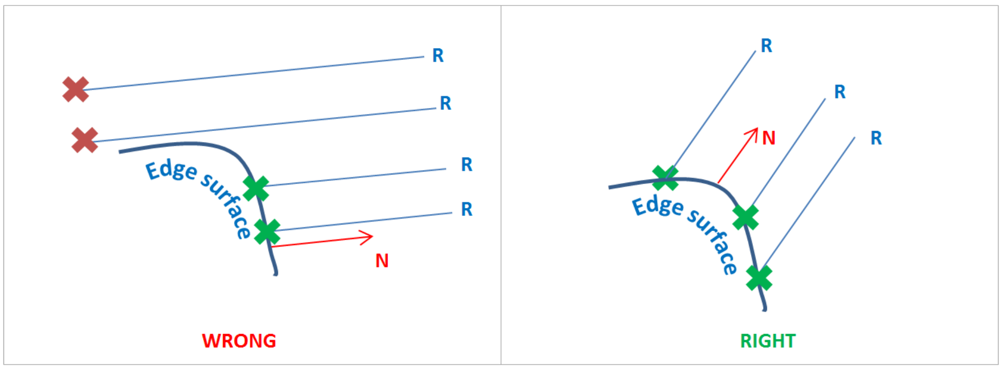


Figure 62: On the left side the initial point is not set directly on the edge, resulting in a normal vector N nearly parallel to the upper part of the edge. Thus, the cast ray R do not intersect with the surface properly. Therefore this is an example of how the initial point should not be selected. The right example is how it should be done by setting the initial point exactly on the edge, resulting in intersection of all cast rays.

cases like this are not common in facial surgeries since implants will not be bend by an angle of this value. Anyway, this case needs to be considered if this work is used as a basis for additional implementations, were it might gain on importance.

The results section also shows some examples of generated implants which align very well to the surface curvature. For the use in a surgeons daily routine, however, it is necessary to work on a data set, were repositioning of the fractured bones already was performed using a software like described in [35], [4] and [26]. Compared to other software tools, commercial available, this software is developed for the positioning of miniplates available on the market rather than generating individual implants like provided by Materialise's patient specific cranio maxillofacial implants [5].

More, the evaluation questionnaire shows that the software is an appropriate tool, reaching high satisfaction by the end users, for pre planning facial reconstruction surgeries. In detail, the users were deeply contented with the very short training time and rated it with a perfect score. Even though, depending on the expertise, the usage of the visualizing panel has to be explained more in detail for the surgeons and the medical background more to the computer engineers, the overall training time is less than five minutes. Further, the subjects were also very satisfied with the provided functions and the adjustment of the software to achieve a satisfying goal. Even though the tool is easy to be extended and thus usable for other working areas, like cartilage surgeries, the software is developed based on the proposed purpose, thus the high rating in this section was predictable. The user interface also reached the maximum score since it impresses with an appealing and professional design, where the objects are arranged properly to provide clearness and a good overview of the included functions and possibilities. Subjects also rated the resulting outcome very high. Not only the accuracy of placing them but also the representation and final position, including the curvature, is very pleasing. A big factor of the proposed thesis is the time factor, since the idea was to implement a software which should also decrease time consumption. This goal is definitely reached, since the average subject took around five minutes until gaining a satisfying result and also rated the consumed time, excluding time spent on training, very high, with maximum score. The overall time consumption including training time, therefore results in very low time consumption since also the training, as mentioned before was rated very good, with a maximum of ten minutes spent. This is the result of a generally automatic work flow, where the user interactions are decreased to just a few ones. Finally, the more interesting ratings are those with a lower value, as for example, the already mentioned difficulty of usage. Also lower rated, with a score of five points, was the adjustment of the implant. Since rotating the baseline in the desired orientation followed by it's repositioning results in a baseline not orientated in the before adjusted direction, the adjustment of the implants was experienced not as an very easy task. Anyway, these two lower ratings still gained a high value but give ideas for improvement. Together with the very high rated overall impression, all of the subjects would use the software in a daily bases if assumed that the resulting implant like visualized in the tool would fit with just a few adjustments, for application on the patient, in it's 3D printed version.

7.1 Future Outlook

The proposed software provides a selection of three implant models from the MedArtis Trauma 2.0 series, which are the most used ones by the medical partners. However, it is not possible to attend all cranial and maxillofacial fractures with the models provided by the software tool and also surgeons from other institutions use different miniplates from this series or even different implant types, why ideas for future implementation include the involvement of a broader range of implant models, thus reaching more end users due to higher flexibility. Also the provided functions should be improved for gaining a better user experience like discussed in the previous section. The mode of individual placement of the single baseline points by dragging them to the desired position on the surface is part of future improvements as well, since this function is included though it is not very user friendly designed yet. Additional, flexibility could be increased by providing a generation mode of patient specific implants like it is provided in other works where [43] and [49] have to be mentioned. These papers propose the generation of patient individual designs which are also saved in STL format and thus being able to be print with the 3D print technology, allowing surgeons to generate the implant within a few minutes, thus possible to be used even during surgery.

Beside improvements of already existing functions, ideas worth to be implemented in the future, include the extension of the software tool with new features. A very important one, is a method for bone repositioning. Since a fracture can cause bones to be displaced, shifting them back to it's original place is generally mandatory. The current state of the software needs this function to be outsourced to another software, thus the implementation of being able to reposition fractured bones is very important to include. More, by adding the option of photo realistic reconstructions or visualizations [44] of overlying soft tissue gives the surgeon an even better idea of what the final result, on the closed patient, will look like. Future improvements will then be possible to visualize a result where also the soft tissues are modeled to have an idea of how the set implant influences the aesthetics of a patient.

Even though the application is developed for facial reconstructions, it is easy to adopt or extend the underlying processes to be used in other body areas, like for example in the vertebral disk. Based on the work of Schwarzenberg in *Cube-Cut: Vertebral Body Segmentation in MRI-Data through Cubic-Shaped Divergences* [38] and Zukic in *Robust Detection and Segmentation for Diagnosis of Vertebral Diseases using Routine MR Images* [51] where vertebral disks are segmented, future work could also include the generation of implants for similar body parts.

As a final proposal for future works, it is very important to test the generated implants in the surgical process, to determine how precise the implants are fitting on the patients defected surface and to determine which areas have to be improved for a better introduction of the software in the surgeon's daily routine. Also it is very important to compare the visualized 3D computer model with the real world outcome and how different they really are. The idea for the far future shows a system, which generates a perfect fitting implant based on the CT scan as single input. By ensuring a broad range on implant generation modes, including individual implants and also those commercial available, the software should be very flexible and easy adjustable to the end user's needs. Independent of the applied area, 3D printing provides a fast and precise solution for implant generation of any type. With all this functions the final system may be used in the surgeons daily life, enhancing precision and flexibility, by lowering cost and time affords at the same time.

A Appendix

Computer-aided Reconstruction of Facial Defects

Markus Gall, Jürgen Wallner, Katja Schwenzer-Zimmerer, Dieter Schmalstieg, Knut Reinbacher, Jan Egger

***Abstract*—In this contribution, a novel method for computer-aided planning of facial surgery using miniplates is proposed. The planning software is able to fit the most common miniplates, to the desired position in a 3D facial model. The placement respects the local surface curvature. The implants can be manufactured with 3D printing technology.**

I. Introduction

Facial reconstruction after bone fractures is an important application of computer-aided surgery [1]. A common method uses so-called miniplates [2], straight titanium plates with at least two finishing ring sections at their ends, where screws for implant-bone fixation are drilled. The implants are available in countless different variations, with additional ring sections in the middle, in bent or loop shapes. In this contribution, we propose a novel method for computer-aided planning of facial surgeries using miniplates.

II. Methods

First, the user chooses an implant type and selects any location on the surface of the facial model to place the implant's center point. Using the center as a seed point, the baseline curvature is calculated by casting rays along the baseline and checking for surface intersection positions. Using the resulting curved baseline, the implant shape is generated by placing precomputed polygonal meshes at the locations along the curved baseline corresponding to the implant's dimensions. Each ring element of the implant is oriented to be aligned with the surface tangent plane at the chosen location, so that a perfect fit is guaranteed. Finally, the straight sections bridging the rings are generated by deforming a template mesh with rectangular footprint. Runtime is optimized by limiting computations to the region of interest around the seed point.

III. Results

The proposed interactive planning software has been implemented in C++ with the medical prototyping platform MeVisLab [3]. Computation runs in real-time on a standard desktop computer (Intel Core i7–930 CPU, 4×2.80 GHz, 6 GB RAM, Windows 8.1), allowing for interactive feedback. Figure 1 shows the result of miniplate placement at a variety of positions.

M. G. is with the Institute for Computer Graphics and Vision of the Graz University of Technology, Austria (e-mail: gall@student.tugraz.at)

J. W. is with the Department of Maxillofacial Surgery of the Medical University of Graz, Austria (e-mail: j.wallner@medunigraz.at)

K. S. is with the Dept. of Maxillofacial Surgery of the Medical University of Graz, Austria (e-mail: katja.schwenzer-zimmerer@medunigraz.at)

D. S. is with the Institute for Computer Graphics and Vision of the Graz University of Technology, Austria (e-mail: schmalstieg@icg.tugraz.at)

K. R. is with the Department of Maxillofacial Surgery of the Medical University of Graz, Austria (e-mail: knut.reinbacher@medunigraz.at)

J. E. is with the Institute for Computer Graphics and Vision of the Graz University of Technology, Austria and BioTechMed, Graz, Austria (phone: +43 316 873 5076; fax: +43 316 873 5050; e-mail: egger@icg.tugraz.at)

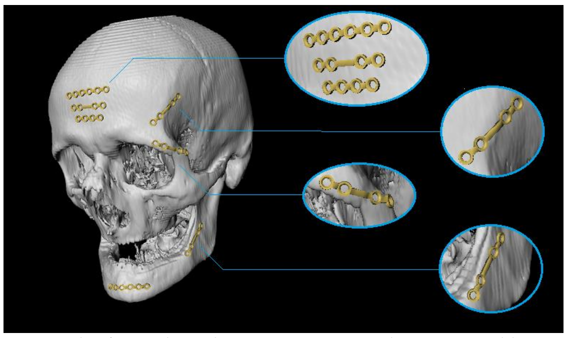


Fig.1: Result of miniplate placement in various directions and locations.

IV. CONCLUSIONS

The software was tested with real patient CT data provided by the Clinical Department of oral and maxillofacial surgery of MedUni (medizinische Universität) Graz. The outcome shows very well aligned and with respect to the surface well bent miniplates, allowing the surgeons to get a good understanding of what a post-intervention will look like. Further, the implant models are stored in STL-file format, which is a common format used in the 3D-print technology. Therefore, surgeons have the opportunity to use the generated implant with 3D printing as a bending tool, for precise bending of the miniplates, or using it as a miniplate itself in the surgical process. Moreover, the physicians describe the handling as very user-friendly and accurate. By selecting the placement point on the patient's surface, the surgeons are able to place the implant at any desired position with the option of further change in position as well as changes in the implant's pointing direction and implant type. In summary, the developed software provides a tool for surgeons, operating in the facial area which is comfortable to use and accurate at the same time.

There are several areas for future work, like offering more complex implants to the user and a comparison and evaluation with commercial software products.

Acknowledgment

BioTechMed-Graz ("*Hardware accelerated intelligent medical imaging*") and MedArtis for providing technical details about the Modus 2.0 series.

References

[1] L. E. Ritacco, F. E. Milano, and E. Chao, "Computer-Assisted Musculoskeletal Surgery," *Springer Press*, pp. 1-326, Nov. 2015.

[2] D. A. Hidalgo, "Titanium Miniplate Fixation in Free Flap Mandible Reconstruction," *Annals of Plastic Surgery,* 23(6):496-507 Dec. 1989.

[3] J. Egger, et al., "Integration of the OpenIGTLink Network Protocol for Image-Guided Therapy with the Medical Platform MeVisLab," *Int J Med Robot.*, 8(3):282-90, Feb. 2012.

Computer-aided Reconstruction of Facial Defects

M. Gall [a] • J. Wallner [b] • K. Schwenzer-Zimmerer [b] • D. Schmalstieg [a] • K. Reinbacher [b] • J. Egger [a,c]

[a] TU Graz, Institute for Computer Graphics and Vision, Inffeldgasse 16, 8010 Graz, Austria
[b] MedUni Graz, Medical University of Graz, Auenbruggerplatz 2, 8036 Graz, Austria
[c] BioTechMed, Krenngasse 37/1, 8010 Graz, Austria

Purpose

Facial reconstruction after bone fractures is an important application of computer-aided surgery[1]. A common method of osteosynthesis are adaptive miniplates[2], titanium made metal plates placed with at least two ring sections per fracture fragment. For plate fixation on the bone special fixation screws are drilled. The implants are available in different sizes and dimensions and are usually bent intraoperativly to adapt them on the underlying bone. In this contribution, we propose a novel method for computer-aided planning and the creation of individually designed patient implants in facial reconstruction using miniplate osteosynthesis.

Methods

CT-datasets from the clinical routine were used in a prospective study for the creation of individual designed osteosynthesis materials. An interactive planning software has been implemented in C++ with the medical prototyping platform MeVisLab[3]. Computation runs in real-time on a standard desktop computer (Intel Core i7–930 CPU, 4 × 2.80 GHz, 6 GB RAM, Windows 8.1), allowing for interactive feedback. On the workstation the user chooses an implant type and selects any location on the surface of the facial model to place the implant's center point. Using the center as a seed point, the baseline curvature is calculated by casting rays along the baseline and checking for surface intersection positions. Using the resulting curved baseline, the implant shape is generated by placing precomputed polygonal meshes at the locations along the curved baseline corresponding to the implant's dimensions. Each ring element of the implant is oriented to be aligned with the surface tangent plane so that the plate fits perfectly to the underlying bone structure. Finally, the straight sections bridging the rings are generated by deforming a template mesh with rectangular footprint. Runtime is optimized by limiting computations to the region of interest around the seed point. Finally plate positions and adaption was independently assessed by two specialists for maxillofacial surgery by completing given tasks by the system. Figure 1 gives an overview of the workflow.

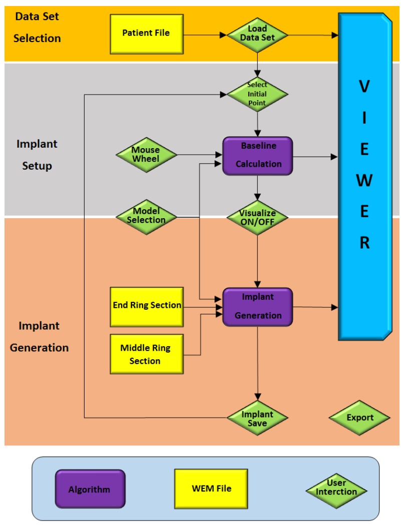


Figure 1 – Overview of the implant generation. In the orange section, the user loads the patient's data set which is then visualized by the *viewer* block. Followed by the implant set up (grey section) where the user sets the initial point (implant center). Next, the baseline is to calculated which shows the user the current position and direction of the implant, depending on the mouse wheel value, the selected implant model and the initial set point is also visualized by the viewer. In the apricot section the implant is calculated *Implant Generation* Block) by activating the *Visualize ON/Off* button. For the implant generation also the implants ring sections, middle and end parts, are required. The final implant is then visualized together with the baseline and the patient's data set. By using the block *Implant Save*, the user saves the current implant(s) for this session and starts setting up new ones. By exporting, the current visualized implants are stored as a STL file on a local path.

Results

Computer-aided bone plate adaption was able for every type of minplate that was used with the software. Virtual plate adaption. provided correct positioning and satisfying results at any position on the facial bones. Medical specialists did neither require any further training time to use the software's functions, nor they fail in completing any given task by the system. Figure 2 shows the result of adaptive miniplate placements at a variety of positions and Figure 3 shows the user interface including a loaded data object, baseline and **individually** generated implant.

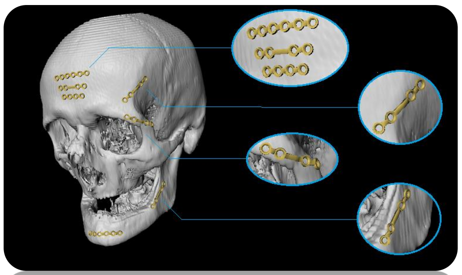
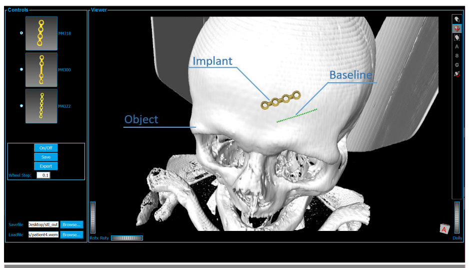


Figure 2 – Result of computer aided adaptive miniplate placements in various directions and locations.

Figure 3 – User interface including object, baseline and generated implant.



Conclusion

The software was used on patient CT data provided from the clinical routine by the Clinical Department of Oral and Maxillofacial Surgery of Medical University Graz. Bone plates were well adapted with respect to the underlying surface and anatomical structures, providing an perfectly fitting osteosynthesis material for an ideal postoperative result in a reduced operation time. Further, the generated implant models can be stored in STL-file format, which is a common format used in 3D-printing. Therefore, surgeons have the opportunity to create the individually designed implant with a 3D printer, anstead of time consuming intraoperative bending of osteosynthesis materials. Moreover, the physicians describe the handling as very user-friendly and accurate. By selecting the placement point on the patient's surface, the surgeons are able to place the implant at any desired position with the option of further change in position as well as changes in the implant's pointing direction and implant type. In summary, the developed software provides a tool for surgeons, to design and in a second step produce individually created patient implants for osteosythesis of facial defects in cranio maxillo facial, within the clinical center but without using any monetary services provided by the industry. Additionally this tool can be easily be tested and further developed by other groups without, since the software is based on an open-source platform.

There are several areas for future work, like offering more complex implants to the user and a comparison and evaluation with commercial software products.

Video Tutorial

https://www.youtube.com/watch?v=Od5xxuERJ8E

Acknowledgement

BioTechMed-Graz ("Hardware accelerated intelligent medical imaging") and MedArtis for providing technical details about the Modus 2.0 series.

References

[1]L. E. Ritacco, F. E. Milano, and E. Chao, "Computer-Assisted Musculoskeletal Surgery," Springer Press, pp. 1-326, Nov. 2015.
[2]D. A. Hidalgo, "Titanium Miniplate Fixation in Free Flap Mandible Reconstruction," Annals of Plastic Surgery, 23(6):496-507 Dec. 1989.
[3]J. Egger, et al., "Integration of the OpenIGTLink Network Protocol for Image-Guided Therapy with the Medical Platform MeVisLab," International Journal of Medical Robotics, 8(3):282-90, Feb. 2012.

EMBC'16 Orlando, FL USA
38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society
August 16-20, 2016
Orlando, FL, USA

References

[1] Autodesk Inventor (San Rafael, Californien, USA). website: http://www.autodesk.de/products/inventor/overview. Last visited on May, 2016. 9

[2] ISO Norm 9241/110 software evaluation questionnaire. website: http://www.ergo-online.de/site.aspx?url=html/software/verfahren_zur_beurteilung_der/beurteilung_der_software_ergo.html. Last visited on June, 2016. 94

[3] Materialise (Leuven, Belgium): Software and services for medical 3D printing. website: http://hospital.materialise.com/. Last visited on May, 2016. 43

[4] Materialise MIMICS Care Suite for cranio-maxillofacial surgeons. website: http://hospital.materialise.com/mimics-care-suite-cranio-maxillofacial-surgeons. Last visited on May, 2016. 42, 43, 99

[5] Materialise OBL. website: http://hospital.materialise.com/cranio-maxillofacial-implants. Last visited on June, 2016. 100

[6] MedArtis. Implant Dimensioning. from March 2016. 34

[7] MedArtis AG (Basel, Switzerland). website: http://www.medartis.com/home/. Last visited on May, 2016. 8

[8] MedArtis: Ordering catalog, modus. website: http://www.medartis.com/uploads/MODUS_00000000_v1_02.pdf. Last visited on June, 2016. 33

[9] Mevislab: Getting started tutorial. website: http://mevislabdownloads.mevis.de/docs/current/MeVisLab/Resources/Documentation/Publish/SDK/GettingStarted/index.html. Last visited on May, 2016. 34

[10] MeVisLab MeVis Medical Solutions AG, Fraunhofer MEVIS (Bremen, Germany). website: http://www.mevis.de/. Last visited on May, 2016. 8

[11] MeVisLab: Reference manual. website: http://mevislabdownloads.mevis.de/docs/current/MeVisLab/Resources/Documentation/Publish/SDK/MeVisLabManual/index.html. Last visited on May, 2016. 34

[12] O. Barkan; J. Weill; A. Averbuch and S. Dekel. Adaptive compressed tomography sensing. *Computer Vision and Pattern Recognition*, pages 2195–2202, June 2013. 32

[13] B. G. Baumgart. Use of polyhedra in computer vision. *National Computer Conference and Exposition*, pages 589–596, May 1975. 27

[14] R. B. Bell. A comparison of fixation techniques in oro-mandibular reconstruction utilizing fibular free flaps. *Journal of Oral and Maxillofacial Surgery*, 65(9):39, September 2007. 42

[15] R. B. Bell. Computer planning and intraoperative navigation in cranio-maxillofacial surgery. *Oral and Maxillofacial Surgery Clinics of North America*, 22(1):135–156, February 2010. 42

[16] T. M. Buzug. *Einführung in die Computertomographie: Mathematisch-physikalische Grundlagen der Bildrekonstruktion.* Springer Berlin Heidelberg, April 2005. 32

[17] F. Wilde; C. P. Cornelius and A. Schramm. Computer-assisted mandibular reconstruction using a patient-specific reconstruction plate fabricated with computer-aided design and manufacturing techniques. *Craniomaxillofac Trauma Reconstructions*, 7(2):158–166, June 2014. 42

[18] G. Cramer. *Introduction à l'analyse des lignes courbes algébriques.* chez les frères Cramer et C. Philibert, 1750. 22

[19] R. Smith-Bindman; J. Lipson; R. Marcus; K. Kim; M. Mahesh; R. Gould; A. Berrington de Gonzlez and D. L. Miglioretti. Radiation dose associated with common computed tomography examinations and the associated lifetime attributable risk of cancer. *Arch internal medicine*, 169(22):2078–2086, December 2009. 31

[20] D. Z. Dhu. *Computing in Euclidean Geometry*, volume 1, chapter Optimal Triangulation, pages 29–43. World Scientific, 1992. 26

[21] P. Mildenberger; M. Eichelberg and E. Martin. Introduction to the dicom standard. *European Radiology*, 12(4):920–927, September 2002. 30

[22] H. T. Gabor. *Fundamentals of Computerized Tomography.* Springer, 2 edition, 2009. 32

[23] J. Egger; Z. Mostarkic; S. Grosskopf and B. Freisleben. A fast vessel centerline extraction algorithm for catheter simulation. *Twentieth IEEE International Symposium on Computer-Based Medical Systems (CBMS'07)*, pages 177–182, June 2007. 21

[24] D. A. Hidalgo. Titanium miniplate fixation in free flap mandible reconstruction. *Annals of Plastic Surgery*, 23(6):469–507, December 1989. 8

[25] J. Hiesh and SPIE (Society). *Computed Tomography: Principles, Design, Artifacts, and Recent Advances*. SPIE PM. 2009. 32

[26] P. C. Everett; E. B. Seldin; M. Troulis; L. B. Kaban and R. Kikinis. A 3D system for planning and simulating minimally-invasive distraction osteogenesis of the facial skeleton. *MICCAI 2000: Third International Conference, Pittsburgh, PA, USA*, 1935:1029–1039, October 2000. 99

[27] J. Egger; J. Tokuda; L. Chauvin; B. Freisleben; C. Nimsky; T. Kapur and W. Wells. Integration of the openigtlink network protocol for image-guided therapy with the medical platform mevislab. *The international Journal of medical Robotics and Computer assisted Surgery*, 8(3):282–390, September 2012. 8

[28] A. Katz. *Computational Rigid Vehicle Dynamics*. Krieger Publishing Co., Inc., Melbourne, FL, USA, 1997. 23

[29] E. B. Dam; M. Koch and M. Lillholm. Quaternions, interpolation and animation. Technical report, DIKU-TR-98/5, Department of Computer Science University of Copenhagen, 1998. 25

[30] M. Gopi; S. Krishnan and C.T. Silva. Surface reconstruction based on lower dimensional localized delaunay triangulation. *Computer Grapics Forum*, 19(3):467–478, September 2000. 26

[31] J. B. Kuipers. Quaternions and rotation sequences. *Proceedings of the International Conference on Geometry, Integrability and Quantization*, pages 127–143, September 1999. 24

[32] J. Chapuis; A. Schramm; I. Pappas; W. Hallermann; K. Schwenzer-Zimmerer; F. Langlotz and M. Caversaccio. A new system for computer-aided preoperative planning and intraoperative navigation during corrective jaw surgery. *Trans. Info. Tech. Biomed.*, 11(3):274–287, may 2007. 43

[33] C. K. Chua; K. F. Leong and C. S. Lim. *Rapid Prototyping Principles and Applications*. World Scientific, 3 edition, 2010. 29

[34] F. Wilde; K. Winter; K. Kletsch; K. Lorenz and A. Schramm. Mandible reconstruction using patient-specific pre-bent reconstruction plates: comparison of standard and transfer key methods. *International Journal of Computer Assisted Radiology and Surgery*, 10(2):129–140, May 2014. 42

[35] R. Marmulla and H. Niederdellmann. Surgical planning of computer-assisted repositioning osteotomies. *Plastic and Reconstructive Surgery*, 104(4):973–975, September 2016. 99

[36] L. E. Ritacco; F. E. Milano and E. Chao, editors. *Computer-Assisted Musculosketal Surgery*. Springer Press, 2015. 8

[37] T. Móller and B Trumbore. Fast, minimum storage ray/triangle intersection. *Journal of Graphics Tools*, 2(1):21–28, 1997. 21, 23

[38] R. Schwarzenberg; B. Freisleben; C. Nimsky and J. Egger. Cube-cut: Vertebral body segmentation in mri-data through cubic-shaped divergences. *PLoS ONE*, 9(4):1–17, April 2014. 101

[39] Y. S. Zhang; K. Yue; J. Aleman; K. Mollazadeh-Moghaddam; S. M. Bakht; J. Yang; W. Jia; V. DellErba; P. Assawes; S. R. Shin; M. R. Dokmeci; R. Oklu and A. Khademhosseini. 3D bioprinting for tissue and organ fabrication. *Annals of Biomedical Engineering*, pages 1–16, April 2016. 31

[40] S. H. Little; M. Vukicevic; E. Avenatti; M. Ramchandani and C. M. Barker. 3D printed modeling for patient-specific mitral valve interventionrepair with a clip and a plug. *JACC: Cardiovascular Interventions*, 9(9):973–975, May 2016. 31

[41] M. Rasse. *Traumatologie des Mund-, Kiefer-, Gesichtsbereichs*, chapter Grundlagen der Traumatologie, pages 1–26. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. 10, 14, 15, 16

[42] M. Rasse. *Traumatologie des Mund-, Kiefer-, Gesichtsbereichs*, chapter Spezielle Traumatologie, pages 27–82. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. 16, 17, 18, 19, 20, 21

[43] M. Gall; X. Li; X. Chen; D. Schmalstieg and J. Egger. Computer-aided planning of cranial 3D implants. *CARS 2016 Computer Assisted*

Radiology and Surgery - Proceedings of the 30th International Congress and Exhibition Heidelberg, Germany,, 11(1):1–286, June 2016. 101

[44] P. Voglreiter; J. Wallner; K. Reinbacher; K. C. Schwenzer-Zimmerer; D. Schmalstieg and J. Egger. Global illumination rendering for high-quality volume visualization in the medical domain. *face 2 face - Science meets art*, October 2015. 101

[45] K. Shoemake. Animating rotation with quaternion curves. *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, 19(3):245–254, July 1985. 25

[46] R. Gassner; H. Ulmer; T. Tuli and R. Emshoff. Incidence of oral and maxillofacial skiing injuries due to different injury mechanisms. *American Assosiation of Oral and Maxillofacial Surgeons*, 57(9):1068–1073, September 1999. 10

[47] T. Tuli. Facial trauma: How dangerous is skiing and snowboarding. *American Association of Oral and Maxillofacial Surgeons*, 68(2):293–299, February 2010. 10

[48] F. Rengier; A. Mehndiratta; H. von Tengg-Kobligk; C. M. Zechmann; R. Unterhinninghofen; H.-U. Kauczor and F. L. Giesel. 3D printing based on imaging data: review of medical applications. *Int J CARS*, 5(4):335–341, May 2010. 30

[49] X. Chen; L. Xu; Y. Yang and J. Egger. A semi-automatic computer-aided method for surgical template design. *Scientific reports*, pages 1–18, February 2016. 101

[50] P. Li; W. Tang; C. Liao; P. Tan; J. Zhang and W. Tian. Clinical evaluation of computer-assisted surgical technique in the treatment of comminuted mandibular fractures. *Journal of Oral and Maxillofacial Surgery, Medicine, and Pathology*, 27(3):332–336, May 2015. 42

[51] D. Zukic. Robust detection and segmentation for diagnosis of vertebral diseases using routine mr images. *Computer Graphics Forum*, 33(6):190–204, March 2014. 101