Computerized Automatic Modeling of Medical Prostheses

Der Technischen Fakultät der
Universität Erlangen–Nürnberg

zur Erlangung des Grades

DOKTOR–INGENIEUR

vorgelegt von

Konrad Sickel

Erlangen — 2013

Deutscher Titel:

Computergesteuerte automatische Modellierung von medizinischen Prothesen

Als Dissertation genehmigt von der
Technischen Fakultät der
Universität Erlangen-Nürnberg

Tag der Einreichung:	28.09. 2012
Tag der Promotion:	18.04. 2013
Dekan:	Prof. Dr.-Ing. Marion Merklein
Berichterstatter:	Prof. Dr.-Ing. Joachim Hornegger
	Prof. Dr. rer. nat. Dipl.-Ing. Tim Lüth

It is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail.

Abraham Maslow, 1966

Zusammenfassung

In dieser Arbeit wird die Anwendung von Methoden der Künstlichen Intelligenz für das automatisierte Design von individualisierten medizinischen Prothesen untersucht. Der Schwerpunkt liegt dabei auf regelbasierten Expertensystemen. Der Begriff Design wird im Folgenden als Modellierung der physikalischen Form der Prothese verstanden und nicht als das funktionale Design. Das Design einer individualisierten Prothese ist sehr komplex, da diese exakt auf die individuelle Anatomie eines Patienten abgestimmt sein muss und in vielen Fällen weitere Funktionen erfüllt. Um dies zu erreichen, basieren die Prothesen auf einem gegebenen Modell der Patientenanatomie. Diese Anatomiedaten können beispielsweise mit Hilfe von CT Daten oder auch von gescannten Abdrücken erzeugt werden. In der vorliegenden Arbeit gehen wir davon aus, dass die Daten in Form von triangulierten 3-D Gittern zur Verfügung stehen.

Das automatisierte Modellieren einer individuellen medizinischen Prothese wurde durch die Entwicklung eines Expertensystems gelöst. Dieses besteht aus einer sogenannten *expert system shell* und einer Wissensdatenbank. Zusätzlich wurde ein System zur Merkmalserkennung für die gegebenen 3-D Daten entwickelt. Beide Systeme wurden in eine vorhandene Modellierungssoftware integriert. Das daraus resultierende Gesamtsystem wird im Folgenden als *Expert System for Automatic Modeling* (ESAM) bezeichnet. Die Architektur des entwickelten Gesamtsystems ist generisch und kann für verschiedenste Designaufgaben mit Anpassung der Wissensdatenbank und der Merkmalserkennung spezialisiert werden.

Das Expertensystem ist konzipiert für die Steuerung und Kontrolle der Modellierungssoftware. Es steuert die Parameter der vorhandenen Modellierungsfunktionen, führt diese Funktionen aus und kontrolliert das Ergebnis. In diesem Zusammenhang wurde zur Speicherung und Strukturierung des Expertenwissens eine gut verständliche und einfach erweiterbare Wissensrepräsentationsprache entwickelt. Die benötigte Wissensdatenbank wurde in Interaktion mit Fachexperten generiert. Außerdem wurden Methoden zur Erweiterung, Verbesserung und Pflege der Wissensdatenbank untersucht. Zur Regelerzeugung und Regelkritik wurde zum einen genetische Programmierung, zum anderen eine heuristische Methode angewendet und ausgewertet. Bei Letzterer werden vom ESAM erzeugte Modellierungsdaten untersucht und ausgewertet. Dafür wurde eine Software zur Auswertung von Regelabhängigkeiten, Regelqualität und menschlicher Interaktion mit dem System entwickelt. Die dabei gewonnenen Erkenntnisse wurden in die Wissensdatenbank integriert und führten zu einer messbaren Verbesserung des Systems. Beispielsweise wurde die ESAM-Fertigungsrate um ca. 30 % verbessert.

Für die Erkennung der benötigten Merkmale auf den triangulierten Oberflächenmodellen werden zwei verschieden Methoden angewendet. Die erste Methode basiert auf der Analyse von generischen Merkmalen der gegebenen 3-D Daten. Diese Merkmale können Senken, Kanten, Erhebungen oder Kombinationen dieser sein. Der generische Ansatz erlaubt eine einfache Anpassung für verschiedenste Anatomien. Die zweite Methode basiert auf der Übertragung der Merkmale von einem beschrifteten Modell auf ein neues Modell. Diese Methode nutzt im ersten Schritt Clustering, um eine geeignete Menge von repräsentativen Modellen zu identifizieren. Diese werden im zweiten Schritt von einem Experten beschriftet. Anschließend können die Merkmale

von einem bereits beschrifteten Modell auf ein unbeschriftetes übertragen werden. Um eine qualitativ gute Übertragung zu gewährleisten, wird das neue Modell mit dem beschrifteten registriert. Experimentell konnte gezeigt werden, dass die zweite Methode bessere Ergebnisse erzielt. So konnte die durschnittliche Abweichung von ca. 3.8 mm um mehr als 30 % auf ca. 2.6 mm verringert werden.

Die Verifikation des ESAM wurde an dem Designproblem von in-dem-Ohr (idO) Hörgeräten durchgeführt. Ein Industriepartner stellte sowohl die notwendige Fachexpertise zur Erstellung der Wissensdatenbank, als auch die Möglichkeit, ESAM in einer realen Produktionsumgebung zu verifizieren, zur Verfügung. Um Qualitätseinbußen in der Fertigung zu vermeiden, wurde die Verifikation mit einem halbautomatischen ESAM System durchgeführt. Dieses erlaubt es den Modellierungsexperten, das System jederzeit zu korrigieren. Während der Verifikation des ESAM wurden Tausende idO Hörgeräte gebaut. Im Vergleich zum manuellen Design wurde dabei eine Verbesserung der Designkonsistenz um 10 % erreicht, bei gleichzeitiger Reduzierung der Designzeit um 30 %. Die durchschnittliche Akzeptanz einer Expertensystemregel liegt bei 76 %. Außerdem verhindert der Einsatz des ESAM Design- und Prozessfehler, da es den Anwender durch den Prozess leitet. Auf diese Weise werden weder Prozessschritte vergessen noch unnötige Schritte durchgeführt. Als Konsequenz der guten Ergebnisse und Vorteile des entwickelten Systems wird es weiterhin bei unserem Industriepartner eingesetzt.

Abstract

In this thesis we study artificial intelligence methods, rule-based expert systems in particular, for the task of automatically designing customized medical prostheses. Here, the term design denotes the shaping or modeling of the prostheses and not their functional design. The challenge of the task at hand lies in designing prostheses that fit perfectly to the anatomy of the patient, and in many cases have to support additional functionality. Hence, each prosthesis has to be unique. Therefore, medical prostheses are usually designed starting with a template of the patient's anatomy, e. g. acquired using CT data or scanned and digitized molds. In this thesis we assume that the template data is given as a triangle mesh in 3-D.

To address the challenge of automatically designing medical prostheses, we develop an expert system framework consisting of an expert system shell, a knowledge base and a feature detection unit. The framework is integrated into an existing modeling software. In the following, we denote the complete system as *Expert System for Automatic Modeling* (ESAM). The architecture of ESAM is generic and can be used for all kinds of design tasks. The specialization for the application in mind can be achieved by providing the necessary design rules and by adjusting the feature detection algorithms.

Our expert system specializes in monitoring and controlling a CAD software. Thus, it defines the parameters of the CAD tools, executes the tools and monitors the results by constantly analyzing the current shape. As part of the expert system we develop a knowledge representation language to structure and store the expert knowledge. The language is easy to understand, flexible and can be extended as required. The knowledge base is created in interaction with experts of the field. In addition, we study methods to extend, improve and maintain the knowledge base. We evaluate two methods for rule creation and rule critic. On the one hand, we apply genetic programming as a rule learning technique. On the other hand, we use a heuristic method based on data generated by ESAM. For the latter, we develop a tool that generates statistics about rule performance, rule relationships and user interaction. The thereby gained knowledge is integrated into the knowledge base, resulting in a higher performance of ESAM, e. g. the completion rate increased by about 30 %.

We apply two types of feature detection methods for the detection of surface features on the given templates. The first method analyzes the surface of the given template for peaks, depressions, ridges and combinations of these generic features. The generality of the detected features allows a simple adjustment to different anatomies. The second method uses registration in order to copy features from a labeled template to an unlabeled one. As a first step, it applies clustering techniques to identify a suitable set of representative templates. In the second step, these templates are labeled by a domain expert. Subsequently, the labels can be transferred based on the result of an ICP registration. Our experiments show that the second approach results in a higher quality of the detected features, e. g. mean deviation is reduced from approximately 3.8 mm by about 30 % to approximately 2.6 mm.

ESAM is verified using the example of customized in-the-ear hearing aid design. An industry partner provides the domain knowledge necessary to create the knowledge base as well as the possibility to verify the system in a real production envi-

ronment. In order to ensure the quality of the designed and manufactured in-the-ear hearing aids, the system is verified while running in a semi-automatic mode. The semi-automatic mode allows a modeling expert to monitor and correct the system if necessary. During the verification and practical usage of ESAM thousands of customized in-the-ear hearing aid shells are manufactured. It could be shown that compared to the manual approach the design consistency improves by about 10 % and the design time is reduced by about 30 %. The overall acceptance rate of an expert system rule is 76 %. In addition to that, ESAM provides a framework, which guides the modeler through the complex design process, and thereby reduces the amount of design errors and avoids unnecessary process steps. As a consequence of these positive evaluation results our industry partner continues to apply ESAM on its production floor.

Acknowledgment

I would like to take this chance to thank all the people who helped me during the development of this work. First and foremost my supervisor Prof. Joachim Hornegger, Ph.D., the head of the Pattern Recognition Lab (LME) of the University of Erlangen-Nuremberg. He was the one who encouraged me to go on the long journey for obtaining a doctoral degree. Furthermore, Jörg Bindner, who is with the Siemens AG, who made it actually possible by providing a project and financial support. Vojtech Bubnik, Andreas Reh and Bert Barthelmes, who are with the Siemens AG, for providing a enjoyable working environment and supporting the thesis with ideas and collaborations. All my colleagues at the Pattern Recognition Lab for their support and inspiration. In particular, I would like to thank Michael Balda for proof reading and valuable feedback. Special thanks are to my wife, Karin, for proof reading and extensive support - without you it would not have been possible!

Erlangen, April 2013
Konrad Sickel

Contents

1 Introduction **1**
- 1.1 Objective . . . 1
- 1.2 Motivation for the Application of a Rule-based Expert System . . . 1
- 1.3 Rule-based Expert Systems: Terms and State-of-the-Art . . . 3
  - 1.3.1 Rule-based Expert System Design . . . 4
  - 1.3.2 State-of-the-Art . . . 5
- 1.4 Main Contributions . . . 7
- 1.5 Thesis Overview . . . 7

2 Background on Medical Prostheses **9**
- 2.1 Terminology . . . 9
  - 2.1.1 Necessary Conditions for Automation Framework . . . 9
- 2.2 Historical Roots . . . 10
- 2.3 Types of Medical Prostheses . . . 12
  - 2.3.1 Exoprostheses . . . 12
  - 2.3.2 Open Prostheses . . . 14
  - 2.3.3 Endoprostheses / Implants . . . 16
- 2.4 Medical Prostheses Design: State-of-the-Art . . . 18
  - 2.4.1 Rapid Prototyping . . . 18
  - 2.4.2 Craniomaxillofacial Implants . . . 19
  - 2.4.3 Dental Crowns . . . 21
  - 2.4.4 Customized Hearing Aids . . . 23
- 2.5 Conclusions . . . 27

3 Detection of Anatomical Features on Organic Shapes **29**
- 3.1 Anatomical Features of Ear Impressions . . . 29
- 3.2 Surface-based Feature Detection . . . 33
  - 3.2.1 Generic Feature Detection Algorithms . . . 33
  - 3.2.2 Surface-based Feature Detection for Ear Impressions . . . 38
  - 3.2.3 Update of Detected Features . . . 40
- 3.3 Clustering-based Feature Detection . . . 41
  - 3.3.1 Robust Alignment of Ear Impressions . . . 43
  - 3.3.2 Identification of a Representative Set of Ear Impressions . . . 51
  - 3.3.3 Agglomerative Hierarchical Clustering . . . 52
  - 3.3.4 Expectation Maximization Clustering . . . 53
  - 3.3.5 Chinese Restaurant Clustering . . . 56

3.3.6 Usage of Clustering-based Features . . . 63
3.4 Experiments and Results . . . 63
3.4.1 Evaluation of the ICP Algorithm for Ear Impressions . . . 63
3.4.2 Experiments for Identification of a Representative Sample Set 68
3.4.3 Evaluation of Surface-based Feature Detection . . . 72
3.4.4 Evaluation of Clustering-based Feature Detection . . . 72
3.5 Conclusions . . . 76

4 Rule-based Expert System **77**
4.1 Script Language . . . 77
4.1.1 Motivation and Requirements . . . 77
4.1.2 The ES Shell – Script Parser and Interpreter . . . 78
4.1.3 CAD-Integration and Functionality . . . 81
4.1.4 Internal Functions . . . 83
4.1.5 Rules . . . 88
4.2 Knowledge Acquisition . . . 89
4.3 Preliminary Experiments and Results . . . 90
4.3.1 Experiments . . . 90
4.3.2 Results . . . 91
4.4 Conclusions . . . 92

5 Rule Learning **93**
5.1 Data and Learning Objectives . . . 93
5.1.1 The Data . . . 93
5.1.2 The Learning Objectives . . . 95
5.2 Genetic Programming for Rule Learning . . . 96
5.2.1 Introduction and Motivation for Using Genetic Programming . 96
5.2.2 The Primitive Set . . . 98
5.2.3 Initial Population . . . 99
5.2.4 Fitness Evaluation . . . 99
5.2.5 Genetic Operators . . . 101
5.2.6 Genetic Programming Framework . . . 103
5.2.7 Evaluation of the GP-Framework . . . 105
5.2.8 Application Evaluation and Conclusions . . . 107
5.3 Semi-automatic Rule-learning . . . 107
5.3.1 Rule Types . . . 107
5.3.2 Classification of Rules based on Performance . . . 108
5.3.3 Drawing Conclusions from User Modifications . . . 111
5.4 Conclusions . . . 126

6 Expert System for Automatic Modeling **129**
6.1 The ESAM Framework . . . 129
6.2 ESAM Development and User Acceptance . . . 131
6.2.1 ESAM Development . . . 131
6.2.2 Evolution of the Knowledge Base . . . 131
6.3 ESAM Verification . . . 133
6.3.1 ESAM User Acceptance . . . 133

6.3.2 General Rule Performance . . . 134
6.3.3 Design Consistency . . . 137
6.3.4 Design Quality . . . 138
6.3.5 Design Speed . . . 140
6.4 Comparison to Other Works . . . 141
6.4.1 Statistical Shape Models of the Human Ear Canal . . . 142
6.4.2 Generating Shapes by Analogies . . . 143
6.4.3 Specialized CAD / Modeling Software for Customized Design 144
6.5 Conclusions . . . 145

7 Conclusions **147**
7.1 Summary . . . 147
7.2 Conclusions and Outlook . . . 149

A Knowledge Representation Language – *The Script* **151**
A.1 Lexical Analyzer . . . 151
A.2 Parser . . . 153

List of Figures **161**

List of Tables **167**

Bibliography **171**

Chapter 1

Introduction

For, usually and fitly, the presence of an introduction is held to imply that there is something of consequence and importance to be introduced.

Arthur Machen

1.1 Objective

The objective of this thesis is to study the feasibility of applying artificial intelligence methods, rule-based expert systems in particular, to improve the design of customized medical prostheses.

This objective is subdivided in three goals:

1. To develop a framework to automate the design of medical prostheses.
2. To verify the developed framework in a real world environment.
3. To apply learning techniques to further improve the framework either automatically or semi-automatically.

In addition, we show that the results obtained in the project help to improve the design quality and design consistency as well as to reduce the labor involved in manufacturing customized prostheses, and, thus providing higher quality prostheses to the impaired.

1.2 Motivation for the Application of a Rule-based Expert System

The objectives stated above may be fulfilled using different methods and approaches. Thus, in the beginning of this thesis several design decisions had to be made. For example, we considered several approaches of the artificial intelligence domain. For example, (multi-) agent planning systems, decision trees, expert systems, knowledge representation / reasoning with or without probability factors, etc.

Furthermore, statistically driven methods, e. g. statistical shape models, were considered as well. Given that Rasmus R. Paulsen [Paul 04a] already applied a statistics

based approach in his Phd thesis and Alexander Zouhar [Zouh 09, Zouh 10] also works on statistics based methods with similar objectives, we decided to apply a reasoning based method. In the end, we chose to apply the expert system methodology, which we motivate in the following paragraph.

In the environment of medical prostheses design a vast amount of knowledge is available, be it in form of written instructions or experience acquired during the design of thousands of individual prostheses. In order to preserve and utilize this knowledge, we developed a rule-based expert system. It contains and permanently stores the expert knowledge in its rule base. The acquisition and further encoding of the knowledge in a rule base has a number of key benefits [Turb 88, Turb 95]:

- *Increased availability:* The knowledge is available on any suitable hardware without the need of a real expert.
- *Permanence:* Once the knowledge is stored, it cannot retire or die, but will last indefinitely.
- *Scarce knowledge and multiple expertise* is contained, because of an extensive knowledge acquisition process, often including multiple domain experts. Therefore, it may exceed the knowledge of a single human expert.
- *Unemotional:* Expert systems (ES) are not effected by stress or emotions.
- *Educational benefits:* ES can be applied as training or educational systems to spread the expertise without the need of a human expert.
- *Accuracy:* They allow direct and precise control over the automation process and therefore offer the possibility to react quickly on process changes without the need to recompile the whole system.
- *Transparency/Explanation:* They natively offer an explanation component.
- *User interaction:* ES support the possibility of user interaction and therefore the integration of expert judgment to further enhance the found solution.

In our case, the explanation component and the possibility for user interaction are crucial. A customized medical prostheses may be implanted into a skull, fitted to a leg residual or into an ear. Therefore, its design should be transparent and comprehensible. Hence, *black box* or one step solutions are regarded with mistrust by designers and manufacturers of prostheses. A rule-based expert system is transparent, because of its explanation component. Furthermore, the possibility to actively influence the decision making, is a psychologically important factor to use automation in the design and manufacturing environment of customized medical prostheses at all. The above mentioned reasons led to the decision, to employ an expert system to automate the design of customized medical prostheses.

1.3 Rule-based Expert Systems: Terms and State-of-the-Art

Giarratano and Riley authored a well known book *Expert Systems: Principles and Programming* about ES, including theory and practical advice in combination with the ES framework CLIPS [Giar 05]. In the following, we summarize their work to give a brief introduction about ES and include other sources where necessary.

The terms expert systems, knowledge-based systems (KBS), rule-based systems (RBS), knowledge-based expert systems (KBES), and rule-based expert systems (RBES) are used synonymously in the literature. A common term would be intelligent system, but mostly expert system is used as a generic term [Hopg 01]. In literature also neural networks, case-based reasoning, intelligent agent systems, ontology and more are considered as ES methodology [Liao 05]. In our work, we will use the term expert system or rule-based expert system to express that our knowledge is encoded in rules and utilizes knowledge only available from experts in a certain domain.

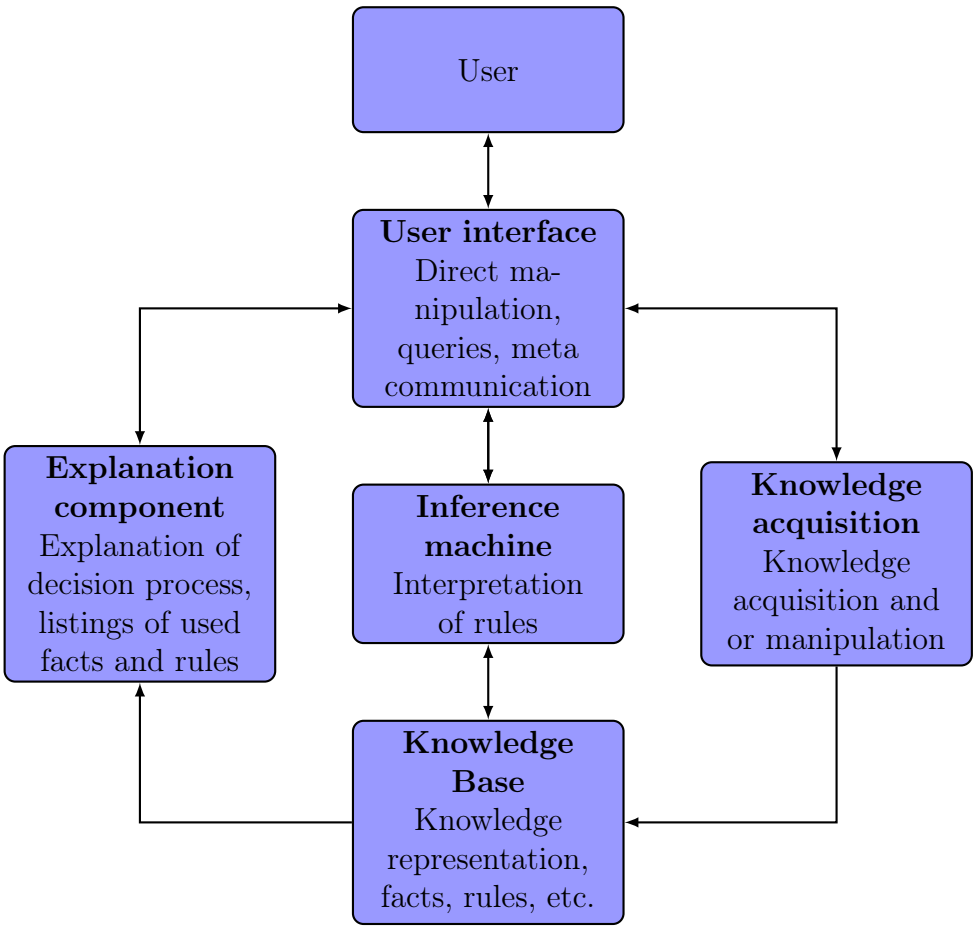


Figure 1.1: Structure of a rule-based expert system [VDI 92].

ES are artificial intelligent (AI) tools which were introduced in the mid-1960s and are working in a narrow domain [Liao 05]. They can infer[1] intelligent decisions and offer a way to trace the decision making process. Expert systems incorporate a repository of expert knowledge, which is acquired and represented using various knowledge representation techniques, such as rules, objects, semantic nets, schemata, frames, scripts, conceptual graphs and logic [Aker 09]. The distinctive feature of ES is the separation of knowledge and inference [Luca 91]. For the first time ever, this was demonstrated by the famous MYCIN expert system [Buch 84]. Later the name expert system *shell* was introduced to name a generic expert system, providing language, inference machine and explanation facility with an empty knowledge base. In general an ES is characterized by consisting of:

- a knowledge base (also referred to as rule base) which encodes the available knowledge,
- an inference machine which interprets the knowledge to draw conclusions or execute actions,
- and an user interface to communicate with the user.

In case of complex systems often a knowledge acquisition component and an explanation component are added [VDI 92], see Figure 1.1.

1.3.1 Rule-based Expert System Design

A challenge in ES design is the extraction of expert knowledge. In extensive interviews a knowledge engineer has to gather, structure and formalize the expertise. Typically, a logic based grammar is applied to structure and store the knowledge. The knowledge acquisition process can be envisioned as a cycle, see Figure 1.2. First, the knowledge engineer extracts expertise or clarifies misunderstandings. Second, with the new information a prototype ES is build or an earlier prototype is updated. Third, the prototype is evaluated and criticized by the expert, which directly leads to step one.

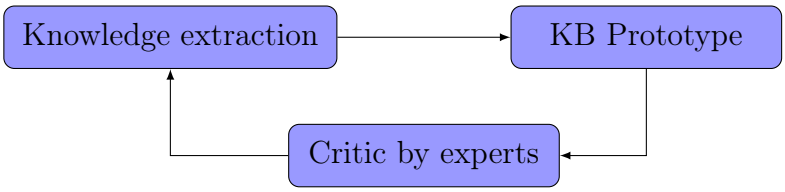


Figure 1.2: Knowledge Base development cycle showing the interaction of knowledge engineer and domain experts.

Knowledge Representation

Knowledge representation (KR) in computer sciences aims to represent knowledge in a manner as to facilitate inferencing from knowledge. In ES, knowledge is usually

[1] In the ES domain, the term inferencing is used instead of reasoning to make a clear distinction between the human reasoning and the machine reasoning process.

represented in production rules, objects, semantic nets, schemata, frames, scripts, conceptual graphs and logic.

Different languages have been developed to store and process knowledge, such as KL-ONE or CLASSIC. These languages are typically based on logic and mathematics, and have easily parsed grammars to ease machine processing. Further examples are XML-based languages and standards, such as Resource Description Framework (RDF), RDF Schema, DARPA Agent Markup Language (DAML), Ontology Inference Layer (OIL), Web Ontology Language (OWL) and the object orientated approach CLIPS+COOL. Rules are commonly used to define the knowledge bases of ES. We will focus on the rule-based representation, because we utilized this kind of knowledge representation. An overview of the other representation techniques is given in Chapter 2 of the book Expert Systems by Giarratano and Riley [Giar 05].

Uncertainty and Missing Information

Uncertainty and missing information can be handled well by knowledge based systems. There exist different approaches to extend an ES if necessary depending on what type of errors (ambiguity, incompleteness, incorrectness, measurement inconsistency, random, systematic, reasoning) are expected or encountered. For example, in case of missing or incomplete information an ES will not break, it can still infer a solution. However, the solution is probably not the optimal one. In addition, it is possible to identify missing data and query the user after the missing information.

In case of uncertainty about the given expertise or facts, a common approach is to add certainty coefficients to facts or rules, as it was done for the diagnosis system MYCIN [Buch 84]. Reasoning under uncertainty requires an additional theory about how to handle and combine certainty factors. Mainly Bayesian probability (inexact reasoning) and fuzzy logic (approximate reasoning) have been applied. A detailed description about incorporating uncertainty can be found in [Giar 05]. In our work, we can assume complete information, with the drawback that the information can be inaccurate or misleading. However, these cases can be identified by our support rules in the knowledge base, which then trigger correction rules.

1.3.2 State-of-the-Art

ES are used in various fields like business, science, engineering, agriculture, manufacturing and medicine. They are often called by usage orientated names like business logic, wizard or configuration tool [Luca 91, Feig 96, Liao 05]. They can be classified in groups depending on their purpose: configuration management, diagnosis, instruction (teaching), interpretation, monitoring, planning, prognosis, remedy and control. The latter typically requires competency in monitoring, planning and diagnosis.

Nowadays, a huge number of well defined and optimized ES shells exist. A shell is a complete development environment for building and maintaining a knowledge-based application. It typically provides the knowledge engineer with an interface to encode and structure the knowledge. Popular examples of ES shells in science are Drools [Bali 09], CLIPS [Rile 10], JESS [Frie 03] and d3web [Lehr 10]. Examples for

commercially distributed ES or ES shells are Corvid[2], XpertRule[3] and Knowledge Automation System[4]. These systems often feature enhanced user interfaces, such as graphical representation and manipulation of rules. Each of these systems offers a unique language to store the knowledge. For example CLIPS provides a LISP like language allowing the definition of rules based on facts and objects.

As inference engine either forward or backward chaining is provided. In forward chaining (data driven reasoning) the system starts with a fact and resolves rules until all possible rules are resolved. In backward chaining (goal driven reasoning) the system starts with a question about a fact and tries to verify or disprove it. It depends on the type of ES, which reasoning should be employed. For example, diagnosis is handled best with backward chaining, while monitoring or control are handled better by forward chaining. All the shells employ optimization strategies like the Rete-Algorithm, backtracking, hierarchical planning or constraint handling to adopt to the problem and speed up the inference process [Forg 79]. Our ES belongs to the forward chaining systems, because its main focus is the control of a computer aided design (CAD) environment, which requires to infer the next step given the current facts.

An exhaustive survey about ES methodologies and applications is given by Liao [Liao 05]. Another review paper by Dowling and Leech [Dowl 07] offers a compact overview about expert systems and decision support systems in general for the example case of audit systems. In the field of diagnosis, in particular in fault diagnosis, various approaches and systems have been developed. A broad overview, including rule-based diagnostic expert systems, model-based diagnostic expert systems and on-line diagnostic expert systems is given by Angeli in [Ange 10].

ES are affected by the trend in the artificial intelligence community to develop hybrid, combined or coupled systems using different techniques of the field like artificial neural networks with ES, genetic algorithms with ES or case-based reasoning [Ange 10]. Recent examples are a fuzzy expert system for high speed milling [Iqba 05], a hybrid system using case-based reasoning and ES by Mok et al. [Mok 08]. Furthermore, ES are used in the automotive industry, e. g. for the configuration of customization options or failure reports analysis [Krus 10].

The developments concerning ES and CAD are of primary importance for us. Unfortunately, only little work is reported regarding the usage of ES in CAD environments. Most of the commercial CAD packages like AutoCAD[5], Magics[6] and ShellDesigner[7] offer a certain amount of customization or APIs (Application Programming Interface) to add automation features [Rich 07]. However, these extensions are limited in a way that they are hard-coded and and cannot be modified, extended or corrected as it would be the case with knowledge based-systems.

[2]Exsys Inc., Albuquerque, New Mexico, US, http://www.exsys.com/index.html, last visited 11/01/2012

[3]XpertRule Software, Salford, UK, http://www.xpertrule.com/pages/index.htm, last visited 11/01/2012

[4]Vanguard Software Corporation, Cary, North Carolina, US, http://www.vanguardsw.com/products/knowledge-automation-system, last visited 11/01/2012

[5]Autodesk, Inc., San Rafael, California, US, http://www.autodesk.com, last visited 09/01/2010

[6]Materialise Group, Leuven, Belgium, http://www.materialise.com, last visited 09/01/2010

[7]3Shape A/S, Copenhagen, Denmark, http://3shape.com/, last visited 09/01/2010.

The usage of an ES in combination with CAD-systems was proposed by Mok et al. [Mok 08], who developed a system to increase the efficiency of injection mold design. Howard and Lewis extended the AutoCAD system with an ES [Howa 06]. However, in contrast to us, their focus was to integrate an effective possibility to select suitable materials for the designed object considering material properties, costs and availability instead of designing the shape of the object with help of the ES. Works by Hieu et al. [Hieu 02, Hieu 10], Strub et al. [Stru 06], Adolph et al. [Adol 01], Song et al. [Song 07] and Gao et al. [Gao 09] who aimed at combining domain knowledge with a CAD system are discussed in more detail in Chapter 2.

1.4 Main Contributions

The main contribution of this thesis is to design and implement a system for automatic design of medical prostheses, which was validated using a real world application and data. Finally, it was successfully introduced to the production environment of a major customized hearing aid manufacturer.

- We developed an automation framework: Expert System for Automatic Modeling (ESAM), consisting of a CAD modeling software, a feature recognition unit and a rule-based expert system [Sick 09].

- The development involved the identification of the necessary anatomical features and the creation of an adequate knowledge representation language to store the expertise in design knowledge extracted from domain experts.

- The automation framework was evaluated using real world data provided by an industrial partner, eventually resulting in the deployment of the developed framework in production [Sick 11a].

- We investigated techniques to improve the detection of anatomically important features on open 2-D manifolds using an adapted iterative closed point (ICP) algorithm in combination with clustering techniques [Sick 10a, Sick 11b].

- We explored machine learning methods to automatically or semi-automatically enhance or extend the knowledge base of our rule-based system. For example, genetic programming was applied as a rule learning technique [Sick 10b].

1.5 Thesis Overview

The thesis is divided into three main parts. Part 1 contains the introductory chapters 1 and 2. In Chapter 2, an introduction into the topic of medical prostheses design is given. In the beginning, the terminology of the field is defined. This is followed by a brief history and overview of medical prostheses. The chapter concludes with a discussion of prostheses types, which would benefit from automation provided by our framework.

Part 2 of the thesis is comprised of chapters 3 to 6. They describe the used methodology and algorithms. In chapters 3 to 5 the basic components of the automation framework are developed. They are wrapped together to our framework in Chapter 6. In particular, Chapter 3 introduces the features used in the framework, followed by their detection algorithms. Afterward, developed extensions for the improvement of feature robustness and accuracy are presented. Chapter 4 describes the developed rule-based expert system. Primarily the developed grammar for storing the rules in the knowledge base is introduced. The chapter concludes on how the rule base was acquired and preliminary experiments showing the suitability of the developed ES for the intended design task. Chapter 5 presents the learning techniques applied to enhance the automation framework. The usage of genetic programming for rule learning is explained, followed by a semi-automatic method using real world manufacturing data provided by an industrial partner. In Chapter 6 the methods introduced earlier are assembled yielding the automation framework (ESAM). Furthermore, the so far achieved results are presented and discussed including a brief comparison with related work.

The last part of the thesis consists of Chapter 7. It concludes the thesis with a summary and an outlook.

Chapter 2

Background on Medical Prostheses

The variety of medical prostheses is immense. They range from little stents inserted in coronary arteries to full leg replacements. In the following sections, the necessary terminology is introduced, followed by a brief historical overview of medical prostheses development. After that the different types of medical prostheses are discussed. The types which are applicable for our envisioned automation system are highlighted and discussed in more detail. The chapter concludes with a description of the state-of-the-art of prostheses manufacturing.

2.1 Terminology

The following terms and definitions are primarily taken from [Schu 94, Reut 05, Psch 07, Mede 10].

A *prosthesis* is an artificial substitute or replacement of a part of the body such as a tooth, a leg, an eye or a facial bone. It is designed for functional or cosmetic reasons or both. A prosthesis may be removable, as in the case of most prosthetic legs or breasts. These removable prostheses are commonly called *exoprostheses.*

Other types of prosthetic devices are permanently implanted, like artificial hips, testicles or craniofacial restorations. These kinds of medical prostheses generally are called *implants* or *endoprostheses.* Implants are man-made devices, in contrast to transplants, which are biomedical tissues.

Between exo- and endoprostheses, a third group exists, which is called *open implants* or *open prostheses.* They are characterized by penetrating the skin of the patient like dental implants or cochlear implants.

Furthermore, current research on prosthetic devices resulted in experimental prostheses which have been integrated with body tissues, including the nervous system. Therefore, they are summarized with the term *neuroprostheses.*

2.1.1 Necessary Conditions for Automation Framework

In this work, we focus on certain types of prostheses. A medical prostheses is suitable for our automation framework, if the following conditions are fulfilled:

1. A template of the original structure may be acquired and is available as a triangular mesh.

2. The template may be manipulated in a CAD software by a human expert following specific rules.

3. The prosthesis or its outer surface may be produced with a rapid prototyping technique.

4. The manufactured shape of the prosthesis should replicate the shape of the original structure as accurately as possible.

In Section 2.3, these criteria are evaluated in order to filter out the interesting prostheses types.

2.2 Historical Roots

A wooden toe found at an Egyptian mummy gives evidence, among others that amputation as means of medical treatment was applied and prosthetic devices were used already 4000 years ago [Dupr 09]. Naturally the first medical prostheses were usually exoprostheses made by craftsmen with focus on cosmetics and minimal functionality [Nerl 00].

Since then, the manufacturing process of medical prostheses changed dramatically. Key points for research on medical prostheses design were usually driven during wartimes for tactical reasons and to provide better care for the wounded [Herr 03, Lafe 10]. A good example for these changes are leg prostheses. The very first were peg legs, which had been connected to an amputation stump with leather straps, see Figure 2.1(a). In the Middle Ages, the first leg prostheses with advanced functionality were developed. A prominent figure was the French army barber-surgeon Ambroise Paré, who not only invented new amputation techniques but also designed superior upper- and lower-limb prostheses [Lafe 10]. These artificial legs already featured a spring mounted foot and a flexible knee joint like the one in Figure 2.1(b). The advancements in amputation techniques allowed to produce better fitting prostheses, which increased their usability [Sach 99].

In the Modern Age, the need for prostheses increased due to wars like the American Civil War, the Franco-Prussian War and the World Wars. For example, after World War II, approximately 15.000 US soldiers and more than 1 million people worldwide suffered an amputation [Lafe 10]. No longer it were only craftsmen, but mainly physicians, engineers and scientists, who worked on the development of prostheses. This trend was accelerated by governmental funding of prosthetic research, like the Artificial Limb Program initiated by the National Academy of Sciences after World War II. The programs purpose was to promote and coordinate scientific research on prosthetic devices [Comm 47, Lafe 10].

These historic events started the progress eventually resulting in nowadays computer-aided-design of medical prostheses, including finite element methods (FEM), which allows the manufacturing of implants, digital hearing aids or full leg replacements, see Figure 2.1(c). A modern leg prosthesis is an elaborate device, allowing complex movements and even competitive sports, see Figure 2.1(d).

Due to advances in medicine the prostheses migrated into the patient's body. Since the 1960s it is possible to replace hips or knee joints. Nowadays, prostheses or

(a) Simple peg leg, made of wood and connected to the amputee using leather straps, courtesy of Science Museum, London [Scie 10a].

(b) Example of a prosthetic leg featuring a flexible knee joint [Scie 10b].

(c) Example of a state-of-the-art full leg prosthesis, courtesy of Otto Bock HealthCare GmbH [Otto 09b].

(d) Leg prosthesis for competitive sports worn by Oscar Pistorius [Elva 10].

Figure 2.1: Evolution of prosthetic legs over time.

prosthetic devices are available for all parts of the human body, e. g. the esophagus, arteries, bones, inner ear, retina and even the heart can be replaced for a short amount of time [Herr 03, Yama 02, Jauh 04].

Also material science gave vital inputs for the improvement of medical prostheses. Instead of wood or steel, nowadays prostheses are often made of polyvinyl chloride

(PVC) or specifically manufactured compounds with high bio-compatibility, such as ceramics and titanium for open implants and endoprostheses [Herr 03]. Furthermore, modern prosthesis are build with injection molding [Wint 09] or rapid prototyping techniques [Gebh 07]. These techniques permit a fast and accurate production of prostheses while producing a visual appealing shape [Wint 09]. In addition, the need for intra-operatively implant modeling of donor bone is eliminated and it is possible to build physical models for operation planning [Eufi 95, Eufi 01, Sodi 05].

2.3 Types of Medical Prostheses

The definitions in Section 2.1 divided medical prostheses roughly in three major groups: exoprostheses, endoprostheses (implants) and open prostheses. We adopt this scheme by organizing the following sections accordingly.Neuroprostheses are not handled separately here, but are covered in the other sections as needed. The following medical prostheses classification shall give the reader an overview of the field, but does not claim to be exhaustive.

2.3.1 Exoprostheses

Exoprostheses are probably the most commonly known prostheses. Typically, they are replacements for lost limbs, like arms, legs and hands. For these kinds of prostheses the recovered functionality is often more important than the disguise of the disability. In other words: form follows function, as it can be seen in the *sport* legs of Oscar Pistorius in Figure 2.1(d).

Artificial Legs

Artificial legs are common and very successful exoprostheses. They are divided in two major groups, indicating if the prosthesis is attached below or above the knee. Transtibial prostheses replace a leg missing below the knee, whereas transfemoral prostheses include the knee [Schu 94]. Typically, both types are attached to the patient's leg residual (stump) either by using belts and cuffs or by suction, where the stump fits into a socket of the prosthesis, see Figure 2.1. A new possibility is osseointegration. In this case the prosthesis is attached directly to the bone of the residual leg and it would be valid to be considered as an open prosthesis, see Figure 2.2(a). Such an osseointegrated prosthesis is intended to assure a long-term fixation, improved limb control and less socket-caused skin disorders [Lafe 10]. However, patients have a higher risk of infection around the fixation component [Till 10].

Artificial Arms

Artificial arms, akin to the legs, are also divided in two groups. Transradial arm prostheses replace an arm below the elbow, whereas transhumeral prostheses include the elbow [Schu 94]. Again, using sockets in combination with straps is the typical attachment strategy [Lafe 10].

(a) Osseointegrated fixation for an artificial leg, courtesy of J. Tillander [Till 10].

(b) Behind-the-ear and in-the-ear hearing aid, courtesy of Siemens AG [Siem 10b].

(c) Cosmetic (passive) artificial hand, courtesy of Bock Healthcare GmbH [Otto 07].

(d) Functional artificial hand, courtesy of Bock Healthcare GmbH [Otto 09a].

Figure 2.2: Examples of different kinds of exoprostheses.

Artificial Hands

An artificial hand is the most complex part of an artificial arm. Simple hand prostheses are cosmetic (passive) ones more or less without functionality besides hiding the amputation, see Figure 2.2(c). In recent years, though, improving artificial hands became a field of high research interest. State-of-the-art are myoelectric devices based on electromyography [Cram 98]. The *myo* devices use sensors to detect electromyographic activity of muscle cells, which are translated into movements of the

prosthesis [Kost 81]. Typically, these hands allow one or two degrees of freedom and require the patient to undergo a long training. To reduce the training time and improve the control, Orabona et al. [Orab 09] proposed to provide a pre-trained model to the patient by applying model adaptations with support vector machines (SVM). Other works focus on improving grasping functionality of the hand only. Wiste et al. [Wist 09] for instance introduced a multi-functional hand, which supports eight different hand postures. Similar work was published by Jung et al. [Jung 08] who designed a hand with flexible tendon-driven fingers resulting in a hand with six degrees of freedom. They achieved this, while meeting the requirements of not exceeding the weight of a normal human hand ($\approx$ 400g).

Hearing Aids

Hearing aids (HA) are usually worn behind the ear or inserted into the ear canal, see Figure 2.2(b). In-the-ear (ITE) hearing aids have to be customized for each patient based on an impression of the outer ear including the ear canal (negative of the ear), see Figure 2.11(a). A customized device is designed in a CAD software. Finally, the designed shape is fabricated by a stereolithography apparatus (SLA) resulting in an exact replicate of the acquired negative ear shape [Gebh 07, Mast 05]. Hence, following the conditions defined in Section 2.1.1 customized hearing aid manufacturing belongs to the target group of our automation framework and is discussed in more detail in Section 2.4.4.

2.3.2 Open Prostheses

Open prostheses are less obvious from the outside than exoprostheses, but nevertheless visible. Typical examples are dentures, cochlear implants or cosmetic prosthetics (cosmeses).

Dentures and Dental Implants

Dentures and dental implants are available in various forms and types. Examples are removable or complete dentures, bridges or dental crowns. Dentures and dental crowns need to be customized to allow good fitting and functionality. To achieve this, individualized crowns are modeled based on teeth models [Stru 06]. On the one hand, they have to be correctly shaped to fulfill their function during chewing. On the other hand, they have to blend with the surrounding teeth in order to be visually appealing. These characteristics mark dental crown design as suitable for our automation framework. Therefore, it is discussed further in Section 2.4.3.

Cochlear Implants

Cochlear implants, in contrast to dental crowns, are customized internally by tuning the parameters of the components only. The external parts (microphone, speech processor and transmitter) and internal parts (receiver and stimulator) are off-the-shelf products, see Figure 2.3. The available models differentiate mainly in the number of electrodes which twine through the cochlea. The electrodes send impulses to the

(a) External part of a cochlear implant.

(b) Internal part of a cochlear implant.

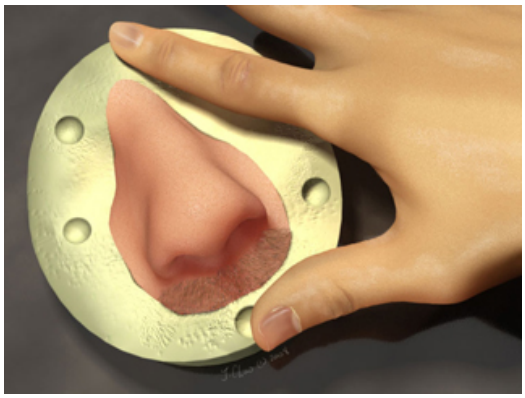
Figure 2.3: Internal and external parts of cochlear implants exemplified with the Nucleus-Freedom-System 2005, courtesy of Ernst et al. [Erns 09].

nerves in the scala tympani and then directly to the brain through the auditory nervous system. The manufacturers use between 16 to 22 electrodes and different stimulation techniques in their devices. Nevertheless, the precondition for a successful rehabilitation is the individual tuning of the speech processor [Erns 09]. Even today this is mostly done subjectively and requires a high amount of cooperation and concentration from both examiner and patient [Legr 07]. Since the cochlear implants are directly connected with the cochlea, they are also categorized as neuroprostheses.

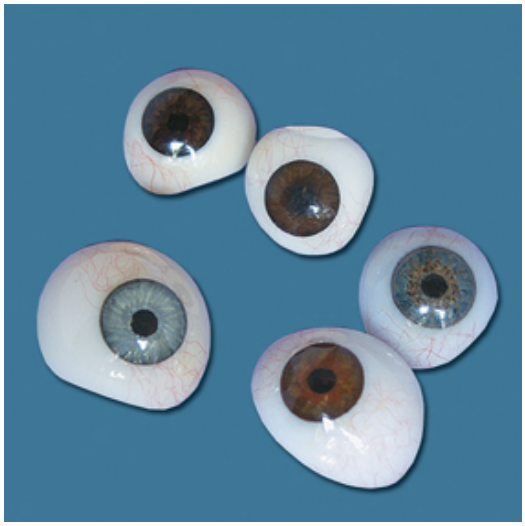
Cosmetic Prostheses

Cosmetic prostheses help patients to lead a normal life by disguising injuries and disfigurements caused by tumors or accidents. Cosmeses can be custom-made or ready-made in various sizes and for various body parts. Custom-made cosmeses have the advantage, among others, that they match the patient's skin color perfectly [Herr 03]. There are several methods to attach cosmeses to the human body, like the usage of an adhesive, suction, stretchable skin or skin sleeves [Psch 07]. Furthermore, often osseointegration is used to accurately and firmly attach cosmeses to a patient's body [Klei 01]. Since several attachment methods do not penetrate the skin of the patient, cosmeses are a good example for the existing ambiguity in classifying prostheses in exo-, endo- and open prostheses. In Figure 2.4 different kinds of cosmeses are displayed.

(a) Silicone cosmesis.



(b) Nose cosmesis.



(c) Eye cosmeses.

(d) Ear cosmesis.

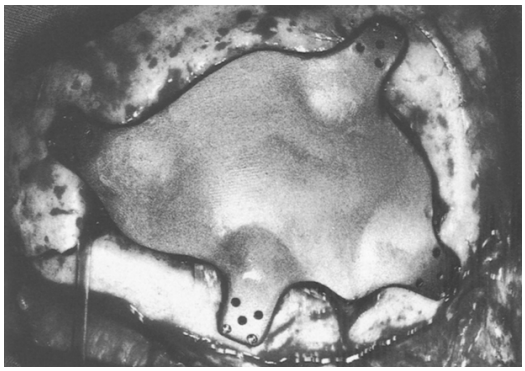
Figure 2.4: Examples of different kinds of cosmeses, courtesy of Dr. P. Ajitha.

2.3.3 Endoprostheses / Implants

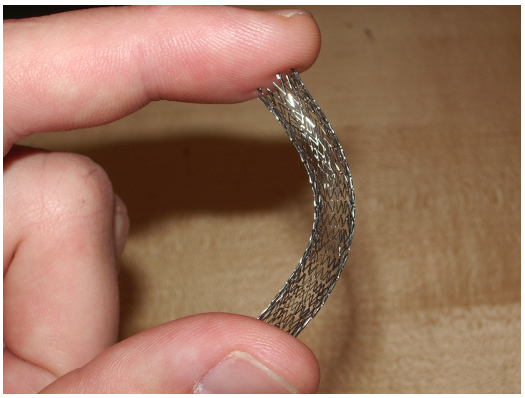
Implants are available for all kinds of purposes. In most cases they replace or support a biological structure broken due to age, accidents or a disease. Examples can be artificial hips, joints and orthopedic implants for stabilizing fractured bones.

Craniofacial and Maxillofacial Implants

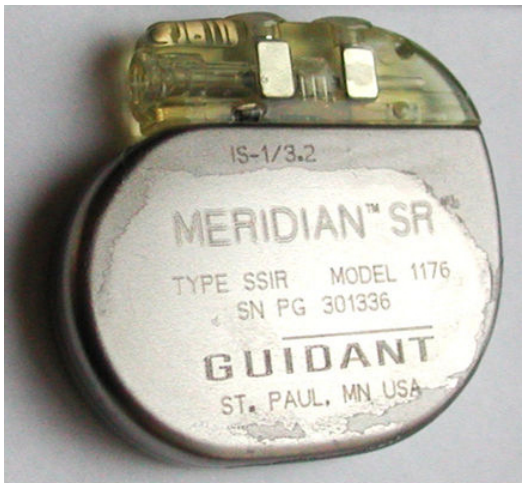
Craniofacial and maxillofacial implants are used in restoration surgery of the skull. These customized implants are used to correct bone defects due to tumor surgery or accidents in the cranial cavity or at the jaw (maxillofacial restoration). This is usually done by acquiring a CT-data set of the skull, correcting the defect in a CAD environment and finally producing the implant with a rapid prototyping technique [Eufi 95], see Figure 2.5(a). Therefore, this kind of implants is well suited for our automation framework and are discussed in more detail in Section 2.4.2.



(a) Inserted skull implant, courtesy of Eufinger et al. [Eufi 95].



(b) Stent for coronary arteries, courtesy of Benjamin Keck.



(c) Heart pacemaker, courtesy of J. Heuser [Heus 10].

(d) Artificial heart, courtesy of J. Nakashima [Naka 10].

Figure 2.5: Examples of different kinds of implants.

Stents

Stents are little tubes inserted into coronary arteries, vessels or the urethra to prevent or counteract a disease-induced, localized flow constriction [Reut 05], see Figure 2.5(b).

Artificial Pacemakers

Artificial pacemaker belong to implants, which contain electronics. A pacemaker monitors the heart's native electrical rhythm. In case of a missing heart beat it stimulates the ventricle of the heart with a short low voltage pulse [Reut 05], see Figure 2.5(c). In case of a serious heart disease or heart failure, it is possible to bridge the waiting time for a donor heart with an artificial heart [Jauh 04], see Figure 2.5(d). Since the number of donor hearts is much smaller than the number of waiting patients, current research focuses on developing long-term artificial hearts or heart supporting systems, like (bi-)ventricular assist systems [Yama 02]. The most recent ones are nonpulsatile pumps, which feature a compact size, may be driven without mechanical contact and may be used for 5 to 10 years [Yama 02].

2.4 Medical Prostheses Design: State-of-the-Art

In the previous sections, craniomaxillofacial implants, customized in-the-ear hearing aids and dental crowns were identified as possible application areas for our automation framework. In the following sections, the state-of-the-art of their design is introduced. Beforehand, a brief introduction to rapid prototyping is given.

2.4.1 Rapid Prototyping

Rapid prototyping (RP) encompasses the science of generative production processes and therefore is a technology. The term is used synonymously for applications of the technique like stereolithography (SL). It describes processes by which 3-D models are produced additively, by fitting or mounting volume elements together [Gebh 03]. Depending on the application, RP can be divided in four groups:

Solid imaging describes the production of concept prototypes, producing relatively rough, but cheap models, that nonetheless display the outer form and features of the final components very well.

Functional prototyping results are more precise and simulate one or more functionalities of the product and are thus more expensive.

Rapid tooling encompasses the production of positives and negatives like dies or molds for production prototypes.

Rapid manufacturing or rapid production is used for the production of products with serial character, e. g. customized hearing aid manufacturing [Gebh 03, Hopk 05, Gebh 07].

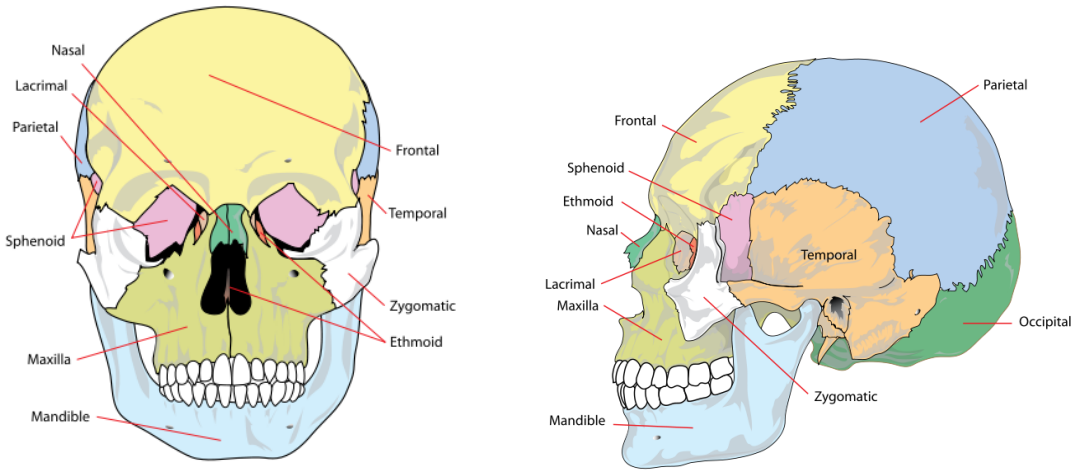
Stereolithography

Stereolithography is the most common rapid prototyping technique. It describes the process of hardening a liquid of monomer that is interspersed with suitable photoinhibitors by exposure to ultraviolet radiation. The radiation sets off a spontaneous polymerization yielding a solid polymer. In practice, the prototype model is sliced

yielding a set of cross sections. These cross sections are painted successively by a fine laser beam in a resin of liquid monomers. Thereby the necessary energy for solidification is generated. Stereolithography is sometimes also referred to as printing or cooking and often abbreviated as SLA (stereolithography apparatus) [Gebh 03, Gebh 07].

2.4.2 Craniomaxillofacial Implants

Craniomaxillofacial implants cover the anatomical areas of mouth, jaw, face, skull and associated structures [Reut 05], see Figure 2.6.



(a) Skull frontal view.

(b) Skull lateral view.

Figure 2.6: Anatomy of the human skull, courtesy of M. Ruiz Villarreal [Vill 10].

Cranioplasty

Cranioplasty is understood to be reconstructive surgery of large defects in the frontal, parietal, temporal, occipital and sphenoidal (neurocranial) areas of the skull [Psch 07]. Its major goal is to provide protection for the brain.

Using intraoperatively modeled implants has several limitations, e. g. the intraoperative environment itself, the necessary guessing of the patients skull curvature and the manual modeling of the implant. In addition, the availability of suitable implant materials is usually limited. These problems are even more severe for larger defect areas [Eufi 95, Eufi 01]. Altogether, there is a high chance that the implant will not perfectly fit the patient and may cause the need for follow-up surgery.

In 1995, an interdisciplinary research group at the Ruhr-University Bochum led by Harald Eufinger developed a workflow for preoperative individual skull implant manufacturing based on CAD/CAM technology [Eufi 95]. They applied computer-assisted prefabrication for customized titanium implants. It has the advantage that the implants are geometrically stable, can be obtained in the necessary quantity, the risk of infectious agents is eliminated and the second surgical field on the patient is avoided. Furthermore, due to the availability and accuracy of the prefabricated implant, the operation time is greatly reduced [Lee 02, Bibb 00]. In addition, it is

possible to do operation planning either in software or with fabricated physical models of the skull and the implant [Lee 03].

The steps in computer-assisted prefabrication are the following [Eufi 95]. First, a 3-D volume of the target area is acquired, e. g. by using a CT scanner. Second, the data is imported into a CAD system. With help of the system, an operator designs the customized implant using one of the following techniques: manual modeling, mirroring, NURBS (non-uniform rational basis splines) or database support. In the manual (free-form modeling) case an expert operator uses virtual tools to fill the target area with material followed by smoothing the contours of the implant [Hieu 10]. To apply the mirroring technique the patient needs a symmetrical anatomy and the opposing side must be intact. After mirroring the healthy side, the operator has to manually adjust and smooth the contours of the implant [Hieu 02, Hier 06]. The application of NURBS uses the assumption that an exact description of the skull surface is not necessary. It has the advantage that the anatomy may be asymmetrical and no healthy mirror is necessary. Furthermore, the resulting implant is very smooth. Hence, NURBS works best on smooth areas, e. g. frontal and parietal regions of the skull [Eufi 01, Hieu 02, Hieu 10]. The fourth technique is the only one utilizing stored and structured knowledge. It is based on searching for a similar, healthy skull in a skull database. This is especially useful for patients with asymmetric anatomy or multiple defects. The healthy skull is used as a template, which usually requires manual post processing to register is with the defect skull [Hieu 02, Hier 06].

The last step in the computer-assisted prefabrication is the transfer of the implant data to a CAM system. Typically, the system is either a computer numeric controlled (CNC) milling or a SLA. The former mills a titanium implant and the latter produces a plastic implant usually coated with titanium afterward [Eufi 95].

(a) SLA model of the defect jaw and titanium prosthesis.

(b) Implant filled with bone marrow and covered with cortical bone.

Figure 2.7: Example of a customized jaw implant, courtesy of Singare et al. [Sing 04].

Maxillofacial restoration surgery is similar to cranioplasty. Usually mirroring is used as a basis for the implant. The mirrored template is shifted and rotated to allow normal movement of the jaw [Sing 04]. However, maxillofacial implants have different requirements for load resistance and attachment procedure than craniofacial implants. Since the implant has to resist the chewing motion and force, FEMs are

applied to ensure sufficient stability of the implant [Dong 05, Futt 98]. This includes the placing and number of screws applied to hold the implant. Another difference is, that the (titanium) implant is filled or covered with bone marrow and cortical bone to achieve better integration of the implant and the bone, see Figure 2.7. Nevertheless, a study by Mehta et al. [Meht 04] concludes that plates and prefabricated implants should only be used if a vascularized osseous free tissue transfer is not possible. So far, the later is the preferred reconstructive modality and it has shown excellent long-term esthetic and functional outcomes, whereas reconstructions using plates have a higher probability of long-term problems [Wei 03].

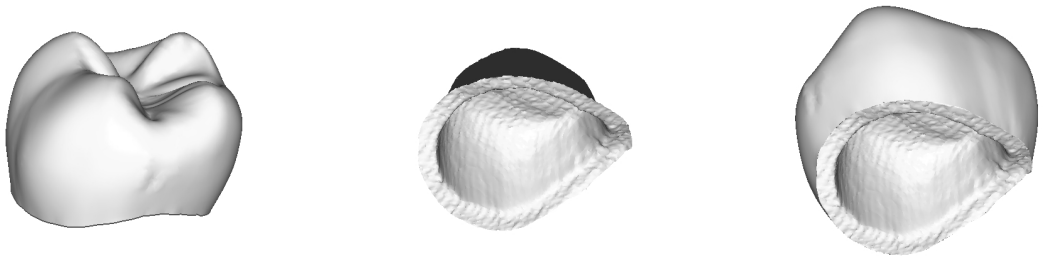
In spite of the conclusions by Mehta et al., for maxillofacial restorations, we believe that eventually prefabricated customized implants will be state-of-the-art in both applications due to the benefits named in the beginning. Hence, to avoid the dependency on skill and expertise of a CAD operator it would be helpful to apply an automation system for implant design.

2.4.3 Dental Crowns

In a modern dentist's office CAD and CAM are common tools for dental restorations [Morm 06]. They are used in various situations, e. g. to design dentures and dental crowns as well as to plan and support operations [Luet 11].

For us, dental crowns are of special interest, since they have to be customized for every patient. A typical treatment at a dentist's office consists of the following steps [Stru 06]: data capture, restoration design and restoration fabrication.

Different system for data acquisition are employed. An intraoral digital 3-D scanner is used by the CEREC system developed by Mörmann et al. [Morm 06]. Other systems utilize mechanical or optical digitizers in combination with dental imprints [Stru 06].



(a) External shape of a designed dental crown.

(b) Internal shape of a dental crown – acquired from the impression of prepared tooth or a dental implant.

(c) External and internal shape merged to the final crown design.

Figure 2.8: Example of customized dental crown design, courtesy of Adolph et al. [Adol 01].

The available commercial systems for dental crown design support the user by providing teeth databases as a design basis [Stru 06]. The provided teeth models are deformable and relatively simple. In order to design an individual prosthesis, the dentist has to adjust the model extensively [Adol 01]. The design of such an individual crown is constraint by the proximal and the opposing teeth. It does not need to be an exact replicate of the original tooth [Song 07]. The dentist has to ensure that the tooth model is adjusted in size to match the proximal teeth and that possible occlusion is taken into account. Occlusion is the contact between maxillary (upper jaw bone) and mandibular (lower jaw bone) teeth [Song 07]. Hence, the profile of the designed tooth crown, in particular the cups and the fossae, have to be adjusted according to the opposing teeth, see Figure 2.9. The adjustments are typically done manually with help of CAD tools or free form techniques and may be supported by precomputed profiles of the opposing teeth [Adol 01, Lee 08]. However, false occlusion may still occur and the dentist may have to grind the crown manually [Song 07]. The final step of dental crown design is to adjust the internal surface such that it fits precisely on the prepared tooth or dental implant of the patient, see Figure 2.8. Consequently, the quality of the result again depends on skill and experience of the operator [Song 07].

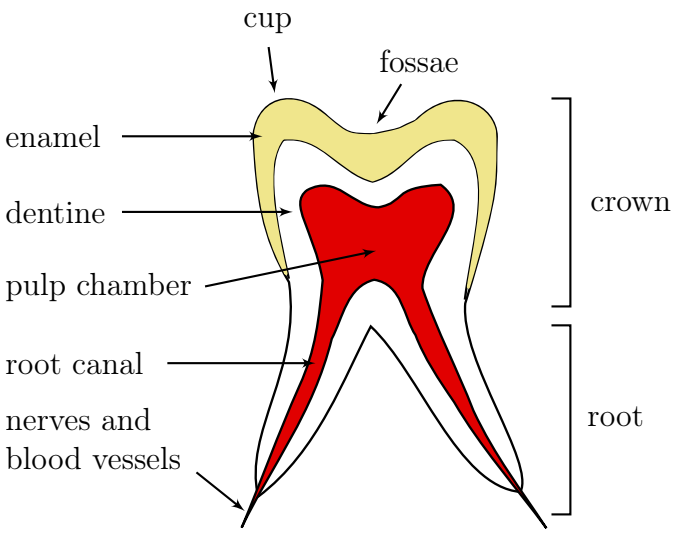


Figure 2.9: Anatomy of the tooth.

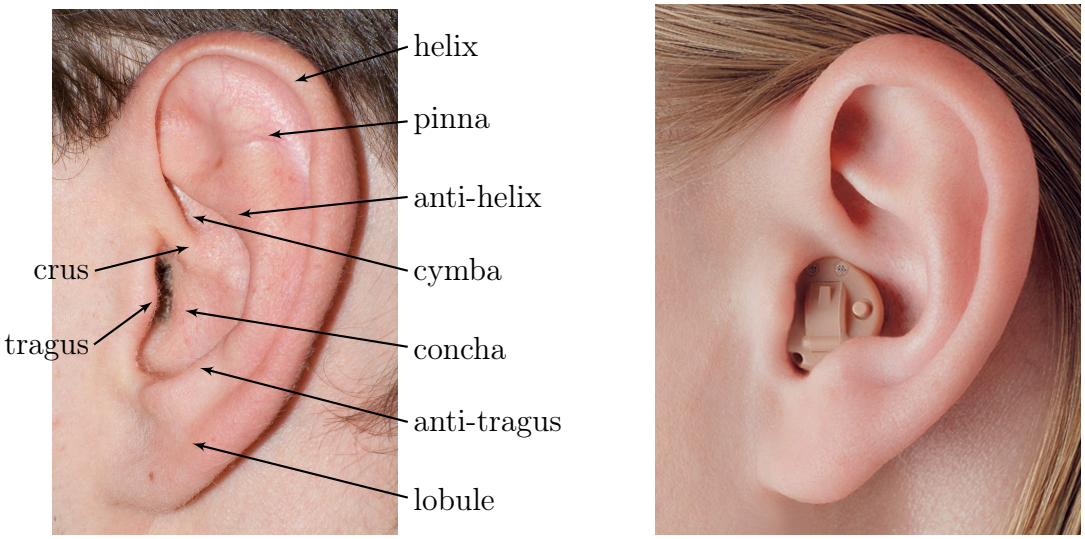
Afterward, the design result is transferred to a CAM machine. CAM in dental restoration is typically applying either a subtractive or an additive method [Stru 06, Gebh 07]. The subtractive method uses burs, diamonds and diamond disks to cut the crown from a prefabricated block. It has the advantage of an immediate result, but at the expense of material being wasted. As an additive technique selective laser sintering may be used for ceramic as well as metal restorations [Zhu 09]. Thus, no material is wasted and very complex shapes are possible. Disadvantage of this approach is, that the crowns have to be enlarged beforehand, because during sintering the crown shrinks [Stru 06]. The shrinkage is sensitive to the used material and may need marginal adaptations.

The current CAD/CAM workflow for dental restoration has already a high degree of automation, because the need for customization is minimized by the usage of teeth databases and the existence of computer-aided navigation and surgery tools like Robodent [Meye 03]. This knowledge based approach offers design support in a very limited way. However, standard teeth crowns pulled from a database need only little adjustments for good fitting and functionality. Furthermore, the esthetic aspect does not require a perfect replicate of the original tooth to be satisfying. In some cases it even is disadvantageous to replicate the original tooth, due to deformations or other esthetic issues.

In spite of the already high degree of automation, dental restoration still requires manual adjustments [Morm 06, Song 07, Lee 08, Zhu 09]. Hence, it would be beneficial to automate adjustments for size and occlusion with help of an automation framework.

2.4.4 Customized Hearing Aids

So far craniomaxillofacial implants and dental crowns were presented as possible application scenarios for the automation framework. Customized hearing aids are another interesting and application area, because compared to the previous examples they contain additional components, which have to be placed to fit the anatomical constrains and support the amplification required.



(a) Outer left ear of the author with annotation of the most relevant anatomical features for customized HA design.

(b) Ear with fitted in-the-ear hearing aid, courtesy of Siemens AG [Siem 10a].

Figure 2.10: Anatomy of the human ear.

Hearing aids are characterized as body aids, behind-the-ear (BTE) or in-the-ear (ITE) devices [Dill 01].We are only interested in the latter ones, because the former ones are not customized in respect to their shape. The outer shape, often referred to as *shell*, of an ITE is completely customized for every patient and every ear. This is necessary, because a perfect fit in the concha (see Figure 2.10(a)) and ear canal

(a) Silicone impression of the authors left ear.

(b) Viper^{TM} SLA System by 3D systems, courtesy of 3D systems [3Dsy 10].

(c) Konica Minolta Sailor III 3-D laser scanner for ear impressions, courtesy of Konica Minolta Sensing [Koni 10].

Figure 2.11: Hearing aid manufacturing.

is mandatory for good wearing comfort and functionality. ITEs are categorized depending on their physical size in three classes [Dill 01]: in-the-ear (ITE), in-the-canal (ITC) and completely-in-the-Canal (CIC) devices. ITEs are the largest ones and provide the strongest amplification with the drawback of being the most visible ones (Figure 2.12(a)). ITCs are smaller than ITEs and therefore less visible, but also providing less amplification, see Figures 2.10(b) and 2.12(b). From the cosmetic point of view, CICs are the most favorable devices, because they are almost invisible. As a consequence, they are suitable for mild hearing losses only, see Figure 2.12(c).

Customized hearing aid manufacturing is a delicate, complex and time consuming process [Pirz 06]. In the last years, most of the hearing aid manufactures introduced CAD/CAM into their manufacturing workflow [Mast 05]. However, like the manual process before, also the digital one needs highly trained and skilled experts to build hearing aids which are small and fit the patient's ear comfortably. Simi-

lar to craniomaxillofacial implants and dental crowns, the workflow consists of four stages [Gao 09]: impression taking and scanning, virtual design, shell building and assembly.

Impression taking and scanning: To acquire an ear impression, the audiologist fills a malleable material into the ear of the patient. The material is allowed to settle resulting in an ear negative like shown in Figure 2.11(a). Subsequently the impression is scanned either at the audiologist's office or at the manufacturer using a 3-D laser scanner [Togn 05, 3Sha 10], see Figure 2.11(c). The result is a 3-D point cloud, which is triangulated to reconstruct a mesh that represents the surface of the outer ear using a CAD software.

(a) ITE shell. (b) ITC shell. (c) CIC shell.

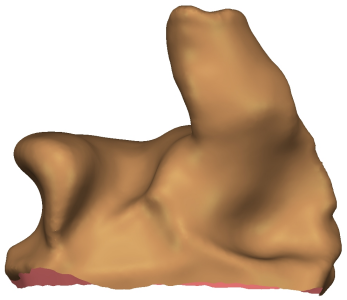
Figure 2.12: Examples of SLA printed shells showing the three major device types: ITE, ITC and CIC.

Virtual Design: The virtual design takes place in a specialized CAD software. The CAD software provides an expert designer with the same functionality he or she would use in the manual process, like grinding, smoothing or cutting. In addition the designer can apply localized and global offsets, integrate a shell wall and a ventilation tube[1] (vent). Furthermore the virtual placement of electronic components is supported. The electronic components include, but are not limited to, a faceplate (consisting of a battery, a microphone and a processing chip) and a receiver, which does the amplification [Dill 01]. Such a virtual component placement ensures that the selected electronics will fit in the hearing aid shell during the actual physical assembly. In Figure 2.13, a shortened example of the virtual design process is shown.

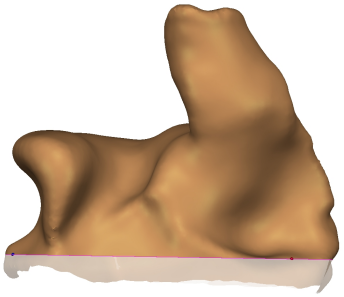
Shell Building: The shell building, often referred to as printing or cocking shells, is done with an SLA [Mast 05, Gebh 07], see Figure 2.11(b) and Section 2.4.1. Using an SLA, the hearing aid shells are printed in a pool of liquid material along with supporting pillars. These are necessary, because otherwise deformations of the still soft shell would be possible. In case of hearing aids, the SLA process takes around 4 - 8 hours for about 80 - 100 shells [Gao 09, Paul 04a]. Finally, the supporting pillars are removed and the shells are ready for the assembly, see Figure 2.12.

Hearing aid assembly: During the physical assembly of the device the components are placed, according to the virtual placement, inside the shell. Once an operator

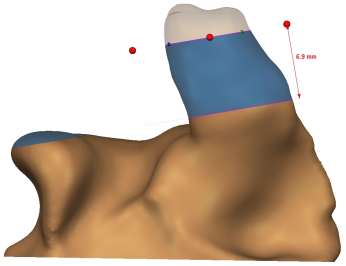
[1]The vent is required because an HA hermetically closes the ear, thereby causing pressure differences and acoustic feedback. The integration of a vent reduces these negative effects [Sick 07]. Vents come in various sizes and shapes to accommodate different feedback properties, and their selection and placement is crucial for proper functioning of the device [Azer 10].



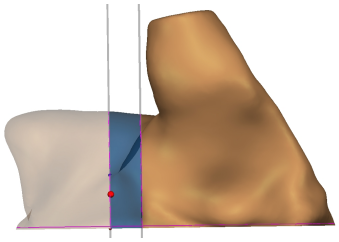
(a) Triangulated right ear impression (ear negative).



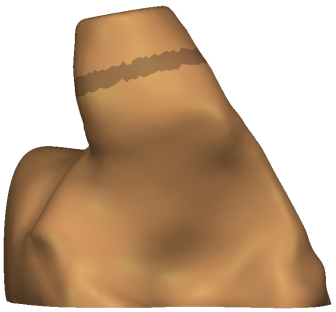
(b) Detailing operation (open cut) to remove excess material at the bottom of the impression.



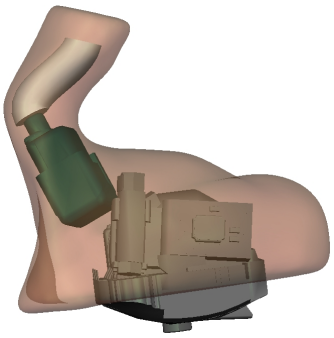
(c) Tapering of the canal tip to allow easy and painless insertion into the ear canal.



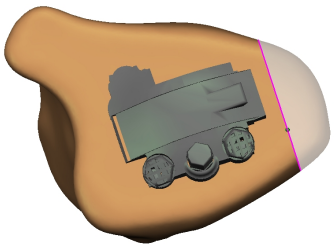
(d) In the case of ITC and CIC devices the cymba is removed with a cut through the valley of the crus.



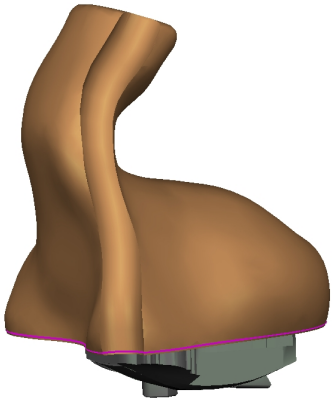
(e) To achieve a tighter fit between first and second bend, a feedback seal (local offset) is applied.



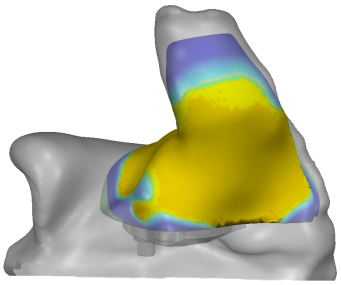
(f) Components like receiver (upper component) and faceplate (lower component) are placed as tight as possible.



(g) For a better visual appearance the device is made as small as possible by removing material, which does not influence the fitting.



(h) For feedback reduction and wearing comfort a ventilation tube is integrated. Usually this is a round tube inside the shell, but may also be a trench like shown in this figure.



(i) Final shell overlaid with the original impression including a colored distance map (blue indicates removed material, yellow indicates a positive offset).

Figure 2.13: Brief overview of the customized HA design workflow.

fitted all components, the device is lacquered and tested. After this, the device is

shipped to the audiologist who customizes the settings of the electric components according to the patient's need.

2.5 Conclusions

In this chapter, we gave an overview about the terms used in the field of medical prostheses and rapid prototyping along with a brief historical overview. The types of medical prostheses: exoprostheses, open prostheses, endoprostheses and neuroprostheses were introduced using concrete examples. Thereby, examples of medical prostheses that would benefit from an intelligent design system were identified: craniomaxillofacial implants, dental crowns and customized hearing aids. The expected benefits are primarily: an improved consistency of the design, a possibility to accurately reproduce a design and a reduced design time.

In the literature only little work was conducted on integration of intelligent systems and CAD tools. Mostly knowledge was included in form of templates or template databases, e. g. tooth databases. In the following chapters of the thesis, the framework is introduced and developed with help of the customized hearing aid scenario. The motivation for this is twofold. First, customized hearing aids are challenging to design, because various constraints have to be fulfilled and so far no satisfactory automation exists. Second, the remaining possibilities for automation in case of craniomaxillofacial and dental crown design are limited. It is expected, that an automation framework suitable for customized hearing aids can be adapted for these cases as well.

Chapter 3

Detection of Anatomical Features on Organic Shapes

ESAM can be customized for the application in mind in two ways: definition of the features extracted from the source shape, and definition of rules to describe the necessary process steps to acquire the target shape, given the source shape and the associated features. In this chapter the former is discussed, while the latter is subject of the following chapter. First, the anatomical features of ears, in particular of ear impressions are introduced and defined. After that, two methods for feature detection on ear impressions are discussed. The first one, a surface analyzing method, is based on collaborative work with Baloch et al. He first published it at MICCAI 2010 [Balo 10a], and we extended it together and published the results in the Journal of Computer-Aided Design [Sick 11a]. The algorithms are introduced in Section 3.2. The second method, based on registration and clustering, is discussed in Section 3.3 and was presented at BVM 2011 [Sick 11b]. The chapter concludes with an experimental section including a comparison of both methods.

3.1 Anatomical Features of Ear Impressions

The basic anatomical features of ears, based on the work of Henry Gray [Gray 18], were already presented in Figure 2.10(a) on page 23. They identify areas on the ear, which can be expressed as peaks, concavities, depressions, elbows, ridges or as combinations of these properties.

A typical ear impression is characterized by a long spindle shaped canal that fits deep in the outer ear and a base that sits in the external ear, where the two are separated by the canal aperture. An HA shell is held in place like clamps by tragus and anti-tragus along the bottom of the ear. These features uniquely determine the side (left or right ear) of the shell. On the top, the shell is held by the narrow helix sandwiched between the anti-helix and the protruded crus, see Figure 2.10. Both of these features are distinctive among individuals in terms of their shape, size and depth. The so-called concha characterizes the depth of the exterior ear, offering a relatively large bowl area to hold electronics inside an HA.

In addition to these anatomically important features, we identified additional features that completely describe the HA manufacturing process. Collectively, these fea-

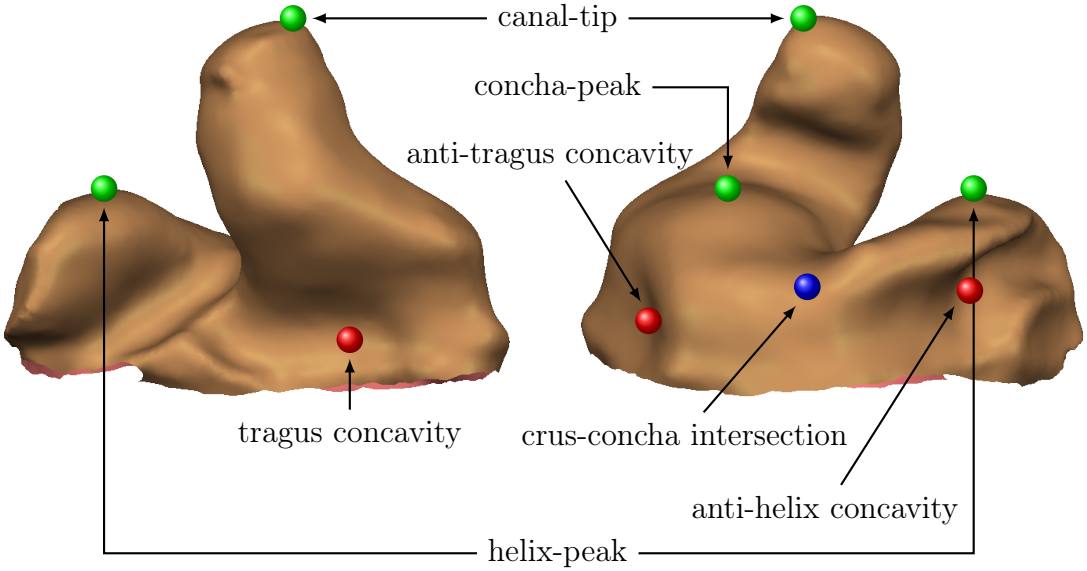


Figure 3.1: Concavity and peak features are typically represented by the most prominent point, e. g. according to a height function.

tures capture the structure of an ear in the Canonical Ear Signature (CES) [Balo 10a]. The CES features are listed in Table 3.1 and annotated in Figures 3.1 to 3.4.

Point Features

Point features are mostly peak or concavity features, which can be represented as 3-D points:

$$\boldsymbol{p} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}, \quad \boldsymbol{p} \in \mathbb{R}^3 \tag{3.1}$$

$$F_{\text{fp}} = \{\boldsymbol{p}_0, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N\}, \quad N \in \mathbb{N}. \tag{3.2}$$

The set of feature points F_{fp} contains all detected feature points. It may be split in subsets, such as F_{peak} for peak feature points and F_{concav} for concavity feature points. In most cases, the center point of the concavity or peak represents the required information. Hence, a point representation is sufficient. Furthermore, most of the features, which are defined by multiple other features are represented as points. For example, the canal-concha intersection is defined as the intersection between the inter-tragal notch ridge and a geodesic running from the concha-peak to the tragus concavity. Examples are shown in Figure 3.1.

Feature class	Feature name	Feature type
peak	concha-peak ⋆	point
	helix-peak ⋆	point
	canal-tip	point
concavity	tragus concavity ⋆	point
	tragus-area	area
	crus concavity	point
	anti-tragus concavity ⋆	point
	anti-helix concavity ⋆	point
elbow	first bend ⋆	plane
	second bend ⋆	plane
	aperture plane	plane
ridge	inter-tragal notch ridge	curve
	crus-ridge	curve
combinations	centerline	curve
	crus-valley plane ⋆	plane
	crus-valley area	area
	low of aperture ⋆	point
	inter-tragal notch top	point
	inter-tragal notch bottom	point
	crus-ridge top	point
	crus-ridge bottom	point
	feedback seal	area
	canal-tip area	area
	crus-concha intersection ⋆	point
	canal-concha intersection ⋆	point
	canal-crus intersection ⋆	point

Table 3.1: List of features composing the Canonical Ear Signature [Balo 10a]. The feature class indicates the appearance on the ear impression. The feature type defines how the detected feature is represented in our system. Concavities for example may be represented by the center point of the concavity or the whole area. Features marked with a ⋆ are considered as core features.

Plane Features

Plane features are typically high level features composed of several other features. The planes are represented by a set of 4-D points:

$$\boldsymbol{h} = \begin{pmatrix} n_x \\ n_y \\ n_z \\ -c \end{pmatrix}, \quad \boldsymbol{h} \in \mathbb{R}^4, \text{where} \tag{3.3}$$

$$c = \boldsymbol{n} \cdot \boldsymbol{p}, \quad \boldsymbol{n}, \boldsymbol{p} \in \mathbb{R}^3. \tag{3.4}$$

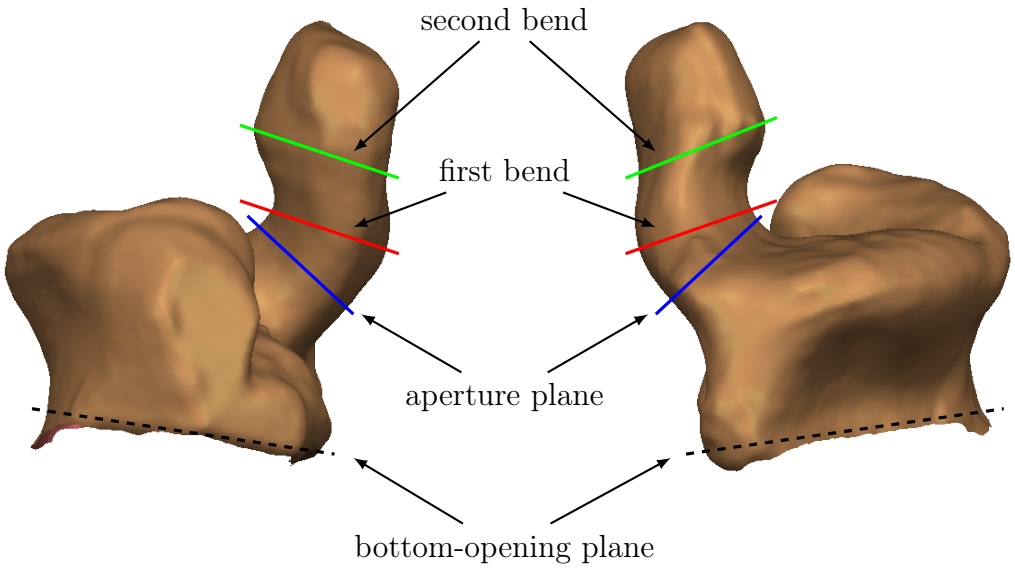


Figure 3.2: Examples of plane features. The aperture plane, first bend and second bend plane features are defined by the elbow like bends in the ear canal. In contrast to that, the bottom-opening plane is derived from the rough opening contour of the mesh.

In Eq. 3.4, the vector $\boldsymbol{n}$ denotes the normal of the plane and vector $\boldsymbol{p}$ a point in the plane. The set of feature planes is denoted with F_{elbow}:

$$F_{\text{elbow}} = \{\boldsymbol{h}_0, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_N\}, \quad N \in \mathbb{N}. \tag{3.5}$$

Examples are the prominent *elbow* like features, which may be present on ear canals are first bend, second bend and aperture plane. Other plane features are the bottom-opening plane and the crus-valley plane. The former has no anatomical meaning and is a result of the data acquisition procedure. It is defined as the least squares plane of the bottom opening contour. The crus-valley plane is characterized by the depression which divides an ear impression in canal/concha and helix part. Examples are shown in Figures 3.2 and 3.27.

Curve or Ridge Features

Curve features, also referred to as ridge features, are represented by an ordered set of points forming a line or curve on the surface of an ear impression:

$$\begin{aligned} C &= [\boldsymbol{p}_0, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N], \quad \boldsymbol{p}_i \in \mathbb{R}^3 \wedge N \in \mathbb{N}, \\ F_{\text{ridge}} &= \{C_0, C_1, \ldots, C_N\}, \quad N \in \mathbb{N}. \end{aligned} \tag{3.6}$$

For our purposes most essential are the inter-tragal notch ridge and the crus-ridge. Both are shown in Figure 3.3. A special curve feature, which is not placed on the surface is the so-called centerline, which represents the skeleton of the ear canal (see Figure 3.17).

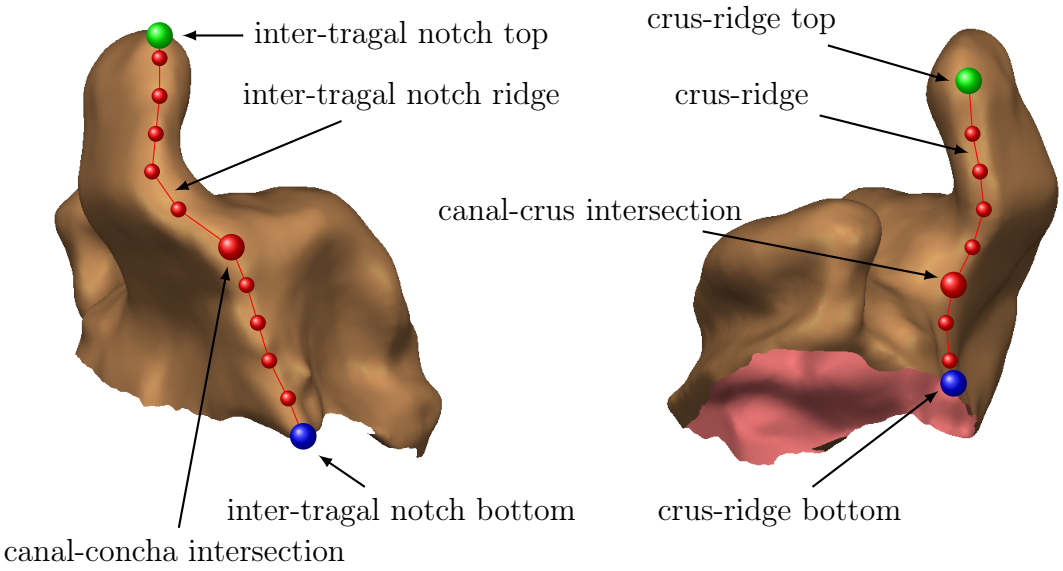


Figure 3.3: Examples of curve features and corresponding feature points.

Area Features

In some cases not only the center points of a concavity or valley are of interest, but the whole area needs to be taken into consideration. The representation is similar to the curve features:

$$\begin{aligned} A &= \{\boldsymbol{p}_0, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N\}, \quad \boldsymbol{p}_i \in \mathbb{R}^3 \wedge N \in \mathbb{N}, \\ F_{\text{area}} &= \{A_0, A_1, \ldots, A_N\}, \quad N \in \mathbb{N}. \end{aligned} \tag{3.7}$$

Examples are the tragus-area and the crus-valley area. The tragus-area is a simple feature solely defined by the depression around the tragus concavity. In contrast, the crus-valley area is composed of several features. It is bordered by the opening contour at the bottom of the scan, the helix on one side and the concha on the other side. Both areas are shown in Figure 3.4. Another area feature is the feedback seal, which is defined as a *band* on the concha side of the canal, approximately 3 mm above and parallel to the aperture plane (see Figure 2.13(e)).

3.2 Surface-based Feature Detection

3.2.1 Generic Feature Detection Algorithms

The approach of Baloch et al. [Balo 10a] to feature detection is based on generalization, where various anatomical features are characterized according to a reduced set of generic feature classes namely peaks F_{peak}, concavities F_{concav}, elbows F_{elbow}, ridges F_{ridge} and areas F_{area}. Although some anatomical features may not be represented by these geometric primitives, they may still be derived from the latter and are denoted as F_{comb}. F_{comb} contains point features $\in \mathbb{R}^3$, which are not expressed by peaks or concavities, but for example by an intersection of two ridges.

$$\mathcal{F} = \{F_{\text{peak}}, F_{\text{concav}}, F_{\text{elbow}}, F_{\text{ridge}}, F_{\text{area}}, F_{\text{comb}}\} \tag{3.8}$$

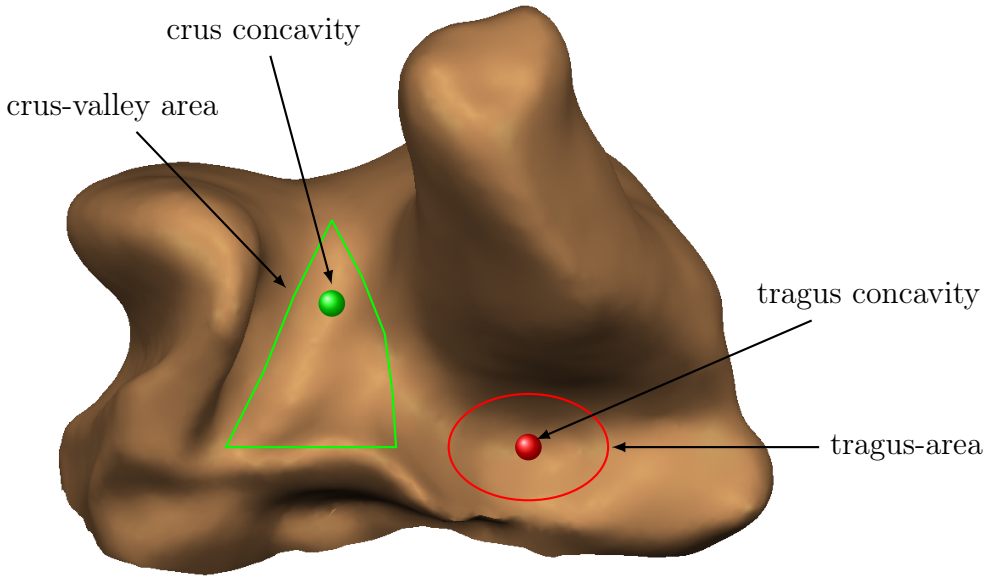


Figure 3.4: Examples of area features and associated feature points.

Algorithms are, therefore, constructed for the generic features, thereby later providing a foundation for the derived features.

Coordinate System

The initial step in our feature detection algorithm is to compute a common coordinate system for each ear impression. Therefore, first the bottom opening is analyzed yielding a least squares plane, in the following referred to as bottom-opening plane. The bottom-opening plane gives a basic orientation and allows the identification of the peak features, such as canal-tip, concha-peak and helix-peak, using the normal of the bottom-opening plane. With this information a coordinate system based on the bottom-opening plane, the concha-peak and helix-peak is calculated, see Figure 3.5.

Peak Detection

A peak point is a prominent topological landmark on a surface S. A surface S is represented by a triangle mesh of points $\in \mathbb{R}^3$. We detect the prominent landmarks via a height function $g : S \to \mathbb{R}$ that assigns to each point $\boldsymbol{p} \in S$ a value equal to its height, $g(\boldsymbol{p}) := g\left(\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right) = z$. For a non-degenerate surface, the critical points of g are the peaks, passes and pits of the surface. We, hence, use it for peak detection by analyzing the level sets of the height function for topological changes. By gradually increasing $g \in [0, H]$ in N steps, we find the intersections of the surface with the corresponding planes (see Figure 3.6). Intersections are subsequently analyzed for topological changes between two successive planes. If a change in topology is detected, the algorithm notices the existence of a critical level between them, and zooms in to analyze the surface with a larger N. The process is repeated until convergence to the peak point. For example, the intersection contour travels from the bottom-opening plane in direction of the ear canal. When the contour splits in two, then

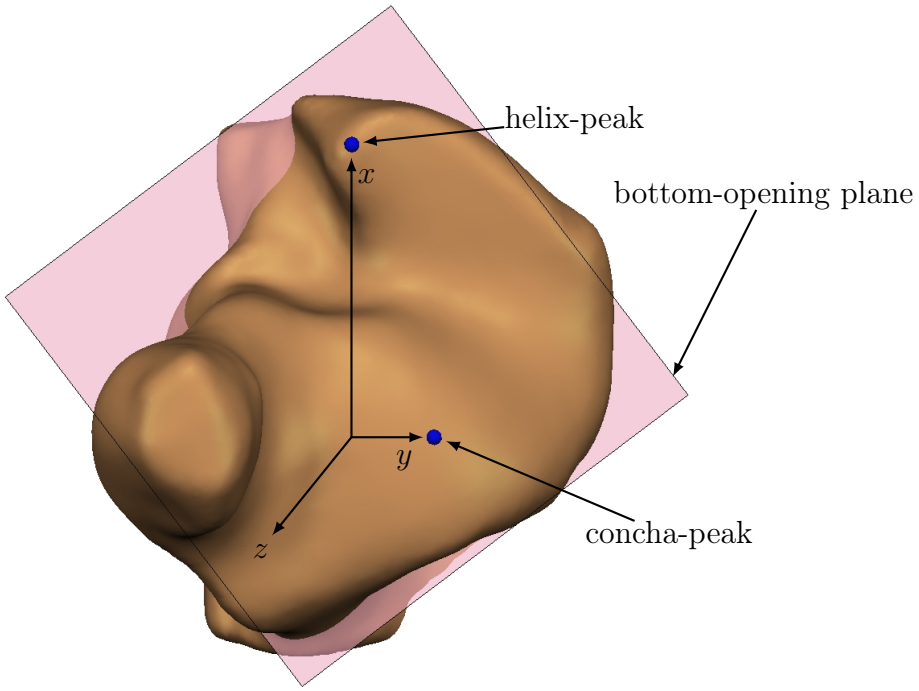


Figure 3.5: Ear impression with inscribed coordinate system. The x- and y-axis are in the bottom-opening plane. The z-axis is the normal of the bottom-opening plane.

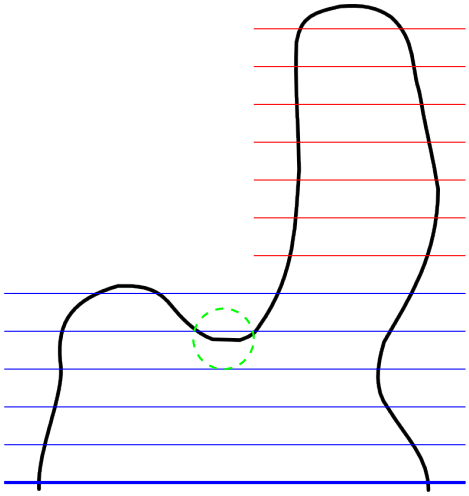
most likely one is intersecting with the ear canal and the other with the cymba (see Figure 3.6(a)). From there both are examined individually yielding the helix-peak and canal-tip feature point.

Concavity Detection

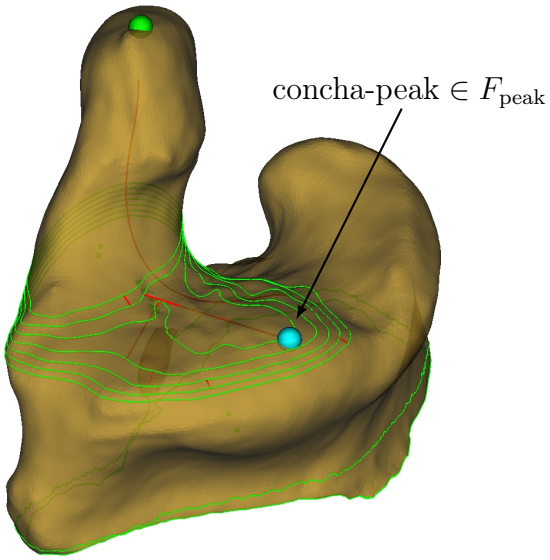
Concavities are marked by depressions on a surface. The algorithm for concavity detection utilizes orthogonal scans on a surface to generate a surface profile that is composed of the intersection contours. Individual contours are then analyzed for variations in signed curvature, where the negative sign identifies a concavity. First, a profile in one direction is considered, and subsections of contours with negative curvature are identified. For these subsections, the points of least curvature are found, with their average computed as a seed point. This seed point is corrected by a scan, orthogonal to the previous scan, shifting it toward the lowest curvature point, see Figure 3.7. Consequently, the seed point is pushed deeper in the valley. The process is repeated iteratively to achieve the absolute local minimum. The corrected seed point corresponds to the center of a concavity, and is employed for region growing based on negative curvature to determine the concave region.

Elbow Detection

Elbows are defined as bends in tubular surfaces, e. g. an ear canal. We, therefore, identify the points that exhibit high curvature followed by a point selection/rejection strategy that fits a plane along the elbow. First, the tubular part $S_{\text{ROI}} \subset S$ is



(a) Drawing of the intersection planes traveling from the bottom-opening plane (marked thick) in direction of the ear canal. The dashed circle highlights the topological change - the split of the intersection contour.



(b) Result of analyzing the intersection contours for the another peak feature, the concha-peak feature point. The intersection contours are shown in light green. The topological change is more difficult to analyze. Thus, this feature point is detected after the canal-tip and helix-peak are known.

Figure 3.6: Detection of peaks based on analyzing the level sets of a height function for topological changes.

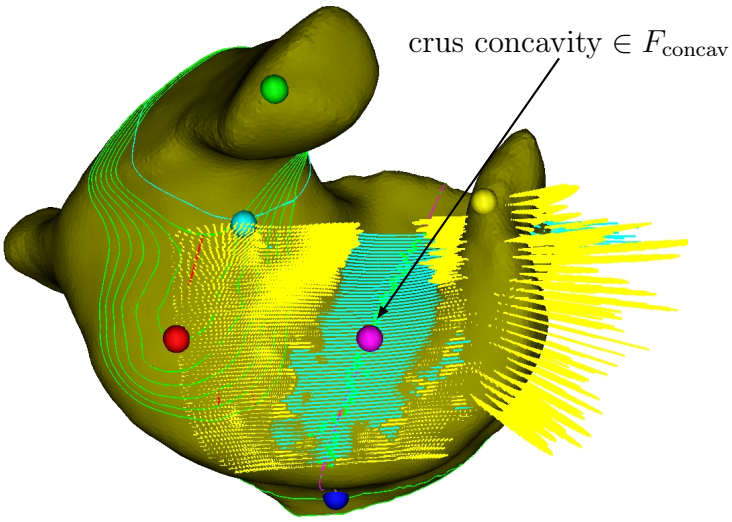


Figure 3.7: Detection of concavities based on analyzing the signed curvature of contours slicing the impression. In the picture, yellow markings indicate convex areas, while the light blue indicate concave areas.

identified and constrained by an upper and lower plane perpendicular to the centerline $\mathcal{L} \in F_{\text{ridge}}$. Both planes provide a contour and center point $C_{\text{up}}, C_{\text{low}}, \boldsymbol{p}_{\text{up}}, \boldsymbol{p}_{\text{low}}$.

The contours are cut like a cake in regular pieces as shown in Figure 3.8. The corresponding points of C_{up} and C_{low} ($c_i(0)$ and $c_i(1)$ in the Figure) together with the halfway center point $\boldsymbol{p}_{\text{half}}$ form a slicing plane. This plane is used to define the points on the radial contour $c(t) \in \mathbb{R}^3, t \in [0,1]$ between the corresponding points. The radial contours are parameterized, and are used to identify a set of points $Q = \boldsymbol{p}_0, \ldots, \boldsymbol{p}_N,\ N \in \mathbb{N}$ of maximal curvature along the radial contours (red boxes in the Figure). The point set Q is used to define a plane. However, not all high curvature points reliably represent an elbow, due to potential presence of bumps. The set Q is, hence, pruned via a point rejection strategy for plane fitting similar to the deterministic RANSAC method to increase the robustness of the elbow [Fisc 81]. The reduced set Q' is then used to define the feature plane.

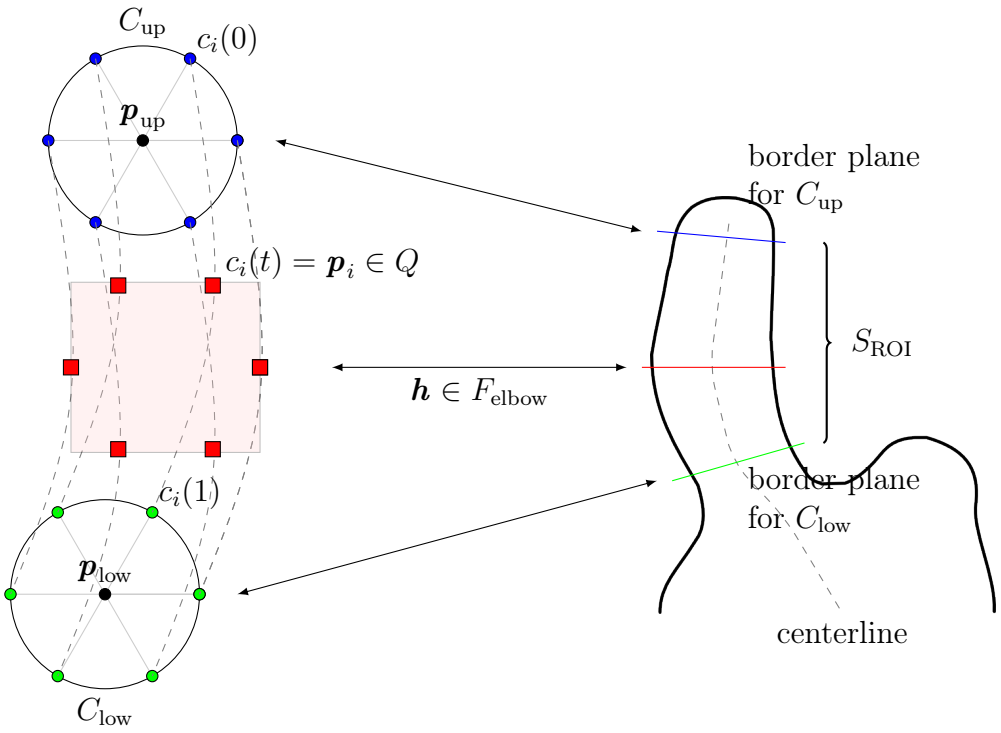


Figure 3.8: Schematic view of the elbow detection. The algorithm computes a contour profile along the canal in a region of interest S_{ROI}. The resulting radial contours $c_i(t)$ are analyzed and the maximum curvature points of each $c_i(t)$ are pooled in a set Q. Finally, these points are used to define an elbow plane after excluding outliers in Q.

Ridge Detection

A ridge $\pi : [0,1] \to S$ is defined as a geodesic on a surface S that passes through points of high curvature. Instead of using points of high curvature and then fitting a contour through them, we use an indirect approach. It involves first the detection of the starting and ending points, $\boldsymbol{p}_{\text{s}} \in \mathbb{R}^3$ and $\boldsymbol{p}_{\text{e}} \in \mathbb{R}^3$ respectively, of the ridge, followed by computation of a geodesic that connects them. To ensure that the geodesic passes

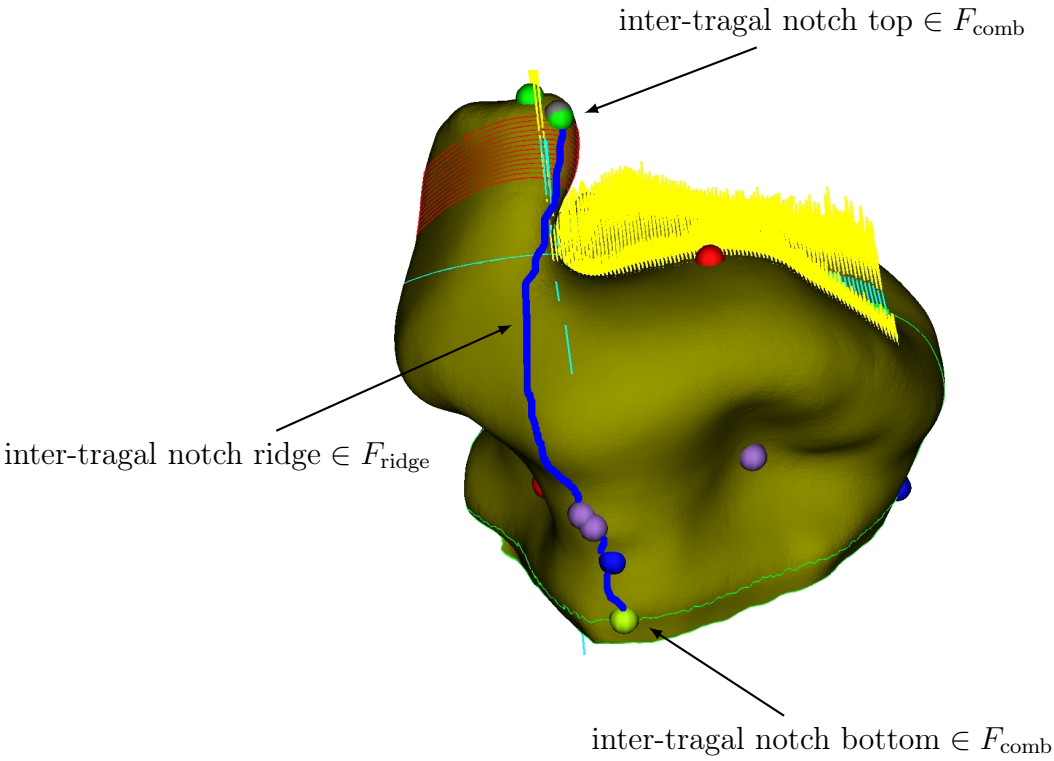


Figure 3.9: Detection of ridges based on previously detected feature points and a curvature constrained geodesic computation between these points. Note: The yellow slicing contours shown on the upper right of the impression are a result of the concavity detection of the crus region as shown in Figure 3.7.

through the ridge, we minimize the cost of going from $\boldsymbol{p}_{\text{s}}$ to $\boldsymbol{p}_{\text{e}}$, where the cost is defined as a weighted combination of the geodesic distance and surface curvature:

$$k(\pi) = \int_0^1 \omega(\kappa(t))\pi(t)dt, \tag{3.9}$$

where $\pi(0) = \boldsymbol{p}_{\text{s}}$ and $\pi(1) = \boldsymbol{p}_{\text{e}}$ and $\omega(\kappa(t)) \in \mathbb{R}$ is selected as a decreasing functional of curvature $\kappa(t) \in \mathbb{R}$. It is ensured that it favors only the positive mean curvature. Hence, the ridge is a minimizer of k, and may easily be computed for a triangulated mesh, through Dijkstra's algorithm with curvature weighted edge lengths [Dijk 59]. Consequently, the accuracy of the resulting ridge depends solely on the robust detection of its end points, and curvature weighting ensures that the geodesic passes through the high curvature ridge. An example for the detection of the inter-tragal notch ridge is given in Figure 3.9.

3.2.2 Surface-based Feature Detection for Ear Impressions

Feature detection on ear impressions is a challenging task. Despite a rough similarity, each ear impression is unique, see Figure 3.10 for examples.

Figure 3.10: Despite a rough similarity, each ear impression is unique. Even for one patient, left and right ear show differences in shape and size. The left column depicts the left and the middle column the right ear of one patient. In addition the scan quality differs strongly dependent on the skill of the audiologist yielding to different types of noise at top and bottom of the impression, as shown in the right column.

Adaptations for Ear Impressions

The aforementioned algorithms for generic features are modified slightly to adapt to the application under consideration. As mentioned earlier, the coordinate system is the basis for all other features. The coordinate system along with the associated features are the basis for a so-called feature dependency tree. Thereby, the bottom-opening plane is the root of the dependency tree.

For example, the ridge features depend on at least two features. The inter-tragal notch ridge and the crus-ridge require the detection of the end points of the corresponding ridges. For the top end points, the ellipticity or curvature analysis of cross-sectional contours of the canal is used. In case of highly elliptical contours, principal component analysis yields a good estimate of the ridge points. Otherwise, high curvature points on their spline representation are utilized. There is also an additional level of complexity. The canal is typically quite noisy near its tip, making it necessary to reject outlying contours through clustering techniques. Eventually, we get two points on the two ridges on the canal, one for the inter-tragal notch ridge, the other for the crus-ridge. For the bottom end points, the shell boundary is analyzed for convexity as well as its association with the top points. Eventually, geodesics are run from the ridge tops to the bottom points as described in Section 3.2.1 on page 37. The derived features usually depend on several other features and apply combinations of the basic algorithms. Among the derived features, the canal-concha intersection and canal-crus intersection intersections are detected as intersections of two geodesics. For canal-concha intersection the inter-tragal notch ridge is intersected with a geodesic between concha-peak and tragus concavity. In case of the canal-crus intersection, the intersection of crus-ridge and a geodesic between crus concavity and tragus concavity is computed (see Figure 3.3).

The crus-valley area is computed as the area enclosed by two geodesics, ridge between helix and crus valley, ridge crus valley and canal concha, and the bottom boundary contour. In Figure 3.11, the result of the detection algorithm is shown. The crus-concha intersection is another derived feature, which is detected by analyzing the tangential profile of the intersection of the shell with the crus-valley plane.

3.2.3 Update of Detected Features

Initially, most of the features are detected on the unprocessed ear impression. Due to the detailing and modeling operations applied on the mesh, they may become invalid during the processing. Therefore, the set of available features is almost doubled, because for each feature a twin exists, which is mapped onto the current shell.

The plane features mark an exception. It is assumed that the detected elbows: first bend, second bend and aperture plane as well as the crus-valley plane are constant. All other features are usually updated based on projection techniques. Features represented as a single point (peaks and some concavities) are projected onto the closest point (Euclidean distance) of the current shell. The projection is constrained by a thresholding distance to avoid invalid projections. For example, if the helix was removed a projection of the helix-peak on the current shell yields meaningless feature points (see Figure 3.12). Further postprocessing of the projected features is usually not necessary.

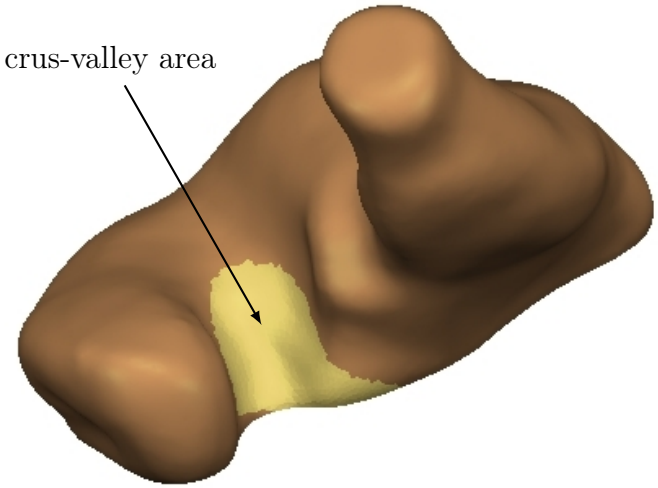


Figure 3.11: Crus-valley area detected on an ear impression.

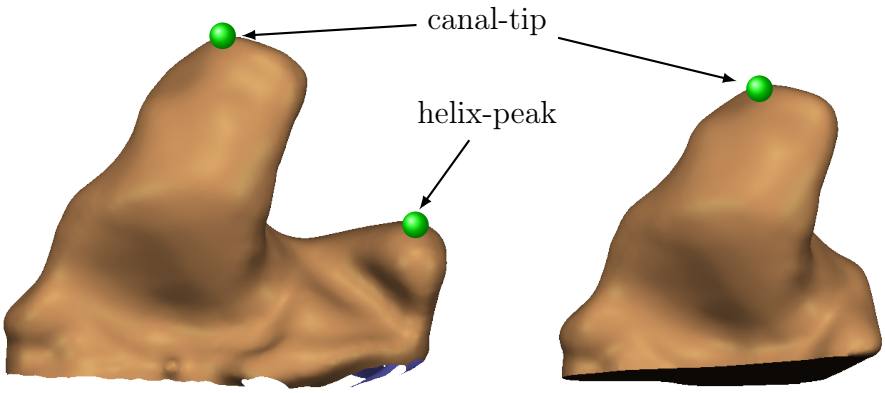


Figure 3.12: Updating of features is not always meaningful. For example the helix-peak cannot be found anymore in the processed impression on the right.

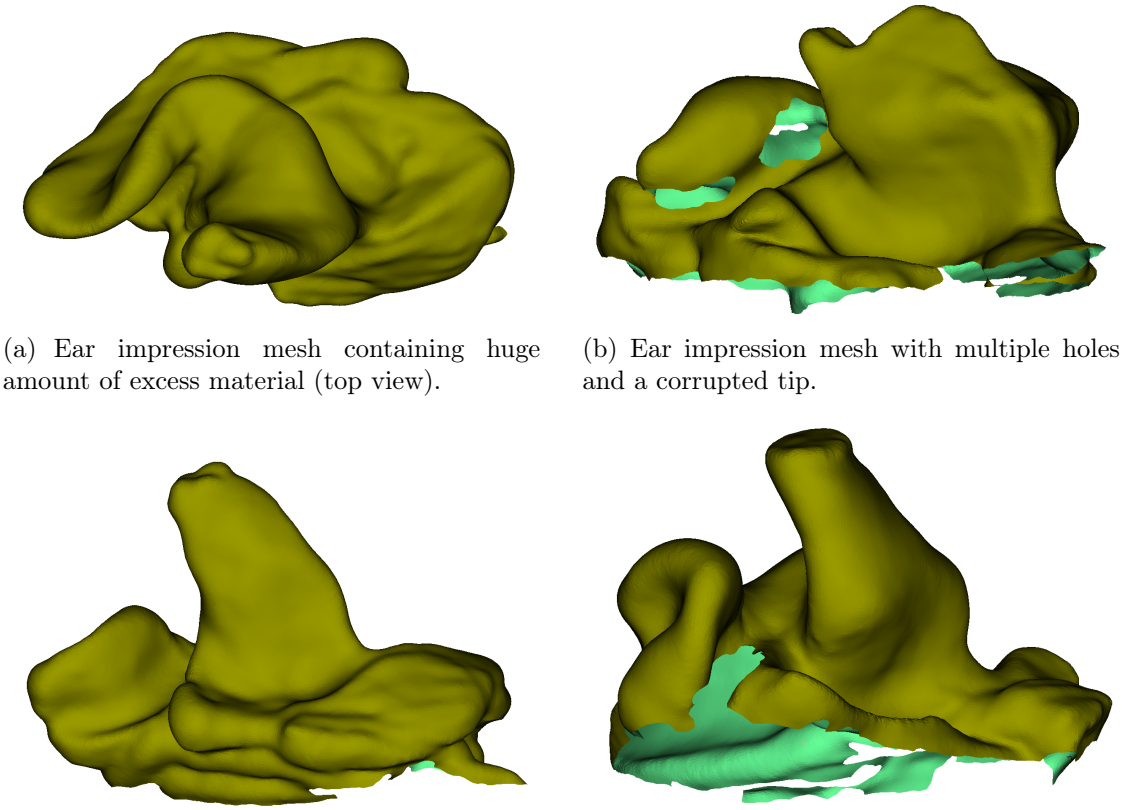
Updating the ridge features requires a two step approach. In the first step, the start and end points are projected, followed by a local optimization of the point positions. In the second step, a weighted geodesic between these points is computed on the current shell.

Features consisting of regions are recovered in a three step process. First the region boundary is projected onto the current shell, again including a distance threshold. This is followed by closing and smoothing the boundary. Finally region growing is applied to fill the region.

3.3 Clustering-based Feature Detection

The so far presented algorithms are working well in most cases [Balo 10a]. However, they can fail in case of bad or very unusual ear impressions. Bad impressions are characterized by multiple open contours, very jagged contours and extreme amount

of excess material either at the top or bottom of the mesh. A collection of bad ear impression meshes is shown in Figure 3.13. Unusual ear impressions exhibit extreme shapes. Examples are very short or corrupted canals, a very small cymba area and extreme bumps or depressions in smooth regions, see Figure 3.14.



(a) Ear impression mesh containing huge amount of excess material (top view).

(b) Ear impression mesh with multiple holes and a corrupted tip.

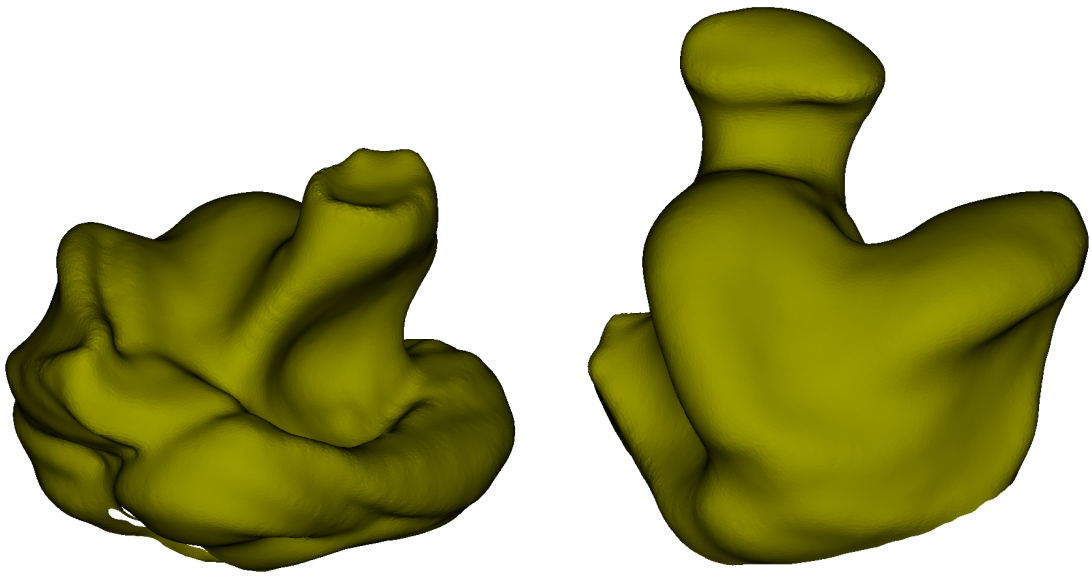
(c) Ear impression mesh containing huge amount of excess material (front view).

(d) Ear impression mesh with jagged contour, including holes and more or less disconnected patches.

Figure 3.13: The result of the scanning and subsequent triangulation largely depends on how the user places the impression in the scanner, if the mold was cut by the audiologist and how much material was used to acquire the impression.

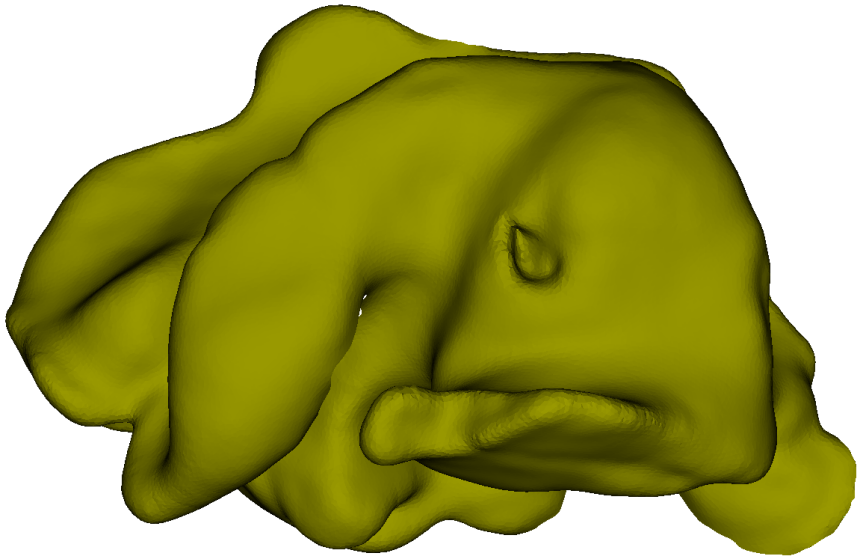
Given that the available design knowledge is closely connected to the anatomical features, there is a strong need to detect the features reliable in such situations, too. Therefore, we propose a clustering-based feature detection scheme. It requires a labeled set of representative ear impressions as reference. The idea is to register a new impression with the impressions contained in the labeled set. The best registration result allows to infer the rough location of the features on the new impression. These rough features can be either used directly or can provide meaningful input for the previously described algorithms. A graphic representation of the clustering-based feature detection scheme is given in Figure 3.15.

To realize the clustering-based feature detection scheme, three methods have to be developed: robust alignment of ear impressions (Section 3.3.1), identification of a representative set of ear impressions (Section 3.3.2), and feature projection from one impression to another (Section 3.3.6).



(a) Impression with a narrow cymba, short canal and large amount of excess material.

(b) Impression with an unusual thick tip and a wide cymba.



(c) Ear impression with a defect (wrong depression) in the concha area, probably due to cerumen or damage happened during the transport of the ear impression.

Figure 3.14: The human ear exhibits a huge variety in size, shape and smoothness. The latter is strongly influenced by the cleanliness of the ear.

3.3.1 Robust Alignment of Ear Impressions

Similar to Zouhar et al. [Zouh 06], we apply a specifically adapted version of an iterative closest point (ICP) algorithm to register ear impressions. Zouhar et al. incorporate features detected on the ear surface in order to speed up the registration. In contrast to that, we specifically avoid the usage of previously detected features, since our goal is to improve the features based on to the registration result. Our

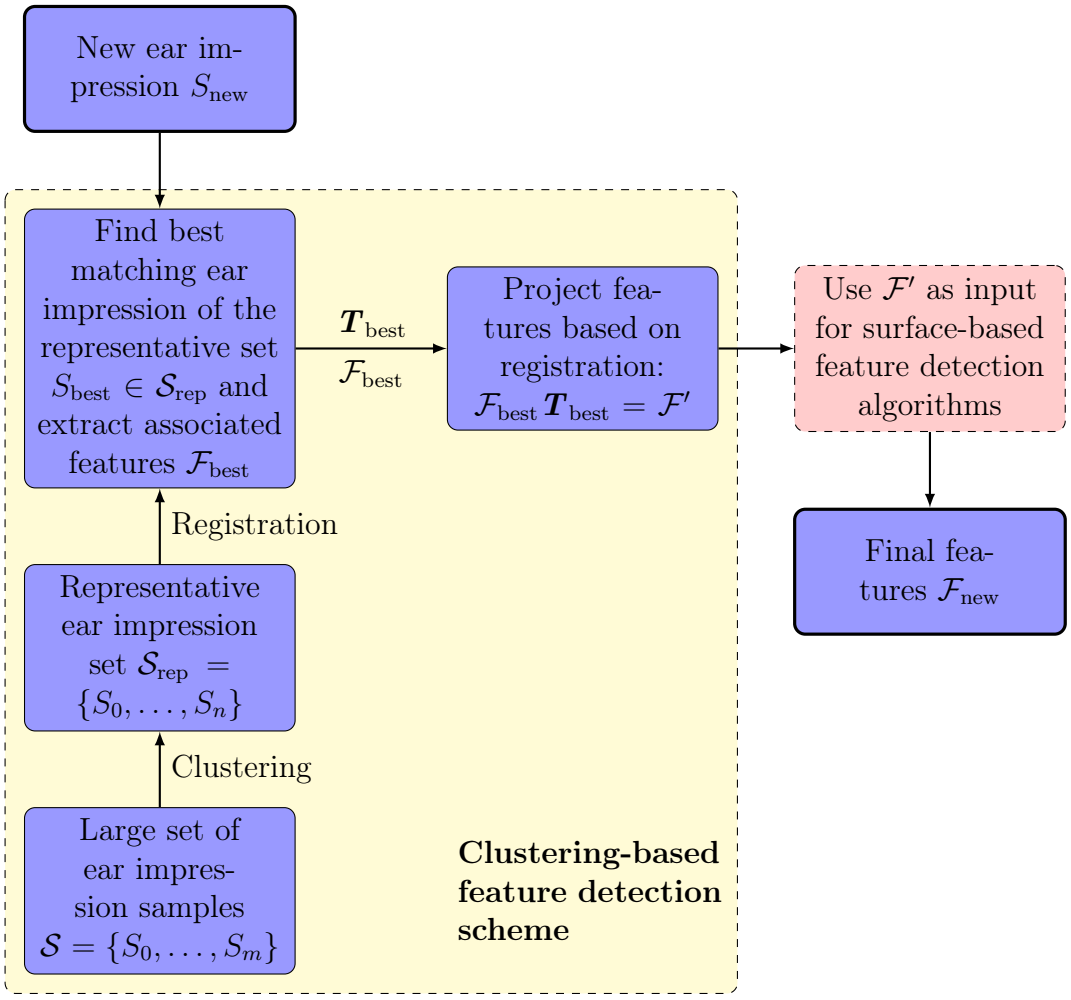


Figure 3.15: Visualization of the clustering-based feature detection scheme. It utilizes prior knowledge of previously labeled impressions to acquire rough features with the help of a transformation $\boldsymbol{T}$ defined by a registration of ear impressions.

approach is divided in two steps: rough registration using a centerline representation and fine registration using the mesh representation. Both steps are based on the ICP algorithm, which is briefly introduced in the following paragraph.

The Iterative Closest Point Algorithm

The ICP algorithm was originally applied to scan-matching tasks in the early 1990s [Besl 92] and has had many variations since then [Sega 09]. Three independently published papers outline what is still considered the state-of-the-art solution for scan-matching. Besl and McKay addressed the registration of point clouds using the *point-to-point* error metric [Besl 92]. Chen and Medioni worked with range data for object modeling and introduced the *point-to-plane* error metric [Chen 92]. Finally, Zhang described and ICP variant, which included a robust method of outlier rejection in the selection phase of the algorithm [Zhan 92].

The ICP computes the registration by iterating the following steps: (i) compute correspondences between two point clouds, (ii) reject correspondences which fail to be close to each other, and (iii) compute a transformation, which minimizes the distance between the corresponding points. The basic algorithm is listed in Algorithm 1. Later we use both, the point-to-point and the point-to-plane error metric. Therefore, we briefly describe a possibility how to solve the ICP in both cases.

Algorithm 1: Iterative closest point

Input : Two point clouds: $P, Q \in \mathbb{R}^3$, ϵ conversion criteria
Output: Transformation $\boldsymbol{T} \in \mathbb{R}^{3\times3}$, which aligns P and Q

```
// initial transformation
T ← T_0;
while ε < Σ_i^N ω_i ‖T · p_i − c_i‖_2^2 do
    for i ← 1 to N do
        c_i ← GetClosestPointInQ(T · p_i);
        if ‖T · p_i − c_i‖ ≤ θ_max then
            ω_i ← 1;
        else
            ω_i ← 0;
        end
    end
    T ← arg min_T { Σ_i^N ω_i ‖T · p_i − c_i‖_2^2 };
end
```

Point-to-Point Error Metric

The point-to-point error metric minimizes the Euclidean distance between the selected point pairs. The optimization problem can be solved by an SVD based method proposed by Arun et al. [Arun 87], a quaternion method [Horn 87], using orthonormal matrices [Horn 88] or a calculation based on dual quaternions [Walk 91]. We worked with the SVD-based method proposed by Arun et al. The optimization function of the ICP, depends on the rotation matrix $\boldsymbol{R}$ and the translation vector $\boldsymbol{t}$:

$$\epsilon = \sum_{i=1}^{N} \|(\boldsymbol{R}\boldsymbol{p}_i + \boldsymbol{t}) - \boldsymbol{q}_i\|_2^2, \quad \text{where} \tag{3.10}$$

$$\boldsymbol{q}_j = \underset{\boldsymbol{q}_x \in Q}{\arg\min} \quad d(\boldsymbol{q}_x, \boldsymbol{R}\boldsymbol{p}_i). \tag{3.11}$$

In Eq. (3.11), function $d(\boldsymbol{x}, \boldsymbol{y})$ computes the Euclidean distance between two points. Given Eq. (3.10), the first step is to decouple translation and rotation. Therefore, the algorithm computes the center points $(\bar{\boldsymbol{p}}, \bar{\boldsymbol{q}})$ of both point sets and translates the points sets accordingly.

$$P' : \boldsymbol{p}'_i = \boldsymbol{p}_i - \bar{\boldsymbol{p}} \tag{3.12}$$

$$Q' : \boldsymbol{q}'_j = \boldsymbol{q}_j - \bar{\boldsymbol{q}} \tag{3.13}$$

The decoupled optimization equation is then

$$\epsilon = \sum_{i=1}^{N} \|\boldsymbol{R}\boldsymbol{p}'_i - \boldsymbol{q}'_i\|_2^2 \,. \tag{3.14}$$

For the sake of clarity, Eq. (3.14) contains two simplifications: Index i matches the closest points of both point clouds and both point clouds contain the same number of points. The decoupled translation vector $\boldsymbol{t}$ can be computed with

$$\boldsymbol{t} = \bar{\boldsymbol{q}} - \boldsymbol{R}\bar{\boldsymbol{p}}, \tag{3.15}$$

and the rotation matrix $\boldsymbol{R}$ with help of matrix $\boldsymbol{H}$:

$$\boldsymbol{H} = \sum_{i=1}^{N} \boldsymbol{p}'_i \boldsymbol{q}'^{T}_i, \tag{3.16}$$

where superscript T denotes transpose. The application of the SVD yields $\boldsymbol{H} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$. The desired rotation matrix $\boldsymbol{R}$ is then given by

$$\boldsymbol{R} = \boldsymbol{V}\boldsymbol{U}^T. \tag{3.17}$$

Depending on the properties of the point clouds, the solution may not be unique. If the data points are coplanar or collinear, Eq. (3.17) can fail to compute the desired solution.

- If P' is collinear, then there are infinitely many rotations and reflections solving Eq. (3.14).
- If P' is coplanar, then there are two unique solutions: the desired rotation matrix and its reflection. The reflection was given by Eq. (3.17), if $\det(\boldsymbol{R}) = -1$. To compute the correct rotation matrix, $\boldsymbol{V}$ has to be adjusted according to the eigenvalues of $\boldsymbol{H}$. One of the eigenvalues must be zero. Changing the sign of that eigenvalues position will not change $\boldsymbol{H}$. Hence, we construct $\boldsymbol{V}'$ by flipping the sign of the ith column of $\boldsymbol{V}$, where index i identifies the zero eigenvalue. The correct rotation matrix is then given by
$$\boldsymbol{R} = \boldsymbol{V}'\boldsymbol{U}^T.$$
- If P' is not coplanar, then one unique solution exists. Hence, Eq. (3.17) computes the correct rotation matrix.

Point-to-Plane Error Metric

The point-to-plane error metric is more robust, accurate and converges faster than the point-to-point error-metric [Chen 92, Rusi 01, Sega 09]. It utilizes the precomputed surface normal $\boldsymbol{n}_i \in \mathbb{R}^3$ as an additional input and allows that smooth or planar areas of the meshes slide over each other easily. The optimization problem is defined as

$$\begin{aligned} \epsilon &= \sum_{i=1}^{N} \left\| \left((\boldsymbol{R}\boldsymbol{p}_i + \boldsymbol{t}) - \boldsymbol{q}'_j \right)^T \boldsymbol{n}_i \right\|_2^2, \quad \text{with} \\ \boldsymbol{q}'_j &= \left\{ \boldsymbol{q} \;\middle|\; \arg\min_{\boldsymbol{q}\in\boldsymbol{h}_j} \|\boldsymbol{R}\boldsymbol{p}_i + \boldsymbol{t} - \boldsymbol{q}\|_2 \right\}, \end{aligned} \tag{3.18}$$

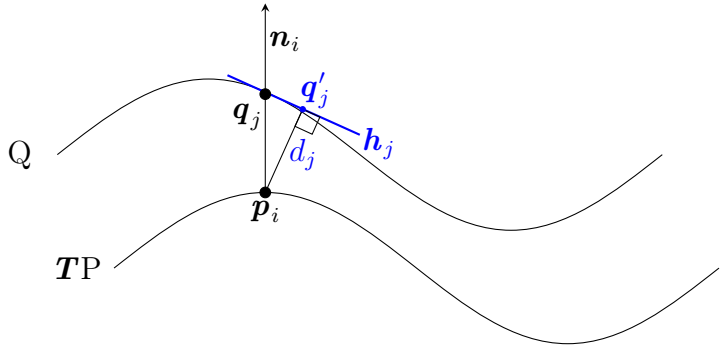


Figure 3.16: The point-to-plane error metric minimizes the distance d_j between the tangent plane $\boldsymbol{h}_j$ and the point $\boldsymbol{p}_i$.

where $\boldsymbol{h}_j$ is the tangent plane of Q at $\boldsymbol{q}_j$. Thus, Eq. 3.18 minimizes the distance $d_j \in \mathbb{R}$ between the tangent plane $\boldsymbol{h}_j$ and $\boldsymbol{p}_i$. In Figure 3.16 an example in 2-D is given. Note: For the sake of readability and coherence with the derivation of the point-to-point error metric, we denote $\boldsymbol{q}'_j$ as $\boldsymbol{q}_i$ and assume that both point clouds contain the same number of points:

$$\epsilon = \sum_{i=1}^{N} \left\| \left((\boldsymbol{R}\boldsymbol{p}_i + \boldsymbol{t}) - \boldsymbol{q}_i \right)^T \boldsymbol{n}_i \right\|_2^2 . \tag{3.19}$$

Equation (3.19) describes a nonlinear problem, but if we assume that we have a rough alignment of P and Q, due to an initial transformation, it is possible to linearize the rotation by approximating $\cos\alpha$ by 1 and $\sin\alpha$ by α. Hence, for rotations α, β and γ around the x, y and z axes, the rotation matrix can be approximated as

$$\boldsymbol{R} = \begin{pmatrix} 1 & -\gamma & \beta \\ \gamma & 1 & -\alpha \\ -\beta & \alpha & 1 \end{pmatrix} . \tag{3.20}$$

Substitution of Eq. (3.20) into (3.19) yields

$$\begin{aligned} \epsilon \; = \sum_{i=1}^{N} \big(& (p_{i,x} - \gamma p_{i,y} + \beta p_{i,z} + t_x - q_{i,x})\, n_{i,x} + \\ & (\gamma p_{i,x} + p_{i,y} - \alpha p_{i,z} + t_y - q_{i,y})\, n_{i,y} + \\ & (-\beta p_{i,x} + \alpha p_{i,y} + p_{i,z} + t_z - q_{i,z})\, n_{i,z} \big)^2 . \end{aligned}$$

If we now define $\boldsymbol{a} = \boldsymbol{p} \times \boldsymbol{n}$ and $\boldsymbol{r} = (\alpha \;\; \beta \;\; \gamma)^T$ then Eq. (3.19) can be compactly written as

$$\epsilon = \sum_{i=1}^{N} \left[(\boldsymbol{p}_i - \boldsymbol{q}_i)^T \boldsymbol{n}_i + \boldsymbol{t}\boldsymbol{n}_i + \boldsymbol{r}\boldsymbol{a}_i \right]^2 . \tag{3.21}$$

To minimize Eq. (3.21) we compute the partial derivatives and set them to zero.

$$\begin{aligned}
\frac{\delta\epsilon}{\delta\alpha} &= \sum_{i=1}^{N} 2a_{i,x}\left[(\boldsymbol{p}_i - \boldsymbol{q}_i)^T \boldsymbol{n}_i + \boldsymbol{t}\boldsymbol{n}_i + \boldsymbol{r}\boldsymbol{a}_i\right] = 0 \\
\frac{\delta\epsilon}{\delta\beta} &= \sum_{i=1}^{N} 2a_{i,y}\left[(\boldsymbol{p}_i - \boldsymbol{q}_i)^T \boldsymbol{n}_i + \boldsymbol{t}\boldsymbol{n}_i + \boldsymbol{r}\boldsymbol{a}_i\right] = 0 \\
\frac{\delta\epsilon}{\delta\gamma} &= \sum_{i=1}^{N} 2a_{i,z}\left[(\boldsymbol{p}_i - \boldsymbol{q}_i)^T \boldsymbol{n}_i + \boldsymbol{t}\boldsymbol{n}_i + \boldsymbol{r}\boldsymbol{a}_i\right] = 0 \\
\frac{\delta\epsilon}{\delta t_x} &= \sum_{i=1}^{N} 2n_{i,x}\left[(\boldsymbol{p}_i - \boldsymbol{q}_i)^T \boldsymbol{n}_i + \boldsymbol{t}\boldsymbol{n}_i + \boldsymbol{r}\boldsymbol{a}_i\right] = 0 \\
\frac{\delta\epsilon}{\delta t_y} &= \sum_{i=1}^{N} 2n_{i,y}\left[(\boldsymbol{p}_i - \boldsymbol{q}_i)^T \boldsymbol{n}_i + \boldsymbol{t}\boldsymbol{n}_i + \boldsymbol{r}\boldsymbol{a}_i\right] = 0 \\
\frac{\delta\epsilon}{\delta t_z} &= \sum_{i=1}^{N} 2n_{i,z}\left[(\boldsymbol{p}_i - \boldsymbol{q}_i)^T \boldsymbol{n}_i + \boldsymbol{t}\boldsymbol{n}_i + \boldsymbol{r}\boldsymbol{a}_i\right] = 0
\end{aligned}$$

The above can be expressed in matrix form

$$\begin{aligned}
\sum_{i=1}^{N} \begin{pmatrix}
a_{i,x}a_{i,x} & a_{i,x}a_{i,y} & a_{i,x}a_{i,z} & a_{i,x}n_{i,x} & a_{i,x}n_{i,y} & a_{i,x}n_{i,z} \\
a_{i,y}a_{i,x} & a_{i,y}a_{i,y} & a_{i,y}a_{i,z} & a_{i,y}n_{i,x} & a_{i,y}n_{i,y} & a_{i,y}n_{i,z} \\
a_{i,z}a_{i,x} & a_{i,z}a_{i,y} & a_{i,z}a_{i,z} & a_{i,z}n_{i,x} & a_{i,z}n_{i,y} & a_{i,z}n_{i,z} \\
n_{i,x}a_{i,x} & n_{i,x}a_{i,y} & n_{i,x}a_{i,z} & n_{i,x}n_{i,x} & n_{i,x}n_{i,y} & n_{i,x}n_{i,z} \\
n_{i,y}a_{i,x} & n_{i,y}a_{i,y} & n_{i,y}a_{i,z} & n_{i,y}n_{i,x} & n_{i,y}n_{i,y} & n_{i,y}n_{i,z} \\
n_{i,z}a_{i,x} & n_{i,z}a_{i,y} & n_{i,z}a_{i,z} & n_{i,z}n_{i,x} & n_{i,z}n_{i,y} & n_{i,z}n_{i,z}
\end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ t_x \\ t_y \\ t_z \end{pmatrix} \\
= -\sum_{i=1}^{N} \begin{pmatrix}
a_{i,x}\,(\boldsymbol{p}_i - \boldsymbol{q}_i)^T \boldsymbol{n}_i \\
a_{i,y}\,(\boldsymbol{p}_i - \boldsymbol{q}_i)^T \boldsymbol{n}_i \\
a_{i,z}\,(\boldsymbol{p}_i - \boldsymbol{q}_i)^T \boldsymbol{n}_i \\
n_{i,x}\,(\boldsymbol{p}_i - \boldsymbol{q}_i)^T \boldsymbol{n}_i \\
n_{i,y}\,(\boldsymbol{p}_i - \boldsymbol{q}_i)^T \boldsymbol{n}_i \\
n_{i,z}\,(\boldsymbol{p}_i - \boldsymbol{q}_i)^T \boldsymbol{n}_i
\end{pmatrix}. \qquad (3.22)
\end{aligned}$$

We now have a linear matrix equation of the form $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. A is a 6×6 covariance matrix composed of $\boldsymbol{a}_i$ and $\boldsymbol{n}_i$. $\boldsymbol{x}$ is a 6×1 vector of unknowns and $\boldsymbol{b}$ is a 6×1 data dependent vector. This equation can be solved using standard methods, like the Cholesky or LU-decomposition [Bron 01]. The obtained vector $\boldsymbol{x}$ provides the optimal incremental rotation and translation.

Alignment of Ear Impressions Using ICP

As previously described, our alignment approach consists of two steps. The main reason is that applying the ICP with point-to-plane metric on two ear impressions can fail, because the assumption of incremental small rotations can be invalid. Hence, a rough alignment using a centerline representation is computed first.

Centerline Computation The initial centerline is computed by equidistantly slicing the mesh parallel to a plane defined by the open contour at the bottom of the impression. For each slice the center point is computed resulting in an ordered set of points similar to an active contour [Kass 88], see Figure 3.17.

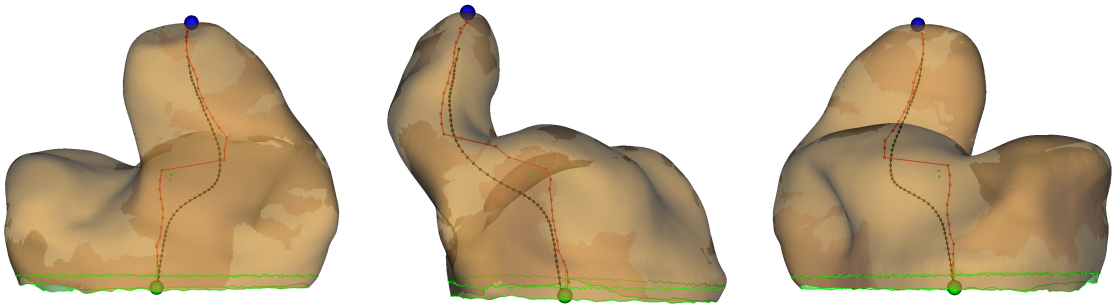


Figure 3.17: Result of the centerline computation showing the rough (jagged) and final (smooth) centerline.

Afterward, the centerline $\mathcal{L} = \{\boldsymbol{l}_1, \ldots, \boldsymbol{l}_n\}$ is smoothed by applying internal and external energies on each point $\boldsymbol{l}_i \in \mathcal{L}$. Beforehand the centerline is resampled to achieve a consistent sampling distance. The external energy E_{ext} optimizes the location of the centerline in respect to the surrounding mesh surface S.

$$E_{\text{ext},i} = \frac{1}{N_v} \sum_{v=1}^{N_v} \frac{\boldsymbol{x}_{v,i}}{\|\boldsymbol{x}_{v,i}\|_2}, \tag{3.23}$$

where $\boldsymbol{x}_{v,i}$ is the intersection of ray $v(s) = \boldsymbol{l}_i + s\boldsymbol{u}$ with S. The direction vector $\boldsymbol{u}$ is randomized for each iteration v and $s \in \mathbb{R}$. Finally, $E_{\text{ext},i}$ is normalized with the total number of random rays N_v, typically $N_v = 100$.

The internal energy E_{int} incorporates the neighborhood information, in form of a derivative, to enforce a smooth line.

$$E_{\text{int},i} = \boldsymbol{l}_{i-1} + \boldsymbol{l}_{i+1} - 2\boldsymbol{l}_i \tag{3.24}$$

The final force for a particular point $\boldsymbol{l}_i \in L$ is a weighted combination of the original point, and the internal and external force applied on that point:

$$\boldsymbol{l}'_i = \boldsymbol{l}_i + \alpha E_{\text{int},i} + \beta E_{\text{ext},i}. \tag{3.25}$$

The coefficients in Eq. (3.25) are usually set to $\alpha = 0.04$ and $\beta = 1$. The application of internal and external energies is repeated until the centerline converges. The final centerline is shown in Figure 3.17.

Centerline Alignment To find the best initial transformation given two centerlines, we evaluate the registration error of multiple ICP runs. The usage of multiple ICP runs is implied by our efficient point matching strategy. We use the fact that the centerlines are ordered and, therefore, we can shift the shorter one along the other. In each iteration the longer centerline is adjusted to the length of the shorter one.

Hence, an algorithm for finding the best point pairs is not necessary. Instead the point indices are used, see Figure 3.18. As result we obtain a transformation matrix $\boldsymbol{T}_i$ for each shift operation, which is sorted according to the registration error and pruned $\mathcal{T} = \{\boldsymbol{T}_1, \dots \boldsymbol{T}_N\}$, where N is typically restricted to ten.

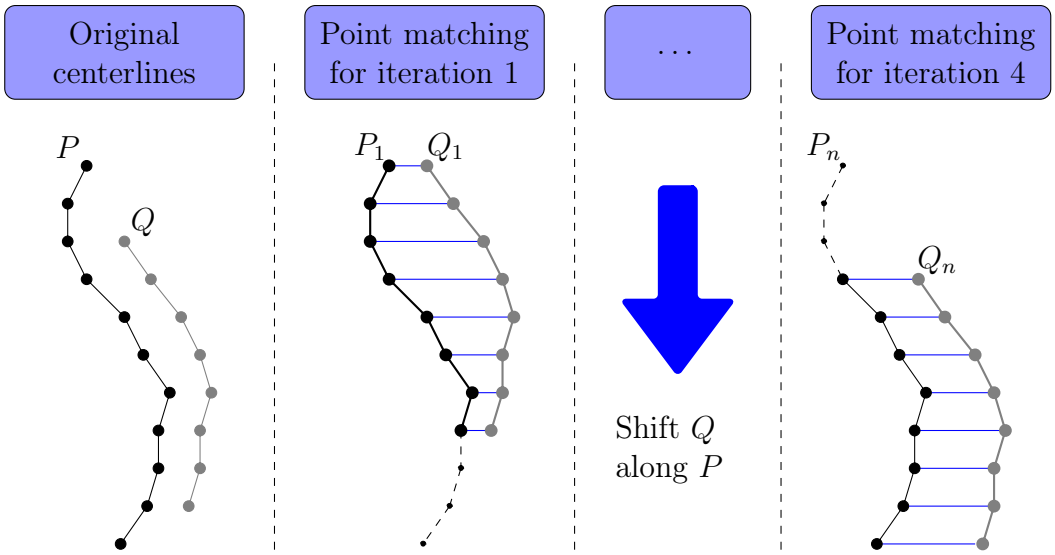


Figure 3.18: Point matching scheme for centerline based ICP.

Final Alignment The final alignment is done using the point clouds of the ear impressions. In order to speed up the registration, the meshes are sub-sampled. Since we already have a rough pre-alignment ($\mathcal{T}$), we can employ the the point-to-plane error metric. Furthermore, we consider different extensions of the ICP suggested by Rusinkiewicz and Levoy [Rusi 01]. Thus, we implemented several point selection, point pair rejection and point pair weighting techniques.

Point selection defines the sub-sampling technique applied.

- Uniform sub-sampling
- Random sub-sampling

Point pair rejection defines a $\theta_{\max} \in \mathbb{R}^+$, which the Euclidean distance of a point pair must not exceed.

- Fixed thresholds
- Standard deviation rejection computes the mean and standard deviation (σ) of the point pair distances and pairs exceeding $\theta_{max} = a\sigma$ are rejected. The tunable factor $a \in \mathbb{R}^+$ is typically set to 2.5 [Rusi 01].
- Worst pairs rejection applies not a distance based $\theta_{\max}$, but rejects a certain percentage of point pairs. Therefore, the point pairs are ordered according to the point pair distance and the worst $a\,\%$ are rejected, where a is typically set to $10\,\%$ [Rusi 01].

Point pair weighting defines a weight $\omega_i \in \mathbb{R}^+$ for each point pair, which is integrated into the optimization equation as shown in Algorithm 1, line 12.

- Distance penalty weighting applies

$$\omega_i = 1 - \frac{d(\boldsymbol{p}_i, \boldsymbol{q}_i)}{d_{\max}},$$

  where function d computes the Euclidean distance and $d_{\max}$ denotes the maximum distance of all point pairs.

- Normal compatibility uses the inner product of the surface normals as weighting factor

$$\omega_i = \boldsymbol{n}_i^{(p)T} \cdot \boldsymbol{n}_i^{(q)},$$

  where $\boldsymbol{n}_i^{(p)}$ denotes the surface normal at point $\boldsymbol{p}_i$ and $\boldsymbol{n}_i^{(q)}$ the surface normal at point $\boldsymbol{q}_i$ respectively.

3.3.2 Identification of a Representative Set of Ear Impressions

In order to identify a representative set $\mathcal{S}_{\text{rep}}$ of ear impressions, a measure of similarity is needed. Therefore, we make use of the previously described ear impression alignment. Besides the transformation matrix, the ICP returns the average squared error (δ_{se}) of the alignment

$$\delta_{\text{se}}\left(S_{\text{P}}, S_{\text{Q}}\right) = \frac{1}{N} \sum_{i=1}^{N} \left\| \boldsymbol{p}_i - \boldsymbol{T}\boldsymbol{q}_i \right\|_2^2 . \tag{3.26}$$

The δ_{se} can be interpreted as a similarity between two aligned ear impressions. Thus, we define a similarity matrix $\boldsymbol{S}$, which contains the results of ear impression alignments

$$\boldsymbol{S} = \begin{pmatrix} s_{11} & \cdots & s_{1N} \\ \vdots & \ddots & \vdots \\ s_{N1} & \cdots & s_{NN} \end{pmatrix} . \tag{3.27}$$

An entry $s_{ij} \in \mathbb{R}$ is, therefore, defined as $s_{ij} = \delta_{\text{se}}\left(S_i, S_j\right)$. Having defined the similarity measure, we rely on clustering techniques to identify a representative partitioning of a large set of ear impressions. We consider three different clustering methods:

1. agglomerative hierarchical clustering (AHC) [Duda 01],
2. expectation maximization clustering (EMC) [Bish 06], and
3. Chinese restaurant clustering (CRC) [Qin 06].

The reason for considering three methods is that it is currently unknown if there is a natural partition of ear impressions, and if there is one, how many clusters it contains. In several interviews with hearing aid design experts, it was not possible to acquire an estimation of the needed number of clusters. They could only specify a range, expecting the natural partitioning to be within three to 100 clusters.

Hence, we chose these three clustering techniques, because they offer different strategies to identify the most natural clustering given a sample set. The AHC measures the level of dissimilarity for each clustering iteration, which provides a heuristic criterion for determination of the number of needed clusters. The EMC analyzes the likelihood of a clustering and compares it for different numbers of clusters, finally choosing the one with the highest likelihood. The CRC goes one step further and analyzes the likelihood of partitions using a Dirichlet process. It is of special interest in our case, because it described in literature as a powerful tool to identify partitions based on data [Blei 11]. All three methods are briefly introduced in the following.

3.3.3 Agglomerative Hierarchical Clustering

Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. In the agglomerative case a *bottom up* strategy is applied, while in the divisive case a *top down* approach is used. Let $\mathcal{S} = \{S_0, S_1, \ldots, S_N\}, N \in \mathbb{N}$ denote the available sample set. In bottom up, each sample (3-D mesh or an unique ear impression) $S_i \in \mathcal{S}$ starts in its own cluster $C_i = \{S_i\}$, while in top down all samples start in one cluster $C_0 = \mathcal{S} = \{S_0, S_1, \ldots, S_N\}$. In both cases the ordering of the samples is irrelevant for the result. Hence, the task in agglomerative hierarchical clustering is to merge clusters until a certain number $k \geq 1$ of clusters is reached. In general, merging is done in a greedy manner. The basic algorithm is shown in Algorithm 2. The algorithm terminates when the given number of clusters is reached. In case $k = 1$ the complete hierarchy, often referred to as dendrogram, is constructed [Duda 01].

Algorithm 2: Agglomerative Hierarchical Clustering

input : Samples: $\mathcal{S} = \{S_1, \ldots, S_N\}$, clusters k
output: Clusters: $\mathcal{C} = \{C_1, \ldots, C_k\}$, $C_i \subseteq \mathcal{S}$

```
1 k̂ ← N;
2 C_i ← S_i, i = 1, ··· , N;
3 while k̂ > k do
4 |   k̂ ← k̂ − 1;
5 |   {C_i, C_j} ← FindClosestCluster();
6 |   C_i ← MergeCluster(C_i, C_j);
7 |   DeleteCluster(C_j);
8 end
```

The critical step for AHC is how to compute the closest or most similar clusters. Hence, it is necessary to define a metric for cluster similarity, usually referred to as linkage criterion. Note: All criteria are expressed with help of the similarity matrix $\boldsymbol{S}$ and entries s_{ij} defined in Eq. 3.27. Four different linkage criteria are considered.

1. Nearest neighbor (also referred to as single-linkage or minimum) clustering

$$\delta_{\mathrm{NN}}(C_i, C_j) = \min s_{ab}, \quad S_a \in C_i \wedge S_b \in C_j. \tag{3.28}$$

2. Farthest neighbor (also referred to as complete-linkage or maximum) clustering

$$\delta_{\mathrm{FN}}(C_i, C_j) = \max s_{ab}, \quad S_a \in C_i \wedge S_b \in C_j. \tag{3.29}$$

3. Mean (not to be confused with average) clustering

$$\delta_{\mathrm{M}}(C_i, C_j) = s_{ab}, \quad S_a \text{ centroid of } C_i \wedge S_b \text{ centroid of } C_j. \tag{3.30}$$

4. Average-linkage clustering

$$\delta_{\mathrm{A}}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{S_a \in C_i} \sum_{S_b \in C_j} s_{ab}. \tag{3.31}$$

Number of Clusters In agreement with Duda et al. [Duda 01], we define the natural number of clusters by the largest difference of the dissimilarity level between two subsequent hierarchy levels. Duda et al. define the dissimilarity δ_{dis} as the distance between the closest clusters. Thus, the distance is computed using one of the Eqs. (3.28) to (3.31). The drop or jump in dissimilarity is computed for every level i

$$\Delta_{i+1} = |\delta_{\mathrm{dis},i} - \delta_{\mathrm{dis},i+1}|.$$

If $\Delta_i \geq \Delta_k, \forall k = 1, \ldots, N$, where N denotes the number of samples, then the natural number of clusters is estimated to correspond to the clustering of iteration $i - 1$.

A more detailed discussion of the agglomerative hierarchical clustering can be found in Duda et al. [Duda 01].

3.3.4 Expectation Maximization Clustering

Some of the shortcomings of a AHC technique described above is the dependency on heuristics to identify the most natural clustering for the given data. In contrast to that, the goal of the expectation maximization (EM) algorithm is to find a maximum likelihood solution for probabilistic models having latent variables [Bish 06]. In EMC, the latent variables are the cluster assignments. The following EM overview is based on the works by C. Bishop [Bish 06] and J. Hornegger [Horn 00].

The EM algorithm is based on the missing information principle: observable information equals complete information minus hidden information, which can be expressed as

$$L(\boldsymbol{X}, \boldsymbol{B}) = \log p(\boldsymbol{X}|\boldsymbol{B}) = \log p(\boldsymbol{X}, \boldsymbol{Y}|\boldsymbol{B}) - \log p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{B}). \tag{3.32}$$

$\boldsymbol{X}$ denotes the observable information, which corresponds to the similarity values. $\boldsymbol{Y}$ denotes the latent variables, which corresponds to the unknown cluster assignments and $\boldsymbol{B}$ denotes the set of model parameters to be optimized.

Let $\hat{\boldsymbol{B}}^{(0)}$ be an initial estimation for the parameter set, then the key equation of the EM is given by

$$\begin{aligned} E\left[L(\boldsymbol{X}, \boldsymbol{B})|\boldsymbol{X}, \hat{\boldsymbol{B}}^{(i)}\right] &= E\left[\log p(\boldsymbol{X}, \boldsymbol{Y}|\hat{\boldsymbol{B}}^{(i+1)})|\boldsymbol{X}, \hat{\boldsymbol{B}}^{(i)}\right] \\ &\quad - E\left[\log p(\boldsymbol{Y}|\boldsymbol{X}, \hat{\boldsymbol{B}}^{(i+1)})|\boldsymbol{X}, \hat{\boldsymbol{B}}^{(i)}\right] \\ &= Q(\hat{\boldsymbol{B}}^{(i+1)}|\hat{\boldsymbol{B}}^{(i)}) - H(\hat{\boldsymbol{B}}^{(i+1)}|\hat{\boldsymbol{B}}^{(i)}). \end{aligned} \tag{3.33}$$

The Q-function denotes the Kullback-Leibler divergence, while the H-function is the negative entropy:

$$\begin{aligned} Q(\hat{\boldsymbol{B}}^{(i+1)}|\hat{\boldsymbol{B}}^{(i)}) &= \sum_{\boldsymbol{Y}} p(\boldsymbol{Y}|\boldsymbol{X}, \hat{\boldsymbol{B}}^{(i)}) \log p(\boldsymbol{X}, \boldsymbol{Y}|\hat{\boldsymbol{B}}^{(i+1)}), \\ H(\hat{\boldsymbol{B}}^{(i+1)}|\hat{\boldsymbol{B}}^{(i)}) &= \sum_{\boldsymbol{Y}} p(\boldsymbol{Y}|\boldsymbol{X}, \hat{\boldsymbol{B}}^{(i)}) \log p(\boldsymbol{Y}|\boldsymbol{X}, \hat{\boldsymbol{B}}^{(i+1)}). \end{aligned} \tag{3.34}$$

It can be shown that the conditional expectation of L is independent of the latent variables $\boldsymbol{Y}$ and of $\hat{\boldsymbol{B}}^{i}$ in iteration $i+1$ [Demp 77]. Therefore, it is sufficient to optimize the right side of Eq. (3.33) as shown in Algorithm 3.

Algorithm 3: Expectation Maximization Algorithm

Input : Joint distribution $p(\boldsymbol{X}, \boldsymbol{Y}|\hat{\boldsymbol{B}})$
Output: Optimized parameter set $\hat{\boldsymbol{B}}$

1 $\hat{\boldsymbol{B}}^{(0)} \leftarrow$ initial estimation;
2 **repeat**
3 **E step:** Evaluate $p(\boldsymbol{X}, \boldsymbol{Y}|\hat{\boldsymbol{B}}^{(i)})$;
4 **M step:** Evaluate $\hat{\boldsymbol{B}}^{(i+1)}$ given by $\hat{\boldsymbol{B}}^{(i+1)} = \arg\min_{\hat{\boldsymbol{B}}} Q(\hat{\boldsymbol{B}}^{(i+1)}|\hat{\boldsymbol{B}}^{(i)})$;
5 **until** $\hat{\boldsymbol{B}}^{(i)} = \hat{\boldsymbol{B}}^{(i+1)}$;
6 $\hat{\boldsymbol{B}} \leftarrow \hat{\boldsymbol{B}}^{(i+1)}$;

The solution increases monotonically, because an increase in the Kullback-Leibler divergence implies a decrease of the entropy [Horn 00]. In case of the availability of a prior $p(\boldsymbol{B})$ a maximum a posterior solution can be found by optimizing $Q(\hat{\boldsymbol{B}}^{(i+1)}|\hat{\boldsymbol{B}}^{(i)}) + \log p(\boldsymbol{B})$.

EM Using Gaussian Mixture Models We assume that the clustering of ear impressions can be described by a mixture of Gaussian distributions and that the data is independent and identically distributed. Then the log-likelihood function is given by

$$\ln p(\boldsymbol{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}, \tag{3.35}$$

where K denotes the number of clusters. The Gaussian mixtures are defined by the means $\boldsymbol{\mu}$ and covariance matrices $\boldsymbol{\Sigma}$. The mixing variable $\boldsymbol{\pi}$ is defined as $p(y_k = 1) = \pi_k$, where $\boldsymbol{y}_i \in \boldsymbol{Y}$ has a 1-of-K representation [Bish 06]. Furthermore, the parameters $\{\pi_k\}$ must satisfy $0 \leq \pi_k \leq 1$ and $\sum_{k=1}^{K} \pi_k = 1$. Derivation of Eq. (3.35) with respect to $\boldsymbol{\mu}_k$ yields

$$0 = \sum_{n=1}^{N} \underbrace{\frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}}_{\gamma(y_{nk})} \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_k) \tag{3.36}$$

After multiplication with $\boldsymbol{\Sigma}_k$, we obtain

$$\begin{aligned} \boldsymbol{\mu}_k &= \frac{1}{n_k} \sum_{n=1}^{N} \gamma(y_{nk}) \boldsymbol{x}_n, \quad \text{where} \\ n_k &= \sum_{n=1}^{N} \gamma(y_{nk}). \end{aligned} \tag{3.37}$$

The derivation of Eq. (3.35) with respect to $\boldsymbol{\Sigma}_k$ and following the same reasoning yields

$$\boldsymbol{\Sigma}_k = \frac{1}{n_k} \sum_{n=1}^{N} \gamma(y_{nk})(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T. \tag{3.38}$$

To maximize the log-likelihood function with respect to the mixing variable $\boldsymbol{\pi}$, a Lagrange multiplier is required to take the constraints into account

$$\ln p(\boldsymbol{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right). \tag{3.39}$$

The derivation is then given by

$$0 = \sum_{n=1}^{N} \frac{\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} + \lambda. \tag{3.40}$$

After multiplication with π_k and summation over k, we find $\lambda = -N$. Therefore, we eliminate λ in Eq. (3.40) and obtain

$$\pi_k = \frac{n_k}{N}. \tag{3.41}$$

Using the acquired results of Eqs. (3.36) to (3.41), the EM algorithm can be written as

1. Initialize the Gaussian mixture parameters.
2. E step: Evaluate the responsibilities $\gamma(y_{nk})$ using Eq. (3.36).
3. M step: Estimate Gaussian mixture parameters using Eqs. (3.37), (3.38) and (3.41).
4. Evaluate the log-likelihood (see Eq. (3.35)) and go back to step 1 if the convergence criterion is not satisfied.

Number of Clusters The weka toolbox[1] implementation of the EM algorithm assigns a probability distribution to each instance which indicates the probability of it belonging to each of the clusters. The decision about the number of clusters is

[1] Waikato Environment for Knowledge Analysis, http://www.cs.waikato.ac.nz/ml/weka/, source code: http://www.java2s.com/Open-Source/Java-Document/Science/weka/weka/clusterers/EM.java.htm, last visited 09/01/2010.

based on cross validation. The algorithm starts by setting the number of clusters to one. This is followed by a split of the data set into ten folds. For each of the ten folds the EM algorithm is performed. The resulting log-likelihod is the average of the ten results. If the log-likelihod increased compared to the last result, then the number of clusters is increased by one and the algorithm starts again. In Algorithm 4 the scheme of the EMC is shown. Note: The weka implementation assumes a Gaussian mixture model with i.i.d. samples.

Algorithm 4: Expectation-Maximization Clustering

input : Observations $\boldsymbol{X}$
output: Number of clusters K and cluster assignment $\boldsymbol{y}_i$ for each observation

$\hat{K} \leftarrow 1$;
$ll_{\text{old}} \leftarrow 0$;
$ll_{\text{new}} \leftarrow 0$;
repeat
  $ll_{\text{old}} \leftarrow ll_{\text{new}}$;
  $ll_{\text{new}} \leftarrow 0$;
  for $i \leftarrow 1$ **to** 10 **do**
    $\boldsymbol{X}' \leftarrow$ `GetRandomSplitof(`$\boldsymbol{X}$`)`;
    `InitEM(`$\boldsymbol{X}', \hat{K}$`)`;
    `// Uses 10 k-Means clusterings to initialize`
    `// the Gaussian mixture parameters`
    $ll_{\text{cur}} \leftarrow 0$;
    while *not converged* **do**
      $ll_{\text{cur}} \leftarrow$ `EStep(`$\boldsymbol{X}'$`)`;
      `MStep(`$\boldsymbol{X}'$`)`;
    end
    $ll_{\text{new}} \leftarrow ll_{\text{new}} + ll_{\text{cur}}$;
  end
  $ll_{\text{new}} \leftarrow ll_{\text{new}}/10$;
  `// Average over ten folds`
  $\hat{K} \leftarrow \hat{K} + 1$;
until $ll_{new} < ll_{old}$;
$K \leftarrow \hat{K} - 1$;

3.3.5 Chinese Restaurant Clustering

The last considered clustering technique is the Chinese Restaurant Clustering (CRC) [Qin 06], which is often referred to as Chinese restaurant process mixture. At first, an informal description of the Chinese restaurant process (CRP) is given. Afterward, we provide the necessary links to the Dirichlet distributions and related processes. Finally, we introduce the implementation of the CRC by Qin in more detail.

Introduction to the Chinese Restaurant Process

The CRP is an alternative formulation of the Dirichlet process mixture model. It provides a clustering method that determines the number of clusters from the data. The underlying generative process can be imagined as a Chinese restaurant seating procedure, which supposedly was experienced by Jim Pitman [Aldo 85].

Let's assume an infinitely large Chinese restaurant with an infinite number of round tables. Each table is endowed with a dish (parameter for Dirichlet distribution). Customers enter the restaurant in a sequence and are seated at a randomly chosen table. The first customer naturally sits at an empty table. Every other customer either sits at an unoccupied table with probability to a scaling parameter or at an occupied table with probability proportional to the number of customers seated at that table. The result provides a clustering of the data, whereas each customer draws an observation from the distribution assigned to his or her table (the dish) [Aldo 85, Frig 10, Ghos 11, Blei 11]. Two example results of the seating procedure for seven customers using four tables is given in Figure 3.19.

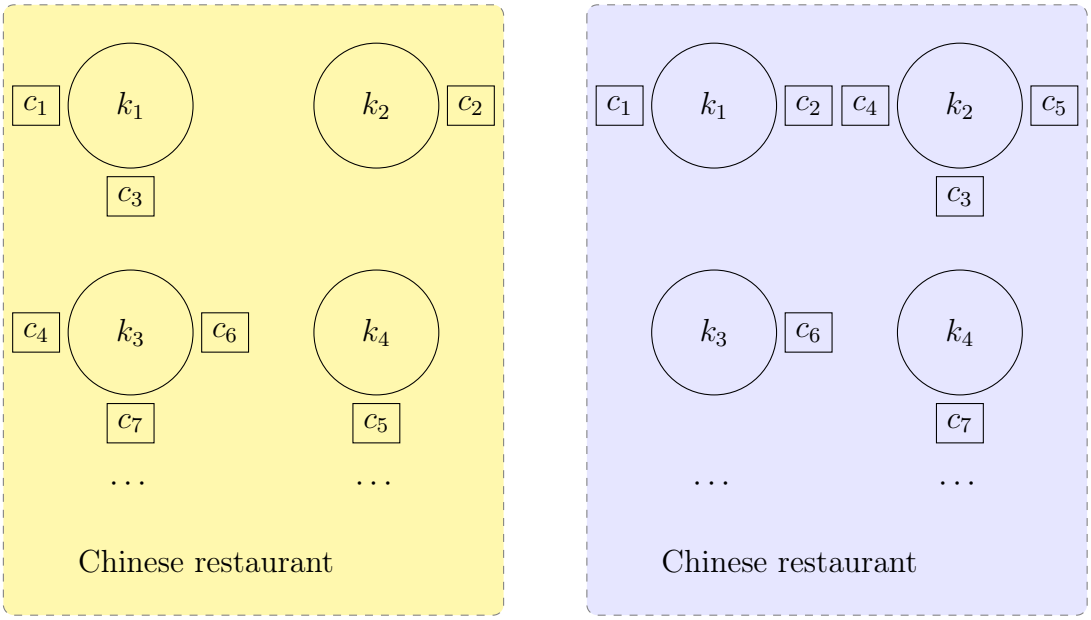


Figure 3.19: Two possible outcomes of the CRP for seven customers $C = \{c_1, c_2, \ldots, c_7\}$ using four tables $K = \{k_1, k_2, k_3, k_4\}$. The first partition (1,3)(2)(4,6,7)(5) and the second partition (1,2)(3,4,5)(6)(7) are different, but according to the Chinese restaurant process their probability is equal.

Terms and Definitions

After the informal introduction of the CRP, here, the underlying definitions and terms are introduced.

Definition 1 (Exchangebility)**.** [Aldo 85, Pitm 95] A finite or infinite sequence of random variables X_i is called **exchangeable** (or N-exchangeable, to indicate the number of random variables) if

$$(X_1, X_2, \ldots) = (X_{\pi(1)}, X_{\pi(2)}, \ldots), \tag{3.42}$$

for each permutation of π. The joint probability distribution of the permuted sequence is the same as of the original sequence. For example, independent and identically distributed random variables are exchangeable.

Definition 2 (DeFinetti's theorem)**.** [Jord 05] If $(X_1, \ldots, X_N)$ are infinitely exchangeable, then the joint probability $p(X_1, \ldots, X_N)$ has a representation as a mixture:

$$p(X_1, \ldots, X_N) = \int \left(\prod_{i=1}^{N} p(X_i|\theta) \right) dP(\theta)$$

for some random variable θ. Note that θ is not limited and can be range over measures, in which case $P(\theta)$ is a distribution on measures. The Dirichlet process is an example of a distribution over measures.

Definition 3 (Dirichlet distribution)**.** [Bish 06, Ferg 73] A Dirichlet distribution is a family of continuous multivariate probability distributions over $K \in \mathbb{N}$ random variables. The random variables are subject to the constraints:

$$0 \leq X_k \leq 1 \quad \text{and} \quad \sum_{k=1}^{K} X_k = 1.$$

If we denote,

$$\boldsymbol{X} = (X_1, \ldots, X_K)^T \quad \text{and} \quad \boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K),$$

then the Dirichlet distribution is given by

$$\text{Dir}\,(\boldsymbol{X} \mid \boldsymbol{\alpha}) = \frac{\Gamma(\alpha_0)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} X_k^{\alpha_k - 1}. \tag{3.43}$$

$\Gamma(\alpha)$ denotes the gamma function and α_0 is the sum of concentration vector $\boldsymbol{\alpha}$

$$\alpha_0 = \sum_{k=1}^{K} \alpha_k.$$

The mean of the Dirichlet distribution is $\boldsymbol{\alpha}/\alpha_0$. The Dirichlet distribution represents a generalization of the Beta distribution. For $K = 2$ they are identical. Thus, the Dirichlet distribution has a finite density everywhere, provided $\alpha_k \geq 1$ for all k.

Example: Let $K = 6$ and imagine it as an ordinary die. To sample $\boldsymbol{X}$, you can roll the die and produce a number in the range of one to six. In case of a huge number of dice, the Dirichlet distribution could describe the distribution of dicing results. If the dice are very good, we would have a uniform distribution described by a

parameter / concentration vector $\boldsymbol{\alpha} = (1000, 1000, 1000, 1000, 1000, 1000)^T$, meaning that each number has the same probability. Older or battered dice could produce a Dirichlet distribution with $\boldsymbol{\alpha} = (1000, 500, 500, 500, 500, 500)^T$, meaning that the number one has a doubled probability compared to the other numbers. Furthermore, large α-values indicate a small variance.

Definition 4 (Dirichlet process)**.** [Ferg 73, Frig 10] A Dirichlet process is a stochastic process over a set Ω whose sample path is a probability distribution on Ω, where the sample path $\boldsymbol{X}$ is an infinite-dimensional set of random variables drawn from the process. The finite dimensional distributions are from the Dirichlet distribution, if

- H is a finite measure on Ω,
- α is a positive real number: $\alpha \in \mathbb{R}^+$ and
- $\boldsymbol{X}$ is a sample path drawn from a Dirichlet process: $\boldsymbol{X} \sim \mathrm{DP}(\alpha, H)$,

then for any partition of Ω, say $\{C_k\}_{k=1}^K$, we have that

$$(\boldsymbol{X}(C_1), \ldots, \boldsymbol{X}(C_K)) \sim \text{Dirichlet}\left(\alpha H(C_1), \ldots, \alpha H(C_K)\right).$$

Note: The Dirichlet process is a stochastic process. Thus, it creates an infinite sequence of random variables, rather than a single random distribution.

The Chinese Restaurant Process view on the Dirichlet Process

Let's assume a finite-dimensional mixture model with mixing probabilities $\boldsymbol{\beta}$ drawn from a Dirichlet distribution. We are only interested in the particular assignment of a component y_i, which is conditioned on all previous assignments. If $\boldsymbol{\beta}$ is marginalized out, we have a Dirichlet-multinomial distribution. Note: Marginalizing out $\boldsymbol{\beta}$ results in having dependencies between the components. Since, we have a Dirichlet-multinomial distribution, we can use

$$p(y_i = k \mid y_1, \ldots, y_{i-1}, \alpha, K) = \frac{n_k + \alpha/K}{i - 1 + \alpha}, \tag{3.44}$$

where $k = 1, \ldots, K$ is a particular value of y_i and n_k is the number of times a customer was seated at table k in the sequence $\{y_1, \ldots, y_{i-1}\}$. Thus, the probability of observing a component (table) is proportional to the number of previous observations of this component.

Let's assume the infinite restaurant: $K \to \infty$. Then Eq. 3.44 results in

$$p(y_i = k \mid y_1, \ldots, y_{i-1}, \alpha) = \lim_{K \to \infty} \frac{n_k + \alpha/K}{i - 1 + \alpha} = \frac{n_k}{i - 1 + \alpha}. \tag{3.45}$$

Eq. 3.45 is the most common form to describe the probability of seating a new customer at an already occupied table. The equation consists only of one adjustable parameter α, commonly called the concentration parameter. Another variant, the so-called two parameter model, uses two parameters: α and β, commonly called discount and strength [Pitm 06].

The so far shown probability for seating customers at occupied tables follows the rich get richer scheme. If we now think about the probability of seating a customer at an empty table k, $n_k = 0$ (sample of an so far new component) for $K \to \infty$, then it is obvious that the probability goes to 0. Hence, we have to consider *all* empty tables. Let's summarize all empty tables in the set T. Let's denote the number of all occupied tables with J, then the number of unseen tables is $|T| = K - J$. Inserting this information into Eq. 3.44, the probability of seating a customer at an empty table is

$$\begin{aligned} p(y_i \in T \mid y_1, \ldots, y_{i-1}, \alpha) &= \lim_{K\to\infty} \sum_{t\in T} \frac{\alpha/K}{i-1+\alpha} \\ &= \lim_{K\to\infty} \frac{\alpha}{i-1+\alpha} \cdot |T| \cdot \frac{1}{K} \\ &= \frac{\alpha}{i-1+\alpha} \lim_{K\to\infty} \frac{K-J}{K} \\ &= \frac{\alpha}{i-1+\alpha}. \end{aligned} \tag{3.46}$$

Eq. 3.45 and 3.46 proof the properties of the CRP stated in the beginning of Section 3.3.5.

1. The probability of seating a customer at an occupied table is proportional to the customers already seated at that table.

2. The probability of seating a customer at an empty table is proportional to the concentration parameter α.

Algorithm 5: Gibbs weighted Chinese restaurant process

Input : N customers
Output: Number of clusters K and cluster assignments $\boldsymbol{Y}$

```
Randomly assign N customers to k tables, with 1 ≤ k ≤ N;
while not converged do
    for i ← 1 to N do
        Remove customer i from its current table;
        Reseat customer i based on Eq. (3.45) and (3.46);
        Compute likelihood of solution, see Eq. 3.51;
    end
end
```

Exchangebility, DeFinetti and the Gibbs Weighted Chinese Restaurant Process

The CRP discussion so far focused mainly on the number of customers at a specific table and the concentration parameter α. For the purpose of clustering ear impressions, the Gibbs weighted Chinese restaurant process (GWCRP) [Qin 06, Lau 07] features

useful properties. It uses the fact that the order in which customers arrive is irrelevant due to the exchangeability. Hence, anybody could have been the last customer and its placement could be predicted using the placements of all $N-1$ previous customers. The GWCRP makes use of this, by starting with a randomly initialized restaurant. Then the algorithm iterates over all customers. Thus, each customer is removed from the table and immediately reseated. The iteration continues until the partition is stable. The Algorithm is summarized in Algorithm 5.

CRC Using Gaussian Mixture Models In the following, we use the approach proposed by Qin [Qin 06] and use Gaussian mixture models and assume that our observations $\boldsymbol{X}$ are independent. Hence, if the observations are independent, then the likelihood can be multiplied. Furthermore, we assume that samples in the same partition / table k share the same normal distribution:

$$x_{ij} = \mathcal{N}(\mu_{kj}, \sigma_{kj}^2). \tag{3.47}$$

Note: In the work by Qin, $i = 1, \ldots, N$ denotes genes and $j = 1, \ldots, M$ denotes experiments. In our setting $x_{ij} = s_{ij}$, where s_{ij} is an entry of the similarity matrix $\boldsymbol{S}$, see Eq. 3.27. Following these assumptions and remembering DeFinetti's theorem, the likelihood equation is as follows [Qin 06]:

$$p(\boldsymbol{X}|\boldsymbol{B}) \propto \prod_{k=1}^{K} \prod_{y_{ik}=1} \prod_{j=1}^{M} \left(\left(\sigma_{kj}^2\right)^{-\frac{1}{2}} \exp\left(-\frac{1}{2\sigma_{kj}^2} \left(x_{ij} - \mu_{kj}\right)^2 \right) \right). \tag{3.48}$$

In the above equation, $\boldsymbol{B}$ consists of the cluster assignments, as well as the means and standard deviations of the normal distributions. The marginal likelihood is computed by integrating out the parameters μ_{kj} and σ_{kj}^2. The necessary priors are defined as follows:

$$\begin{aligned} p(\mu_{kj}|\sigma_{kj}^2) &\sim \mathcal{N}(\mu_0, \sigma^2), \\ p(\sigma_{kj}^2) &\sim \text{Inv-Gamma}(\alpha, \beta). \end{aligned} \tag{3.49}$$

$$
\begin{aligned}
p(\boldsymbol{X}\,|\,\boldsymbol{Y}) &= \prod_{k=1}^{K}\prod_{j=1}^{M}\iint\prod_{y_{ik}=1} p(x_{ij}|\mu_{kj},\sigma_{kj}^2)p(\mu_{kj}|\mu_0,\sigma_{kj}^2)p(\sigma_{kj}^2)d\mu_{kj}d\sigma_{kj}^2 \\
&= \prod_{k=1}^{K}\prod_{j=1}^{M}\iint \underbrace{\left(\frac{1}{2\pi\sigma_{kj}^2}\right)^{-\frac{n_k}{2}}\exp\left(-\frac{1}{2\sigma_{kj}^2}\sum_{y_i=k}(x_{ij}-\mu_{kj})^2\right)}_{p(x_{ij}|\mu_{kj},\sigma_{kj}^2)\quad\text{Gaussian pdf}}\cdot \\
&\quad \underbrace{\left(\frac{1}{2\pi\sigma_{kj}^2}\right)^{-\frac{1}{2}}\exp\left(-\frac{1}{2\sigma_{kj}^2}(\mu_{kj}-\mu_0)^2\right)}_{p(\mu_{kj}|\mu_0,\sigma_{kj}^2)\quad\text{Gaussian pdf}}\cdot \\
&\quad \underbrace{\frac{\beta^\alpha}{\Gamma(\alpha)}\left(\sigma_{kj}^2\right)^{-(\alpha+1)}\exp\left(-\frac{\beta}{\sigma_{kj}^2}\right)}_{p(\sigma_{kj}^2)\quad\text{Inverse Gamma pdf}} d\mu_{kj}d\sigma_{kj}^2 \\
&= \prod_{k=1}^{K}\prod_{j=1}^{M}\left[\frac{\beta^\alpha}{\Gamma(\alpha)}\frac{(2\pi)^{-\frac{n_k}{2}}}{\sqrt{n_k+1}}\frac{\Gamma\left(\frac{n_k}{2}+\alpha\right)}{\left(\beta+\frac{1}{2}\left(\sum_{y_i=k}x_{ij}^2+\mu_0^2-\frac{\left(\sum_{y_i=k}x_{ij}+\mu_0\right)^2}{n_k+1}\right)\right)^{\frac{n_k}{2}+\alpha}}\right]
\end{aligned}
\tag{3.50}
$$

The likelihood of sample i belonging to cluster k is then given by

$$
\begin{aligned}
&\frac{p(\boldsymbol{X}|y_{ik}=1)}{p(\boldsymbol{X}|y_{ik}=0)} = \frac{\Gamma(\alpha)\sqrt{2N}}{\beta^\alpha\sqrt{N+1}}\cdot \\
&\frac{\Gamma\left(\alpha+\frac{N}{2}\right)\left(\beta+\frac{1}{2}\left(\sum_{i=2}^{N}x_{ik}^2+\mu_0^2-\frac{\left(\sum_{i=2}^{N}x_{ik}+\mu_0\right)^2}{N}\right)\right)^{\alpha+\frac{N-1}{2}}\left(\beta+\frac{(x_{1k}-\mu_0)^2}{2}\right)^{\alpha+\frac{1}{2}}}{\Gamma\left(\alpha+\frac{N-1}{2}\right)\Gamma\left(\alpha+\frac{1}{2}\right)\left(\beta+\frac{1}{2}\left(\sum_{i=1}^{N}x_{ik}^2+\mu_0^2-\frac{\left(\sum_{i=1}^{N}x_{ik}+\mu_0\right)^2}{N+1}\right)\right)^{\alpha+\frac{N}{2}}}
\end{aligned}
\tag{3.51}
$$

The remaining parameters of the prior distributions are μ_0, the shape α and the scale β parameter of the Inverse Gamma pdf. The μ_0 is given as the mean error of all sample alignments $\mu_0 = \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}x_{ij}$. The parameter α is set to 1 and β is the doubled standard deviation of all x_{ij}'s. Note that these prior distributions are data dependent. The prior distribution for the cluster indicator is given by the CRP process described in Eqs. (3.45) and (3.46). The above setting was implemented and described in detail by Zhaohui S. Qin in [Qin 06].

The different results of the three employed clustering techniques are evaluated in Section 3.4.2. More information about the three applied clustering methods, including evaluations and comparisons can be found in [Band 07] for CRC and an exhaustive review of clustering methods is given in [Jain 99].

3.3.6 Usage of Clustering-based Features

After a representative set is found with help of the data clustering and subsequently labeled by an expert, the last step is to transfer or project the labeled features to an unlabeled ear impression (Figure 3.15).

To detect the features on a, so far, unknown ear impression S_{new} it is registered with each surface in $\mathcal{S}_{\text{rep}}$. In the following only surface $S_{\text{best}} \in \mathcal{S}_{\text{rep}}$ having the smallest registration error is considered. The features $\mathcal{F}_{\text{best}}$ of S_{best} are transformed using the transformation matrix $\boldsymbol{T}_{\text{best}}$ resulting in a new feature set $\mathcal{F}'$. Since the transformed features $\mathcal{F}'$ typically will not end up directly on the surface S_{new} a final projection is necessary.

So far, we employ only simple projection algorithms. In case of feature points or point sets, the projection is achieved by projecting the given point on the closest point of S_{new}. According to the feature class, an optimization step using a small neighborhood is done. For example, the helix-peak is set to the highest point of a 3 mm neighborhood of the transformed and projected helix-peak. In case of feature planes, no projection is necessary. If a ridge feature needs to be projected, the problem is reduced to the projection of the start and end point, followed by running the ridge detection algorithm.

3.4 Experiments and Results

The experiments section is divided into three parts. First, the evaluation of the developed ICP is carried out. The evaluation focuses on the identification of the best parameters for point selection, point pair rejection and point pair weighting. Second, the results of the surface-based feature detection are stated. Third, the clustering-based feature detection is evaluated and compared to the surface-based feature detection.

3.4.1 Evaluation of the ICP Algorithm for Ear Impressions

Experimental Data

For the evaluation of the ICP, a sample set $\mathcal{S}$ of 400 ear impressions was prepared. The sample set was distorted in two ways yielding $\mathcal{S}_{\text{cut}}$ and $\mathcal{S}_{\text{rot}}$.

Each mesh in $\mathcal{S}_{\text{cut}}$ was cut at the bottom with a slanted cut, removing approximately 25 % of the surface, see Figure 3.20. Each mesh in $\mathcal{S}_{\text{rot}}$ was rotated about 10°. In addition, each vertex $\boldsymbol{x} \in S_i$ was randomly displaced within a small distance range:

$$\boldsymbol{x}_i' = \boldsymbol{x}_i + n^2 \left(\boldsymbol{r} - \mathbf{1}\right). \tag{3.52}$$

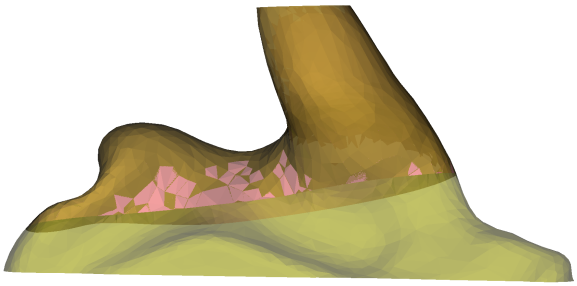
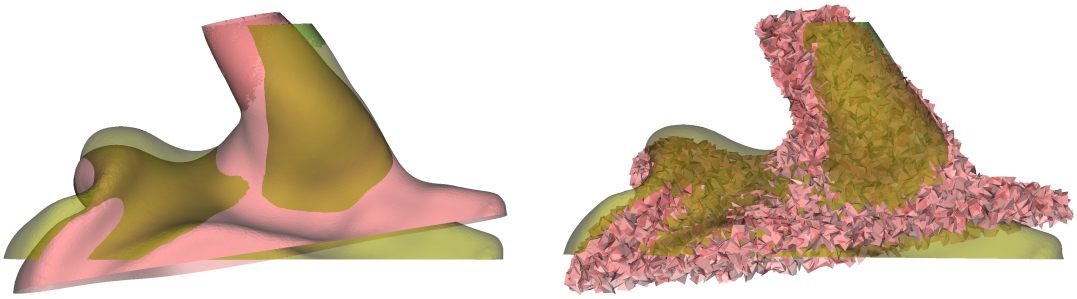


Figure 3.20: Sample of sample set $\mathcal{S}_{\text{cut}}$ along with the original sample.



(a) Sample with applied rotation. (b) Sample with applied rotation and noise.

Figure 3.21: Sample of sample set $\mathcal{S}_{\text{rot}}$ along with the original sample.

In Eq. (3.52), $\boldsymbol{x}_i$ denotes a vertex of a mesh $S \in \mathcal{S}_{\text{rot}}$, n denotes the noise power (typically 0.75), $\boldsymbol{r} = (s, s, s)^T$ is a random vector with $s \in [0, 1]$ and $\mathbf{1} = (1, 1, 1)^T$. An example for the applied rotation and noise is shown in Figure 3.21.

The combination of the modifications applied to $\mathcal{S}_{\text{cut}}$ and $\mathcal{S}_{\text{rot}}$ simulate the expected different orientations, measurement noise and non overlapping contained in the data. During the evaluation, $\mathcal{S}_{\text{cut}}$ and $\mathcal{S}_{\text{rot}}$ are used for the alignment, while the original mesh $S \in \mathcal{S}$ is used to compute the alignment error.

Evaluation

In the following experiments, we focus on the final alignment using the initial registration provided by the centerline alignment. We evaluate, how to select the points, what kind of rejection technique works best and if point-pair weighting improves the registration result. On the basis of preliminary experiments, we used the settings shown in Table 3.2 as default settings.

Selection of Points In order to speed up the alignment, the meshes are subsampled. We compared random and uniform point selection. We also computed the alignment error for the original mesh as reference. The results are listed in Table 3.3 and visualized in Figure 3.22.

They clearly indicate that the uniform selection of points is far worse than selection of random points. Naturally, the selection of all points yields the best result. However, the random selection error is only slightly larger, while maintaining a reasonable alignment speed. Random selection is approximately 16 times faster than using the

Parameter	Default value
number of initial transformations	10, $\mathcal{T} = \{\boldsymbol{T}_1, \ldots, \boldsymbol{T}_{10}\}$
sub-sampling	1000 vertices
selection	random
rejection	two thresholds: $\theta_{\max,1} = 2.5$ and $\theta_{\max,2} = 1.0$
weighting	no weighting

Table 3.2: Default values for ICP experiments.

Selection	Error in mm	Computation time in sec	Average # point pairs
all	0.0244	25.8	13182.1
random	0.0478	1.6	748.2
uniform	0.0657	1.7	719.4

Table 3.3: Results of the different point selection strategies, showing the average registration error and the average computation time. Note: The average computation time includes data loading, data modification, centerline alignment and final registration.

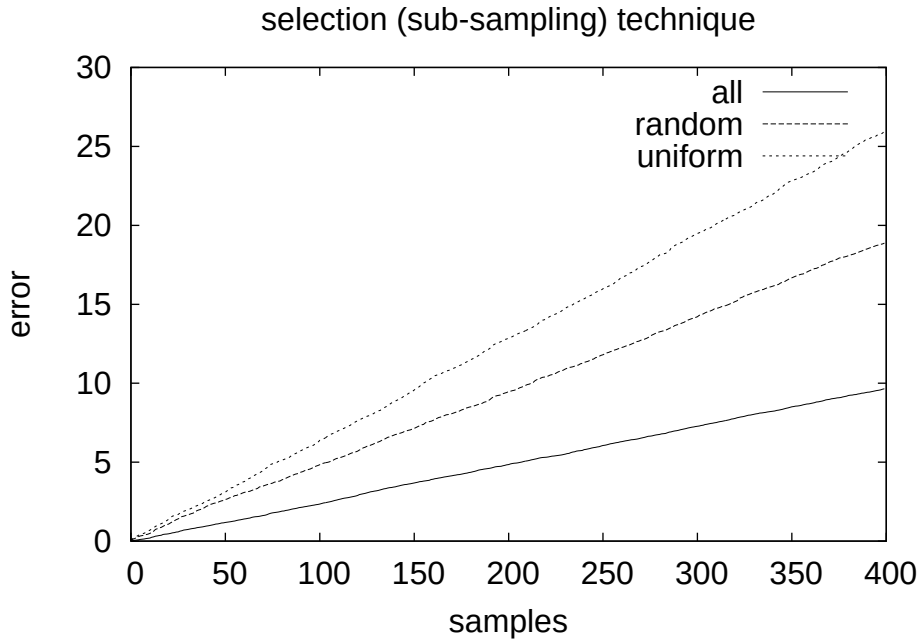


Figure 3.22: Comparison of the cumulative registration error of the three selection techniques.

full mesh. Based on these results, the random point selection, it is used exclusively in the following.

Rejection of Point Pairs The ICP contains the implicit assumption that both meshes overlap completely and that the points are exact rather than measured. To

handle these assumptions a maximum matching threshold $\theta_{\max}$ is applied [Sega 09, Rusi 01]. We compared five rejection strategies, including not using rejection at all.

The simplest one is rejection of point-pairs based on a threshold value $\theta_{\max}$. We investigated two variations. The first one uses a single full ICP iteration with a medium threshold $\theta_{\max} = 2\,\text{mm}$. The second one uses two thresholds, large threshold $\theta_{\max,1} = 2.5\,\text{mm}$ and a small $\theta_{\max,2} = 1.0\,\text{mm}$. It performs two ICP iterations. For the second round, the already achieved alignment is used as starting point. In addition, we evaluated the rejection of the worst pairs and rejection based on the standard deviation of the point pair distance [Rusi 01]. In the former, the the worst 10 % are excluded and in the latter point-pairs with a distance greater than $\theta_{\max} = 2.5\sigma$ are excluded. Here, σ denotes the standard deviation of the Euclidean point pair distances. The results are listed in Table 3.4 and visualized in Figure 3.23.

Rejection	Error in mm	Computation time in sec	Average # point pairs
no rejection	1.1088	1.8	1000
one threshold	0.0800	1.1	814.6
two thresholds	0.0478	1.7	749.1
worst pairs	0.0488	1.2	732.9
standard deviation	0.0491	1.1	748.7

Table 3.4: Results of point rejection strategy showing the average registration error and the average computation time for each registration in seconds

.

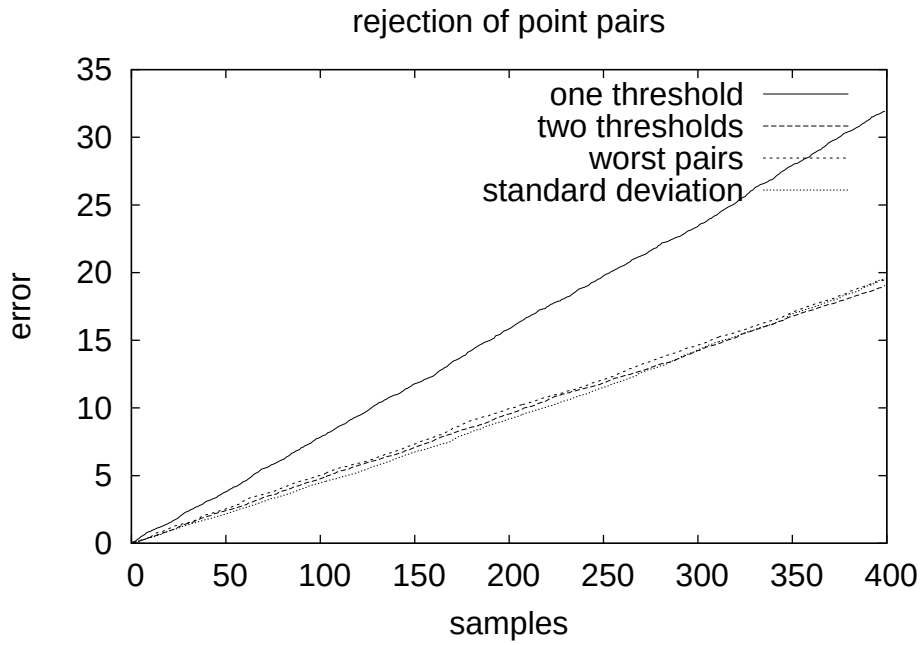


Figure 3.23: Comparison of the cumulative error of the rejection strategies.

It is obvious from the results in Table 3.4 that a rejection strategy is necessary. The large error in case of no rejection is due to numerical problems in the Cholesky decom-

position of $\boldsymbol{A}$, see Eq. (3.22). The simple rejection strategy with a single threshold works better than no rejection, but is significantly worse than the more adaptive techniques. The best performing method uses two thresholds, but its accuracy is expensive considering the average time needed to compute the registration. The best trade off considering time and accuracy is offered by the worst pairs and standard deviation rejection. In the following experiments, the two thresholds technique is used when time is not critical and the worst pairs technique otherwise.

Weighting of Point Pairs It is intuitive to weight good (close to each other) point-pairs higher than bad (far from each other) point-pairs. The weighting is similar to rejection, but it allows smoother adjustments. Considering the results of Rusinkiewicz and Levoy [Rusi 01] it is expected that with a good rejection strategy employed the influence of point pair weighting is small. The weighting is achieved by adjusting the weights in the objective function at line 12 in Algorithm 1. We compared three different weighting techniques, no weighting, weighting using a distance penalty $\omega_i = 1 - \frac{d(\boldsymbol{p}_i, \boldsymbol{q}_i)}{d_{\max}}$, where $d_{\max}$ is the largest distance of all point-pairs and normal compatibility $\omega_i = \boldsymbol{n}_i^{(p)T} \boldsymbol{n}_i^{(q)}$. The results are listed in Table 3.5 and visualized in Figure 3.24.

Weighting	Error in mm	Error without noise	Computation time in sec
no weighting	0.0467	0.0115	1.7 seconds
distance penalty	0.0471	0.0056	1.7 seconds
normal compatibility	0.0502	0.0106	1.7 seconds

Table 3.5: Results of point weighting strategy showing the average registration error (in case of noise and no noise) and the average computation time in seconds. The average number of point pairs is approximately 749.

In comparison to the rejection, the influence of the weighting strategy is low. That is due to the reason of the pre-alignment and a suitable rejection technique in place. In case of applied noise, the weighting did have a negative effect on the registration result. This can be explained with the sensitivity to noise of both weighting techniques. It is especially notable in case of the normal compatibility, because it directly uses the noisy normals ($\omega_i = \boldsymbol{n}_i^{(p)T} \boldsymbol{n}_i^{(q)}$).

Hence, the experiment was repeated without applying random noise. This time, the results show the benefits of the weighting strategies. The distance weighting halves the registration error and also the normal compatibility weighting slightly improves the result.

Summary of ICP Evaluation for Ear Impressions

The developed two-step ICP for ear impressions achieved very good results on the sample set. We evaluated different methods for point selection, point pair rejection and point pair weighting. The results showed that it is necessary to employ a rejection strategy. In our case a thresholding technique using two threshold works best.

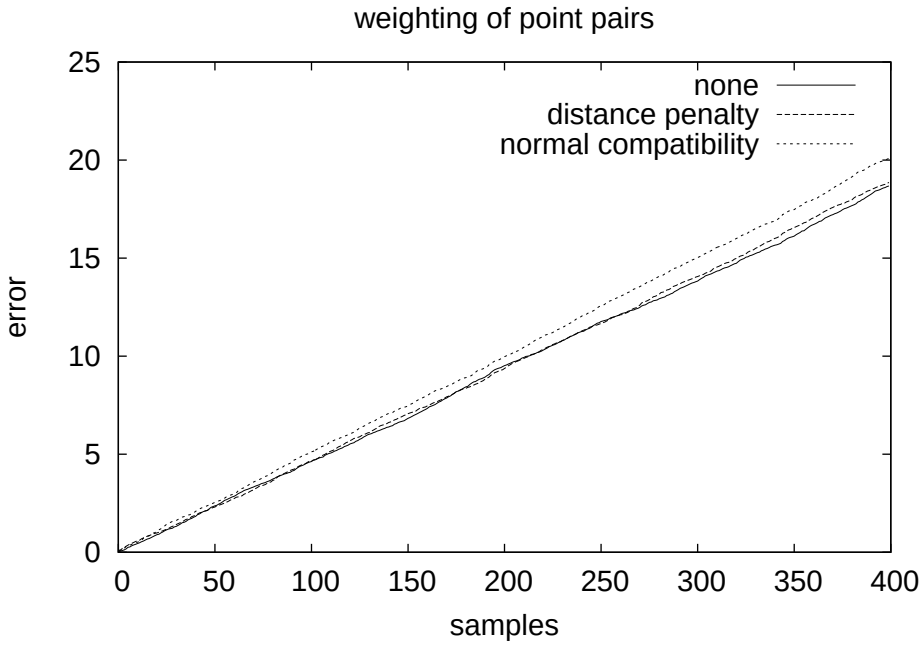


Figure 3.24: Results of point weighting strategy showing the summed up error.

Considering the point selection, it is beneficial to use a random selection, which performed far better than a uniform selection of points. An unexpected result is the effect of the point pair weighting. Since the weighting is sensitive to noise, it did not improve the registration error with the given data, but had a negative effect. Nevertheless, in case of noise free data, it worked well.

3.4.2 Experiments for Identification of a Representative Sample Set

Experimental Data

For the identification of a representative set of ear impressions a data set $\mathcal{S}$ of 473 impressions was used. The sample set contains all kinds of difficult and unusual ear impressions, such as the ones shown in Figures 3.13 and 3.14.

In a pre-processing step the similarity matrix $\boldsymbol{S}$ is filled (Eq.(3.27)). Therefore, $s_{i,j} \in \boldsymbol{S}$ denotes the alignment error between sample $S_i \in \mathcal{S}$ and $S_j \in \mathcal{S}$. To compute the registrations to fill $\boldsymbol{S}$ the settings listed in Table 3.2 are used.

Agglomerative Hierarchical Clustering

In our experiments, we generated partitions using the four linkage criteria defined in Eq. (3.28) to (3.31). The generated partitions are quite similar as can be seen in Table 3.6. The number of clusters is nine with the exception of the mean linkage criterion, which results in 13 clusters. For all criteria the clusters are well sized in the range of 26 to 77 samples. Comparing the intra and inter class correlation shows

little difference. The similarity of the partitions is further confirmed by the rand index [Rand 71] shown in Table 3.7.

Linkage	# Clusters	Mean size	Intra correlation	Inter correlation
D_{A}	9	52.6	0.1039	0.0516
D_{M}	13	36.4	0.1052	0.0446
D_{NN}	9	52.6	0.1077	0.0388
D_{FN}	9	52.6	0.1117	0.0408

Table 3.6: Results of AHC for different linkage criteria.

In Figure 3.25, three cluster centroids of the partition generated by the farthest neighbor criterion are shown. Comparing the distinctive features ear canal, anti-tragus concavity and helix for the three depicted centroids, the following observations can be made.

- Cluster A features a thin ear canal with a pronounced second bend. Furthermore, it posses a short but wide helix. Also distinctive is the deep anti-tragus concavity.
- Cluster B posses a thicker canal, but has a less pronounced second bend than cluster A. The anti-tragus concavity is much less visible compared to cluster A or C. Moreover, the crus valley is deeper and better visible compared to A.
- Cluster C exhibits a thick ear canal with a almost invisible second bend. Most distinctive for this cluster is the long narrow helix. In addition, it features a very pronounced anti-tragus concavity.

We conclude that the acquired partition is feasible and can be used as a representative set.

Expectation Maximization Clustering

For the EMC experiments the weka implementation was used, see Algorithm 4 on page 56. To take into account the random effects of the implementation, the experiment was repeated ten times. However, the results were identically.
The resulting partition consists of nine clusters, with an average of 52.56 samples per cluster and a small standard deviation of 5.097. The resulting clusters allow similar observations as for the AHC result.

Rand index	D_{A}	D_{M}	D_{NN}	D_{FN}
D_{A}	1	0.8425	0.8171	0.8121
D_{M}	-	1	0.8451	0.8366
D_{NN}	-	-	1	0.8124
D_{FN}	-	-	-	1

Table 3.7: Rand index computed pairwise for the AHC clustering results generated by different linkage criterias.

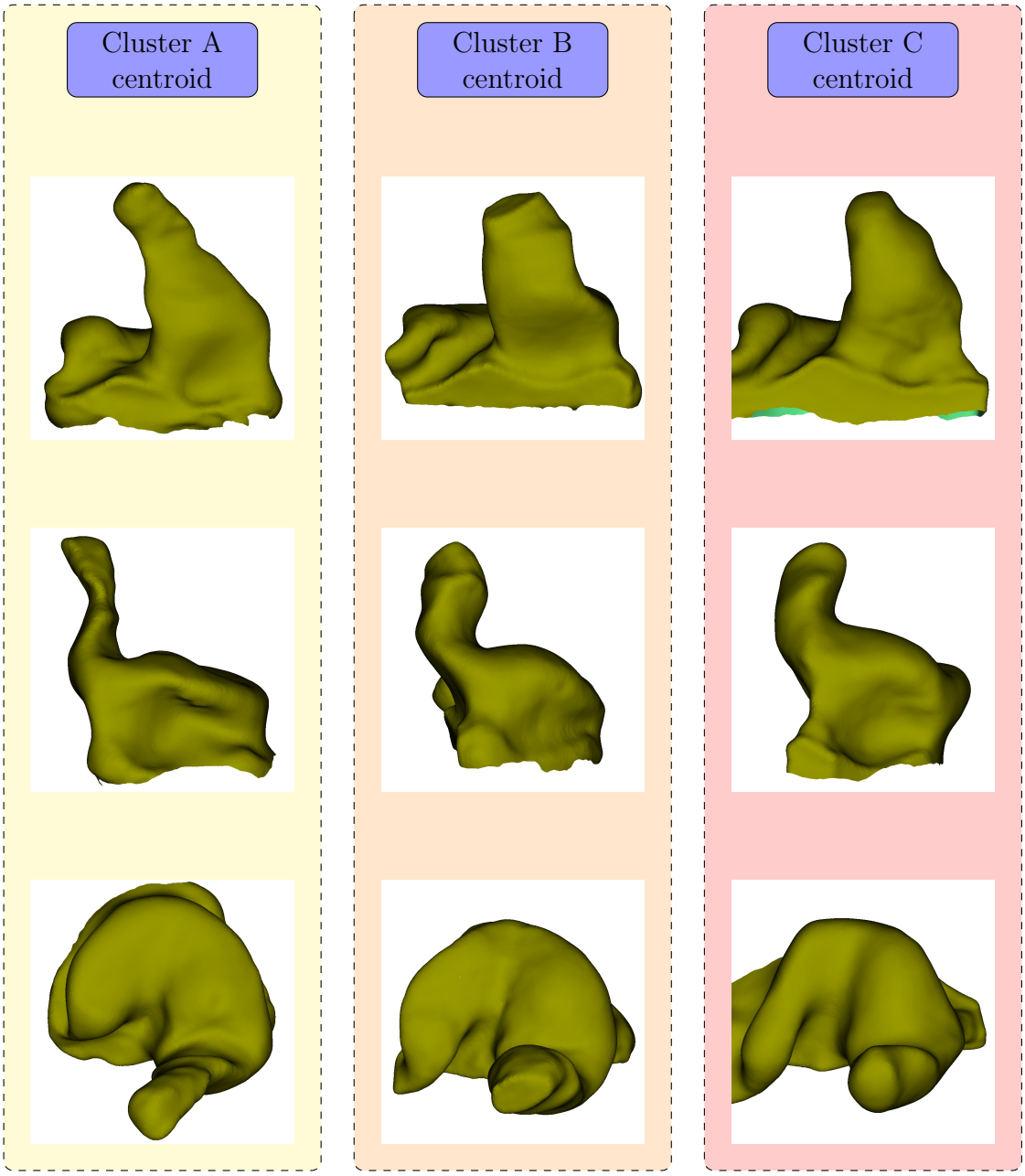


Figure 3.25: The AHC with farthest neighbor criterion resulted in nine clusters. Three of the nine cluster centroids are displayed here. Each of the centroids is shown from different view points to emphasize the differences in their shapes.

Chinese Restaurant Clustering

For the CRC experiments we relied on the implementation by Qin [Qin 06] as described in Section 3.3.2 on page 56. Since it is a probabilistic based approach, we repeated the experiments ten times using the proposed standard values of the algorithm. The results varied between eight and eleven clusters, as can be seen in Table 3.8.

Experiment	# clusters	Mean size	Log-likelihood
1	9	52.6	265782
2	9	52.6	265748
3	9	52.6	265762
4	10	47.3	265718
5	10	47.3	265596
6	11	43.0	265509
7	8	59.1	265855
8	10	47.3	265675
9	11	43.0	265259
10	10	47.3	265699

Table 3.8: Results of clustering experiments using the CRC.

	Rand index	Adjusted rand index
Lowest value	0.9048	0.4854
Largest value	0.9823	0.9034

Table 3.9: Range of rand and adjusted rand index for the ten experiments shown in Table 3.8.

On average a partition with ten clusters is created. The computed log-likelihood is similar in all experiments. This similarity is confirmed by the computed rand (RI) and adjusted rand index (ARI), which are presented in Table 3.9.

Comparison of all Clustering Approaches and Conclusions

In order to identify the set, which should be labeled and used as reference, we compared the different partitions created by AHC, EMC and CRC using the rand index. The results presented in Table 3.10 show that the partitions are similar in terms of the simple rand index. However, if the adjusted rand index is used, the clusterings differ significantly. The CRC and EMC can be still considered similar, while the AHC result stands for its own.

A visual inspections of the different partitions indicated that all algorithms generated feasible partitions of the data. For the following evaluation of the clustering-based feature detection, we will use the result of the CRC clustering as well as the

RI / ARI	AHC	EMC	CRC
AHC	1	0.8156	0.8201
EMC	0.0787	1	0.9061
CRC	0.0689	0.5035	1

Table 3.10: Rand and adjusted rand index for different clustering methods. The upper right triangle contains the values of the rand index, while the lower left triangle contains the values of the adjusted rand index.

AHC result. From our ten CRC experiments, we randomly pick one with ten clusters, since this corresponds to the average number of clusters generated by the CRC.

3.4.3 Evaluation of Surface-based Feature Detection

In [Balo 10a], Baloch et al. carried out an exhaustive evaluation of the surface-based feature detection. A dataset of 198 labeled impressions was acquired and statistical evaluation of the point and plane features was carried out relative to the ground truth. The curve features, on the other hand, were evaluated qualitatively by the experts, due to the absence of corresponding ground truth data. The error δ_{point} of the point features with the ground truth was computed as the Euclidean distance between them. For plane features, both the orientation and the location were considered. The orientation δ_{or} was compared via the angle between the plane normals. Deviation of plane locations δ_{loc} was determined by first finding the center of the intersection contours of the individual planes, followed by computing the mean of the distances of the centers from their counterpart planes. Area features were evaluated by way of the sensitivity δ_{sen} and specificity δ_{spe}. The results are listed in Table 3.11. They are averaged for all features defined in Table 3.1.

Feature type	Error measure	Mean	Standard deviation
plane	δ_{or}	8.1°	16.3°
	δ_{loc}	1.0 mm	1.21 mm
point	δ_{point}	1.98 mm	2.05 mm
area	δ_{sen}	0.83	-
	δ_{spe}	0.93	-

Table 3.11: Results of the surface-based feature detection. The mean and standard deviations are averaged for all features of the same feature type.

Modeling experts were consulted to provide reasonable error tolerances for the interpretation of the results. They specified a tolerance of 3 mm for point features and plane location. Furthermore, for plane orientation a tolerance of 15° was set. The error tolerances were then used to define two performance measures, namely the detection rate, and the tolerance rate. The detection rate Δ_{det} is defined as the percentage of test cases for which an algorithm successfully detected the corresponding feature. The tolerance rate Δ_{tol} is defined as the percentage of test cases for which the detected feature is within the acceptable tolerance. The results indicate acceptable performance for most features (overall mean tolerance rate of 87 % for points and 83 % for planes). The tolerance rate was computed over the cases, which passed the detection test. The results are provided in Table 3.12.

3.4.4 Evaluation of Clustering-based Feature Detection

For the evaluation of the clustering-based feature detection, we concentrate on the core features defined in Table 3.1. We conducted the experiments with two representative sets $\mathcal{S}_{\text{AHC}}$ and $\mathcal{S}_{\text{CRC}}$. The former denotes the cluster centers of the AHC

Performance measure	Feature type	Rate in %
Δ_{det}	point	98.6
	plane	98.8
Δ_{tol}	point	87.0
	plane	82.0

Table 3.12: Results of detection and tolerance rate analysis.

and the latter of the CRC computed in Section 3.3.2. To compare the performance of the feature detection approaches a sample set of 117 labeled ear impressions was acquired. The experts labeled the core feature points and planes. The experts occasionally missed to label a feature. Therefore, approximately 70 reference features are available per feature.

Core Feature Points

For the experiments, the same thresholds and error measures are used as in Section 3.4.3. The results are presented in Table 3.13. The clustering-based approach

	Surface		$\mathcal{S}_{\text{AHC}}$		$\mathcal{S}_{\text{CRC}}$	
Feature name	mean	stdev	mean	stdev	mean	stdev
concha-peak	3.4723	3.6784	2.9173	2.7680	2.9711	2.4448
helix-peak	3.6256	6.4196	3.1382	1.9133	2.6823	2.4088
tragus concavity	2.9067	2.5301	1.9379	1.1776	1.6504	1.1995
anti-tragus concavity	3.4016	3.1810	2.3798	2.0246	2.2709	1.4525
anti-helix concavity	4.6344	3.8631	3.1198	3.1331	3.4512	3.3742
low of aperture	2.8996	1.7168	2.0210	0.9729	2.3623	1.3596
crus-concha intersection	4.6090	5.7477	3.2506	1.6772	3.2049	1.6810
canal-concha intersection	4.4167	5.1365	2.5248	3.0288	2.2720	2.9825
canal-crus intersection	4.6372	6.6597	2.1077	1.1318	2.4223	1.2581
average	3.8448	4.3259	2.5997	1.9808	2.5875	2.0179

Table 3.13: Results of the surface-based and clustering-based feature detection. All values are given in mm and stdev denotes the standard deviation.

clearly outperforms the surface-based approach. The average deviation from the ground truth is reduced about 1.25 mm, which corresponds to an improvement of over 30 %. In addition, the standard deviation is reduced as well, corresponding to a higher accuracy of the detected features.

The improvement is statistically significant [Gell 74]. Analyzing the average results, a Student's t-test yields 2p-value of 0.032. In addition, the comparison of the variances using the F-distribution also results in a statistically significant improvement in respect to $p = 0.01$. Furthermore, improvements in detection and tolerance rate were achieved. Due to the detection process, the detection rate for the clustering-based approach is always 100 %. The tolerance rate, which expresses how well the

feature was detected is of major interest. Here, the clustering-based approach achieves an improvement about 7 - 9 %. Results are listed in Table 3.14.

Measure	Surface	$\mathcal{S}_{\text{AHC}}$	$\mathcal{S}_{\text{CRC}}$
Δ_{det}	98.8 %	100.0 %	100.0 %
Δ_{tol}	62.3 %	71.0 %	69.3 %

Table 3.14: Performance measures for surface-based and clustering-based feature detection. Note: The large difference in the Δ_{tol} reported here and in Table 3.12 is due to the missing reference values, which were replaced with the detected values in the former evaluation.

Both representative sets work equally well. There is no significant difference in the results confirming the conclusion that both clustering results are feasible made in Section 3.4.2.

Core Feature Planes

Similar to the core feature points the same thresholds and error measures are used as in Section 3.4.3. Unfortunately, the clustering-based feature detection is not working in general. From the three considered core feature planes only the crus-valley plane can be detected with sufficient accuracy. The other two planes, first bend and second bend, are not suited well for the clustering-based approach. The experimental results are listed in Table 3.15.

	Surface		$\mathcal{S}_{\text{AHC}}$		$\mathcal{S}_{\text{CRC}}$	
Feature name	mean	stdev	mean	stdev	mean	stdev
first bend	8.62°	23.83°	63.88°	165.35°	51.73°	118.70°
second bend	15.35°	19.16°	36.88°	30.46°	41.17°	35.40°
crus-valley plane	16.04°	26.27°	11.65°	14.11°	12.86°	14.82°
average	13.34°	23.09°	37.47°	69.97°	35.25°	56.31°
first bend	1.52 mm	5.18 mm	14.15 mm	38.74 mm	9.17 mm	23.02 mm
second bend	1.91 mm	2.58 mm	2.78 mm	3.32 mm	2.15 mm	2.15 mm
crus-valley plane	2.77 mm	5.31 mm	2.66 mm	4.28 mm	3.15 mm	4.94 mm
average	2.07 mm	4.36 mm	6.53 mm	15.45 mm	4.82 mm	10.33 mm

Table 3.15: Results of the surface-based and clustering-based feature detection. Error in respect of plane orientation and location is given, stdev denotes the standard deviation.

The reason for the failure of the clustering-based approach is due to the special location of the first bend and second bend on the ear canal. Here, the orientation can vary excessively between different ear impressions. For example, Figure 3.26 shows two aligned impressions. The alignment error is small, but the ear canals are quite different when it comes to the orientation and properties of the bends. An interesting characteristic of the results is that the location of the second bend is typically good

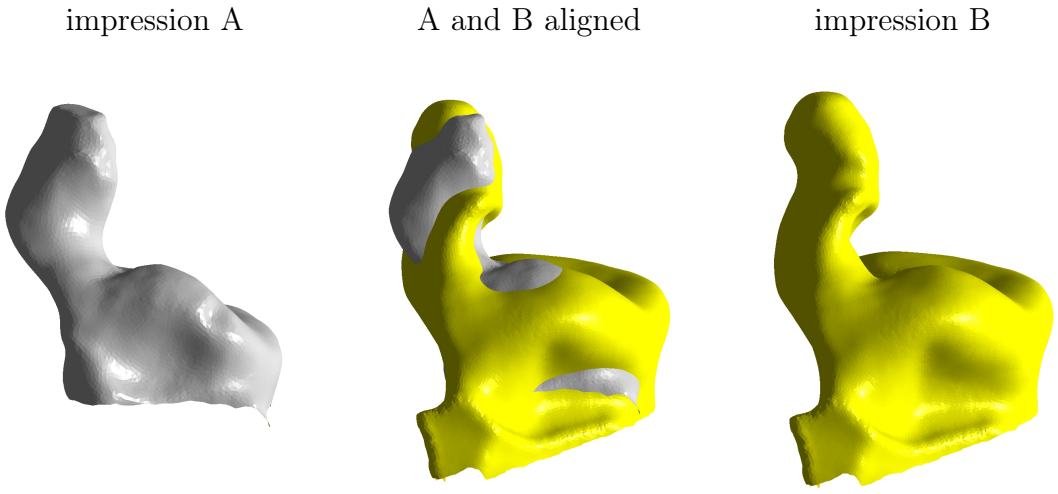


Figure 3.26: Two aligned ear impressions.

and the orientation is wrong. That allows to draw the conclusion that the location (or distance from shell body) of the second bend is stable, but the orientation of the bend varies strongly in the population. Considering the bends, it would be beneficial to first segment the ear canals and then carry out the clustering procedure. A segmentation approach for ear impressions was already presented by Zouhar et al. [Zouh 10]. Another approach would be the application of non-rigid registration as proposed by Unal et al. [Unal 11].

The crus-valley plane (see Figure 3.27) can be detected with a reasonable performance. Especially, the orientation is superior compared to the surface-based feature. In terms of location, both approaches are similar. Finally, considering the tolerance rate for orientation and location, more or less the same conclusions can be drawn. In Table 3.16 the orientation tolerance rate of the clustering-based approaches dominates the surface-based approach, while the location rate is dominated by the surface-based approach.

crus-valley plane	surface	$\mathcal{S}_{\text{AHC}}$	$\mathcal{S}_{\text{CRC}}$
$\Delta_{\text{tol,or}}$	62.8 %	77.8 %	68.1 %
$\Delta_{\text{tol,loc}}$	75.6 %	64.4 %	61.7 %

Table 3.16: Tolerance values for the detected crus-valley plane considering orientation and location.

The evaluation of region features was not carried out, because too less labeled regions were available. Preliminary experiments and the experience made during the point feature evaluation indicate that the clustering-based approach will perform well compared to the surface-based approach.

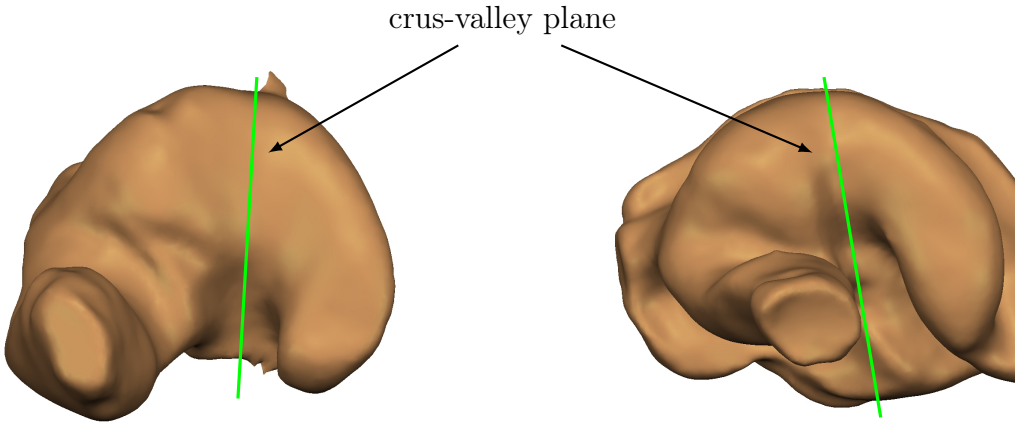


Figure 3.27: Examples of plane features.

3.5 Conclusions

The initially developed surface-based feature detection has an acceptable performance. Nevertheless, in case of bad or unusual ear impressions it can fail. The clustering-based approach has the advantage that the detection of each feature can be ensured. Hence, it can overcome the problem of missing features, which can be encountered using the surface-based detection approach.

Based on a large sample set, the superior performance of the clustering-based approach for feature points could be shown. In case of feature planes, the clustering-based approach failed concerning the bends on the ear canal, but it shows comparable performance for other feature planes. In summary, the developed clustering-based feature detection approach is more robust and accurate as the surface-based approach in terms of feature points and comparable in terms of feature planes.

Chapter 4

Rule-based Expert System

In this chapter, the RBES is introduced. At first the underlying KRL is presented. Thereby the motivation for developing yet another KRL is given and the implementation of the necessary parser and interpreter is discussed. The section is concluded by an overview of the capabilities of the resulting RBES. Thereafter the knowledge acquisition process is explained. The chapter concludes with experiments, to evaluate if the developed KRL and RBES are sufficient for the task. The developed RBES was published in the BVM 2009 proceedings [Sick 09] and the Computer-Aided Design journal [Sick 11a].

4.1 Script Language

4.1.1 Motivation and Requirements

In the AI community various languages and schemes to store knowledge have been developed, e. g. CLIPS, Lisp, Prologue, etc. They are usually firmly connected to a certain ES shell [Giar 05]. We decided to develop our own knowledge representation language (in the following referred to as *script language* or *script*). The reasons for developing yet another language are given by our requirements:

- The KRL is required to be simple and intuitive, allowing non-computer scientists and non-engineers to understand and extend the KB. Therefore, a simple syntax is required, preferably using a procedural representation with if-then-else conditions. For example, the popular ES shell CLIPS uses a Lisp like syntax, which is often difficult to follow for the (prosthesis design) experts [Rile 10].

- The KRL shall be efficient to parse and interpret. Furthermore, the grammar shall be easy to extend.

- The RBES has to support a semi-automatic mode, meaning that an expert is able to step in anytime to correct or modify a certain rule execution. In addition, skipping of rules as well as simple undo shall be possible.

- The RBES tasks involve planning, monitoring and control, which is realized best using a forward-chaining ES, see Section 1.3.

- The RBES has to be integrated in a given CAD software environment. Therefore, access to the source code of the ES shell is required.
- The RBES is evaluated together with an industrial partner. Thus, legal issues concerning the ownership and licenses of the used source code have to be considered.

The available ES shells and KRLs usually fulfill only parts of the stated requirements. For example, Clips is an open ES shell in a forward-chaining setting [Giar 05, Rile 10]. However, the syntax is Lisp-like and, thus, difficult to read and understand for HA designers. Hence, the decision was made to develop our own KRL and ES shell.

We chose the term script language, because we only require a simple language and a lean ES shell. For the KRL, we use a context free grammar similar to PASCAL. Despite its simplicity this representation is sufficiently powerful to encode the knowledge for the shell shaping process [Sick 09]. In addition to the already mentioned advantages, it allows a simple integration along with the possibility of adding new functionalities as the need arises, while keeping the framework as simple and understandable as possible.

4.1.2 The ES Shell – Script Parser and Interpreter

The ES shell is implemented with help of the tools flex[1] and bison[2] [Donn 03]. The implementation as a script parser is possible, because the target process is linear (order of design steps) and does not require backtracking, see Figure 4.4. Thus, the implementation of the Rete [Forg 82] and similar algorithms is not necessary. A visualization of the parsing and interpretation structure is given in Figure 4.1.

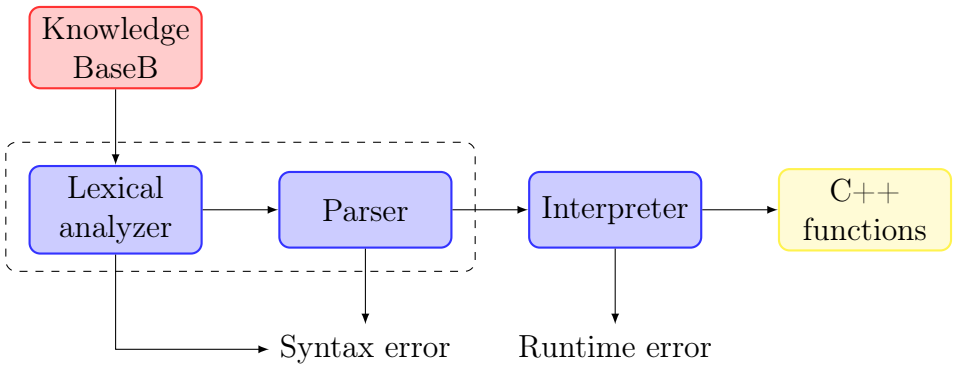


Figure 4.1: Scheme of the parsing and interpretation pipeline.

As shown in the figure, the lexical analyzer and the parser check for syntax errors in the given KB, while the interpreter interprets each command and executes the associated C++ functions. There are a number of choices in designing an interpreter [Aho 85]. In our case, we choose to divide it in two parts. On the one hand we have the parser, which parses the KB and builds an abstract syntax tree (AST). On the other hand, we implemented an AST interpreter. The resulting advantages

[1]The Fast Lexical Analyzer, http://flex.sourceforge.net/, last visited 2011-01-06.
[2]GNU bison, http://www.gnu.org/software/bison/, last visited 2011-01-06.

are the following: (i) parsing and command execution is decoupled, (ii) parsing can be done with existing tools and (iii) interpretation is efficient.

Implementation

The flex tool is used to translate a given script file (the knowledge base in Figure 4.1) into a sequence of tokens for the parser [Levi 92]. The parser is realized with the general purpose parser generator bison [Donn 03]. It converts a grammar description of a lookahead left to right (LALR) context free grammar into a C parser program. A graphic representation of the implementation is given in Figure 4.2.

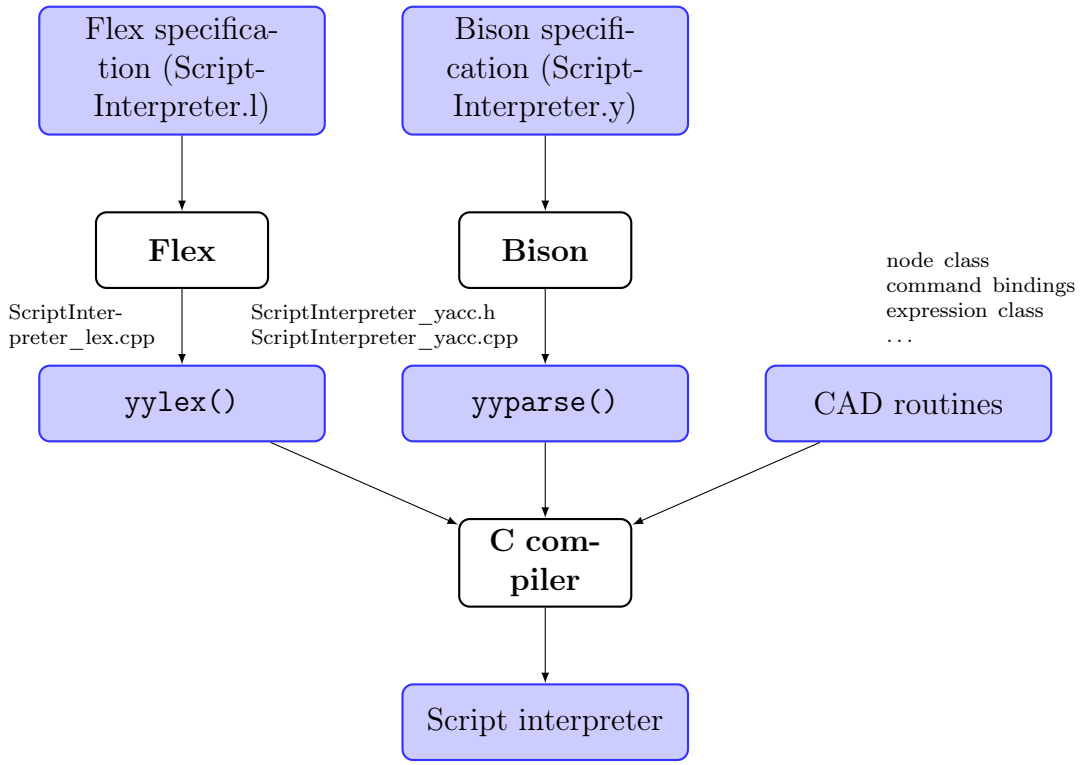


Figure 4.2: Usage of the tools flex and bison to create a KB (script) interpreter. Detailed information about the `ScriptInterpreter.l` and `ScriptInterpreter.y` files can be found in the Appendix A on page 151.

The result of the parsing is an AST. The AST is represented as a directed acyclic graph based on instances of a node class defined in the CAD software. The AST is constructed out of main blocks (frames) and functions using the node class. Thereby, a main block consists of a symbol table and a list of statements. The symbol table holds the variable declarations. Statements may be variable assignments, function calls, branching, loops or CAD-operations. Functions, branches, loops and CAD-operations, in turn, contain blocks of statements. Expressions, such as $1 + 2$, may be assigned to a variable, used in branching, conditional loops and as functions parameters. An expression itself is again a tree, which may contain just one node. In summary, the AST represents an instance of the grammar defined in the yacc file. In Listing 4.1, an excerpt of the script language in Backus-Naur Form is given. It

shows the traditional scheme of a script language, with the specialty of the operations blocks (`opblock`), which is described in more detail in Section 4.1.4 and shown in Figure 4.4.

```
<program> ::= <statementlist_main>
<statementlist_main> ::= <statement_main> | <statementlist_main> <
    statement_main>
<statementlist_if> :: <statement_if> | <statementlist_if> <statement_if>
// similar for while, for, repeat, funcdef, procdef, operation
<statement_main> ::= <statement> | <statement_mainerr> <error> <EOL>
<statement_if> ::= <statement> | <statement_iferr> <error> <EOL>
// similar for while, for, repeat, funcdef, procdef, operation
<statement> ::= <EOL> | <conditional> | <opblock> | <varassignment> | <
    vardefinition> | <RETURN> <EOL> | <RETURN> <expr> <EOL> | <BREAK> <
    EOL> | <EXIT> <EOL>    | <proccall> | <funcdefinition> | <
    statementinvalid> <error> <EOL>
<conditional> ::= <if_elsif_else_endif> | <for_do_endfor> | <
    while_do_endwhile> | <repeat_until>
// ...
<if_elsif_else_endif> ::= <condition_if> <statementlist_if> <elsifs> | <
    condition_if> <elsifs>
<condition_if> ::= <IF> <expr> <THEN> | <IF> <error> <EOL>
// ...
<opblock> ::= <opopen1> <OPEXEC> '(' <symname> ',' <exprlist> ')' <EOL>
    |
 <opopen1> <OPEXEC> '(' <symname> ')' <EOL> |
 <opopen1> <statementlist_operation> <OPEXEC> '(' <symname> ','
 <exprlist> ')' <EOL> |
 <opopen1> <statementlist_operation> <OPEXEC> '(' <symname> ')' <EOL> |
 <opopen2> <symname> ',' <exprlist> ')' <EOL> |
 <opopen2> <symname> ')' <EOL> |
 <opopen1> <error> ')' <EOL> |
 <opopen2> <error> ')' <EOL>
<opopen1> ::= <OPOPEN> '(' <symname> ',' <expr> ',' <opmodifiers> ')' <
    EOL> |
 <OPOPEN> <error> <EOL>
// ...
<expr> ::= '(' <expr> ')' | <TRUE> | <FALSE> | <CHAR> | <INT> | <REAL> |
 <STRING> | <NEWSYMBOL> | <lval> | '-' <expr> | <expr> '+' <expr> |
 <expr> '-' <expr> | <expr> '*' <expr> | <expr> '/' <expr> |
 <expr> <EQ> <expr> | <expr> <NE> <expr> | <expr> '>' <expr> |
 <expr> '<' <expr> | <expr> <GE> <expr> | <expr> <LE> <expr> |
 <NOT> <expr> | <expr> <AND> <expr> | <expr> <OR> <expr> |
 <expr> <XOR> <expr> | <funcall> | <funcall> '[' <expr> ']' |
 <funcall> '[' <expr> ',' <expr> ']' | <FEATURE_SCALAR> |
 <FEATURE_POINT> |  <FEATURE_PLANE> | <FEATURE_CURVE> |
 <FEATURE_AREA> | <FEATURE_AREACOLLECTION> |
 <DEFINED>  '(' <feature> ')' | <PARAMETER>
// ...
```

Listing 4.1: Backus-Naur Form of the developed script language (excerpt).

Further details about the language parser may be found in the Appendix A.2 on page 153.

All functions and constructors used by the interpreter are registered by an existing command registration mechanism available in the CAD software. The command binding mechanism allows to call and interact with the CAD functions. In addition, the extension of the script functionality is simplified. Sections 4.1.3 to 4.1.5 provide examples for many the available functions. For further illustration, an example of the AST interpretation and execution is given below.

Example of Knowledge Base Execution

At first the KB is parsed resulting in an AST. The interpreter maintains the AST, a node stack, a value stack and a frame stack. When a frame (block or function) is entered, a new frame is pushed on top of the frame stack and variables are allocated according to the symbols declared in the frames symbol table. Local variables are accessed at the top most frame, while global variables are accessed on the bottom most frame of the stack. Such an approach supports recursive function calls if needed.

Each node consists of a node pointer and an integer state value. The top most node of the node stack is executed according to its state. During execution the node may push its children on the stack, push or consume values from the value stack and change its state. The node is removed from the stack, when the state value indicates the closed state. The interpreter keeps executing the top most node until the stack is empty or a runtime error is encountered. The node, value and frame stack may be serialized, which allows the undo or redo of frames if necessary. The following example shows the execution of the expression $1 + 2$:

1. Push expression + onto the node stack and set its state value to 0.
2. Execute + with state 0: set node state to 1, push children *const 1* and *const 2* onto the node stack with state 0.
3. Execute *const 2* with state 0: store *const value 2* to value stack, close node.
4. Pop *const 2* from node stack.
5. Execute *const 1* with state 0: store *const value 1* to value stack, close node.
6. Pop *const 1* from node stack.
7. Execute + with state 1: pop values *1* and *2* from value stack, push *const value 3* to value stack, close node.
8. Pop + from node stack.

4.1.3 CAD-Integration and Functionality

The developed language supports standard data types, like booleans, integers, floats, strings and arrays of all types. In addition, 3-D points, planes and matrices are added as special data types (Table 4.1). For each data type, the standard calculation

and comparison operators are made available, which allow vector and matrix based computations (Table 4.2). Furthermore, the script language supports several control structures, such as `if-then-else` blocks, `for`, `while` and `repeat-until` loops. Finally, the definition of functions and procedures is possible.

Type	Example	Description
Boolean	`bool b`	True or false
Integer	`int i`	Natural numbers (-2147483648 to 2147483647)
Floats	`real r`	Floating point numbers ($\pm 3.4 \cdot 10^{\pm 38}$)
String	`string s`	Sequence of symbols
3-D points	`point x`	3-D floating point variable
3-D planes	`plane p`	4-D floating point variable storing a plane in Hessian normal form
Matrix	`matrix m`	4×4 floating point matrix

Table 4.1: Data types supported by the script language.

Operator	Example	Description
+	`int a = 7 + 1`	Sum of integer, real or point values or concatenation of strings
−	`int b = 7 - 1`	Subtraction of integer, real or point values
*	`real c = 7.0 * 1`	Multiplication of integer and real values. Linear algebra operations: point * scalar, matrix * point, matrix * matrix
/	`point d = p / 3`	Division of integer or real values by another scalar value or division of point values by scalar values
>, >=, <, <=	`IF x > y THEN`	Comparison of integer or real values
<>, ==	`IF x <> y THEN`	Equal, not equal operator, which are available for all basic data types

Table 4.2: Operators supported by the script language. All operators support implicit casting.

Functions

Next to the data types and operators, a set of basic functions to encode the rules are provided. These functions are called *internal functions*, since they are provided by the command binding mechanism of the CAD software. They provide an interfaces to the CAD system and support mathematical calculations. In contrast to these functions, all functions and procedures defined in the KB are referred to as *rules* or *logical functions*. They are used to store the available process and design knowledge. In Figure 4.3, a visualization of the language functionality is given.

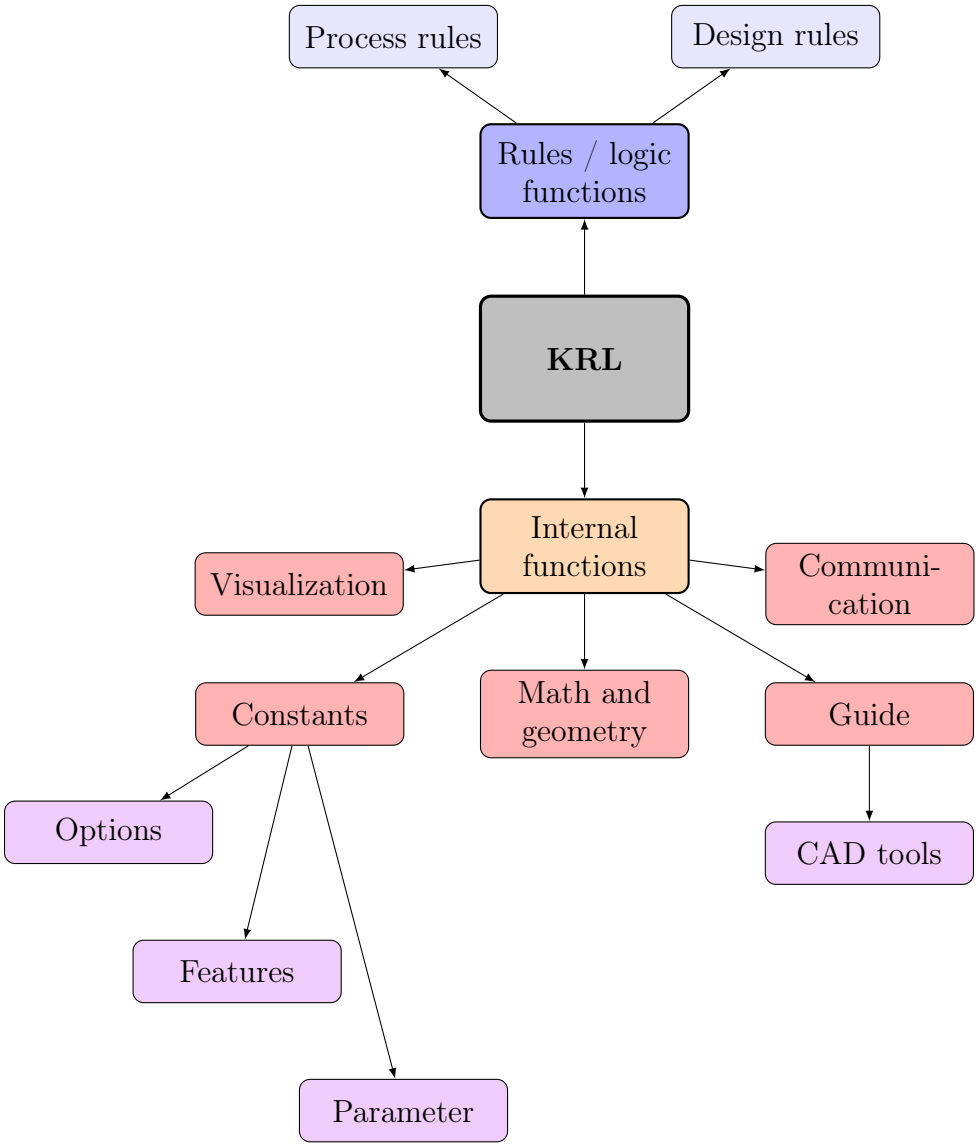


Figure 4.3: Overview of the functionality of the developed knowledge representation language.

4.1.4 Internal Functions

The internal functions are divided into five groups: *visualization*, *constants*, *math and geometry*, *guide* and *communication*. They are indicated by the red boxes in Figure 4.3. Some of these are further subdivided, e. g. *constants*. Below, the purpose and syntax of each group is introduced by examples.

Constants

Features To define individualized rules for the design of a certain shape, it is crucial to access its anatomical features. Therefore, the features listed in Table 3.1 are available as variables in the language. If a rule utilizes a certain feature, the feature detection is invoked and the result is returned to the rule.

To distinguish a feature point detected on the unprocessed surface and one on the current surface the suffix `Undetailed` is used. For example `CanalTipPointUndetailed` denotes the original tip point of the ear canal, while `CanalTipPoint` denotes the tip point after the canal was modified, see Figure 3.12.

Example 1. features

```
plane plane1 = CrusValleyPlaneUndetailed
point point1 = TragusPoint
```

Options The `MfgOption` function provides access to the order details. If a customized hearing aid needs to be designed, the target hearing aid is defined by so called manufacturing options. These options define the device type, vent type, receiver type, tapering and rounding levels, etc. In other words, their input defines the necessary steps to design a certain HA. Each option can be accessed with the `MfgOption` function and a number associated with that option, see Example 2.

Example 2. manufacturing options

```
string device_type = MfgOption(101)
string feedback_seal = MfgOption(111)

IF feedback_seal == "FS" THEN
  ...
ENDIF
```

Parameters Next to the manufacturing options, global process parameters exist, which can be accessed using the `GetParameter` function. Parameters are for example an offset applied on the surface or the wall thickness, the vent wall thickness or the minimum distance between the receiver hole and the vent hole on the canal tip. The global parameters are stored in an xml file and the `GetParameter` function provides read access to the file as shown in the example below. Since, the function (similar to `MfgOption`) returns a string value, it can be necessary to convert it into a number, which can be achieved with the `Number` function.

Example 3. parameters

```
real shell_wall_thickness =
    Number(GetParameter("Parameter.Shell_Wall_Thickness"))
real min_distance =
    Number(GetParameter("Parameter.Min_Distance_Vent_Receiver"))
```

Guide Functions – Interface to CAD Software

The ES is integrated in the CAD system in form of a guide. The guide leads the designer through the process of modeling a customized HA. Thereby, it executes each necessary process step and sets up the CAD environment accordingly. This includes the selection of appropriate tools and parameters as well as the visualization of the rule solution.

Furthermore, two automation modes are available. In the semi-automatic mode, the ESAM sets up the CAD environment and the tool, but waits on the user verification before applying the operation. For example, if a process step requires a cutting plane, then a cutting plane is defined and the appropriate cutting tool is selected. The designer can now verify or modify the plane before application of the tool. In the automatic mode, the verification step is skipped and every operation is applied immediately.

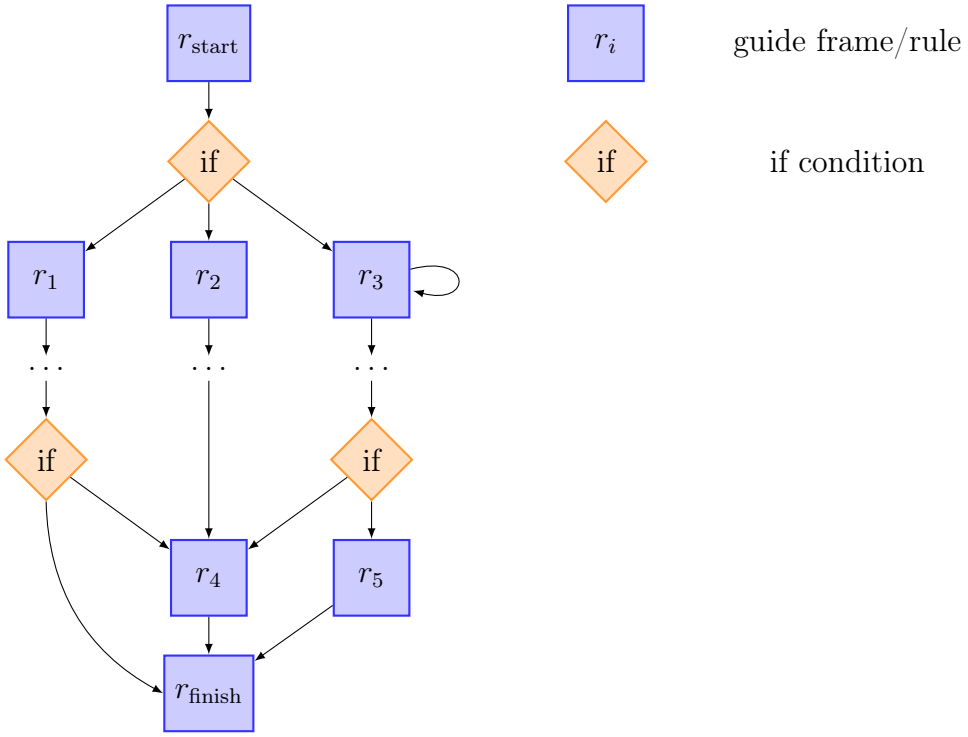


Figure 4.4: Scheme of a knowledge base or script. The guide rules of the KB are the high level building blocks defining the design process. Depending on the given options different paths from the starting to the finishing rule have to be taken. The path of execution is controlled by if-then-else conditions.

The two available guide functions: `OpOpen` and `OpFinalize` are always used as a pair. They form a frame, setting up the CAD environment for the process rules enclosed by them. The `OpOpen` function initializes the CAD system by setting the modeling mode, the process step name and the synchronization mode. The modeling mode is either *Detailing* or *Modeling*. Detailing enfolds all types of cutting and rounding operations. Modeling indicates either movement/placement of electronic components or substantial surface modifications, like integration of an inner wall

or the ventilation tube. The process step name is a string which is shown during execution that informs the user about the current process step. The last parameter is special for the HA design process. Often a patient requires a HA device for left and right ear. Therefore, the CAD system supports binaural modeling. Binaural modeling copies the operations executed on the left side to the right side or vice versa. `Execute` denotes that the rule is applied for each side independently. In contrast to that, `Mirror` denotes the mirroring of a rule application from one ear side to the other. Finally, it is possible to add the keyword `AutoApply`, which causes the execution of the process step without user verification, even in the semi-automatic mode.

Example 4. guide rule example

```
OpOpen(Detailing, "Helix Cut", Mirror+AutoApply)
  ...
OpFinalize(Round, rounding_plane, reference_point)
```

The `OpFinalize` function executes the given CAD tool with the computed parameters. The parameters depend on the tool, e. g. the rounding tool requires a rounding plane and a reference point.

A complete knowledge base or script contains a large number of guide blocks, which need to be processed in the right order. The execution of a guide block is controlled by conditions, typically in if-then-else form. A graphic representation of a KB is given in Figure 4.4. It can be seen in the figure that forward-chaining is necessary and no backtracking is required.

Math and Geometry Functions

Math Functions The math functions offer various ways to manipulate numbers, vectors and matrices. This includes dot and cross product, trigonometric functions, rotation matrices, etc. Furthermore, helper functions for distance calculations, such as point-to-point distance, point-to-plane distance or point-to-ridge distance are provided, see Example 5.

Example 5. math functions

```
point x = point(1,2,3)
matrix rot = Rotation(x, 20.0 * PI / 180)
real f = x.dot(point(4,5,6))
point pcross = x.cross(point(4,5,6))
plane p = Plane(TragusPoint, AntiTragusPoint, AntiHelixPoint)
f = Dist(p, x)
real cos20 = cos(20.0 * PI / 180)
```

Geometry Functions The geometry functions offer access to the surface geometry. Examples include the computation of surface contours, region growing or shrinking, etc. In addition, several functions to modify the orientation and location of planes in respect to other planes or points are available (Example 6).

Example 6. geometry functions

```
point contour[] = GetContour(FirstBendPlane)
point contour_center = GetContourCenter(FirstBendPlane)
point region[] = RegionGrow(TragusPoint, 2.5)
plane p = Plane(TragusPoint, AntiTragusPoint, AntiHelixPoint)
plane p2 = p.Move(CanalTipPoint, 2.5)
point proj = p2.ProjectOnPlane(TragusPoint)
point proj2 = p2.ProjectOnContour(TragusPoint)
```

Visualization Functions

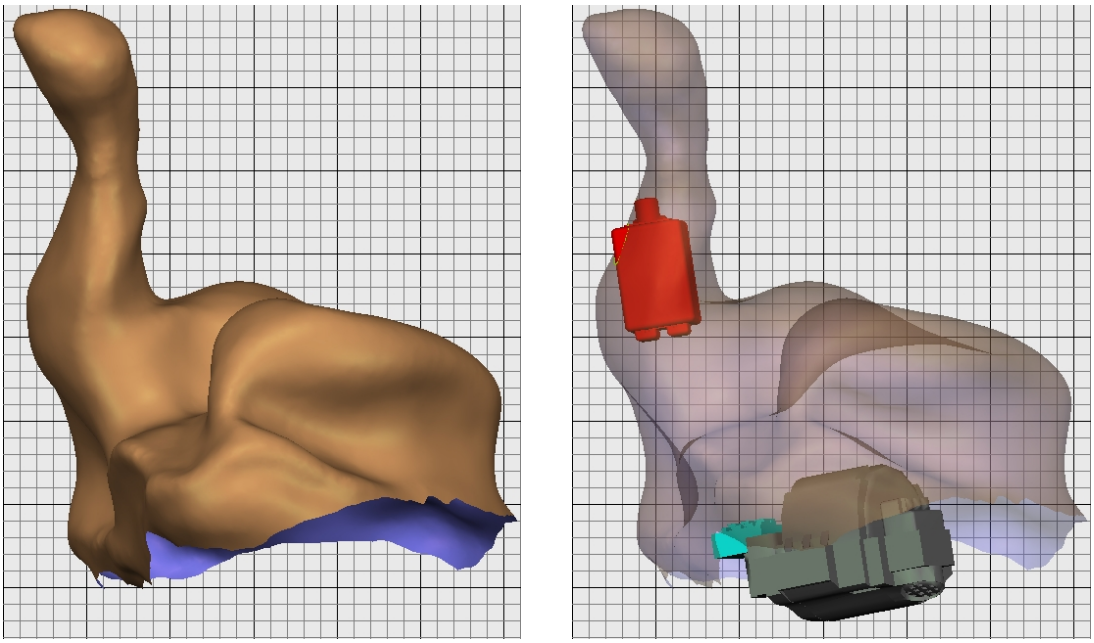
Visualization functions are used to support the guide functions. They also set up some of the CAD environment settings, such as showing the surface transparent or toggling the visibility of certain components (Figure 4.5). These functions mainly improve the efficiency of the designer during modeling in the semi-automatic mode. They provide the correct setting for the visibility of the components, the tools and the surface.

Example 7. visualization functions

```
ShowShellTransparency(true)
ShowComponents(false)
ShowCollisionDetection(false)
ShowColorMapping(true)
```

Communication Functions

The communication functions provide the explanation component to the user (see Section 1.3). The ES can communicate, whether information is missing or problems are encountered or that a certain process is executed right now. A text box is integrated in the CAD environment to show this information. Available functions are `Info`, `Warning` and `Abort`. The former two show just a string, while the latter shows a string and stops ESAM due to the encountered problem.



(a) Hearing aid shell shown solid without electronic components.

(b) Hearing aid shell shown transparent with electronic components visible.

Figure 4.5: Visualization functions provide functionality to switch the visualization of the CAD environment according to the needs of the process step, for example, displaying the surface either transparent or solid.

Example 8. communication functions

```
Info("Hello World!")
Warning("Ear canal is too short!")
Warning("Receiver hole too close to vent, dist = " + dist)
Abort("Crucial Feature is missing!")
```

4.1.5 Rules

Rules or logical functions are defined in the KB itself using the KRL and the previously defined internal functions. We distinguish two types of rules. On the one hand, there are the process rules, controlling the execution of guide blocks. They typically consist of an if-then-else conditions or a loop with one or more guide blocks in their body. On the other hand, we have the design rules. A design rule is the body of a guide frame, computing and defining the needed parameters to execute a certain tool. A simplified example is given below.

Example 9. rules

```
FUNCTION is_cic_type() : bool
  IF (MfgOption(101) == "CC") OR (MfgOption(201) == "N3") THEN
    RETURN true
  ENDIF
RETURN false
ENDFUNC

IF is_cic_type() THEN      // start of process rule
  Info("CIC device -> helix will be removed")
  OpOpen(Detailing, "Helix removal for CIC", Mirror+AutoApply)
   // start of design rule
    plane rounding_plane = CrusValleyPlane.Move(HelixPeakPoint, 2.0)
    int level = 2
    point reference_point = TragusPoint
    // end of design rule
  OpFinalize(Round, rounding_plane, reference_point, level)
  Info("Helix removed")
ENDIF                      // end of process rule
```

In the above example, at first, a function `is_cic_type()` is defined. This function returns true, if at least one of two options accessed with the `MfgOption` function matches the expected values. The function is immediately used by the following process rule (the if condition). The enclosed guide frame is therefore only triggered in case of a CIC device. The shown design rule computes a plane and a reference point. Furthermore, a level is defined. All three parameters are used as input for the `Round` tool specified in the `OpFinalize` function. The rounding operation is applied immediately (`AutoApply`) and mirrored to the other side if applicable.

4.2 Knowledge Acquisition

We adopted the general iterative approach for building the KB as described in [Giar 05]. This approach is based on the creation of an initial or prototype KB, which is then refined based on the critics of experts. The process is repeated until a complete KB is constructed. For the knowledge acquisition process, three input sources were available: work instructions provided by the manufacturer for its designers, interviews with expert HA designers and supervised hands-on training of the knowledge engineer. A graphic overview is given in Figure 4.6.

First, the written work instructions were directly translated into rules and the resulting prototype KB was evaluated with several sample cases. Naturally, the first KB lacked a lot of knowledge. With help of the expert operators, the gaps in the written instructions were filled and tested.

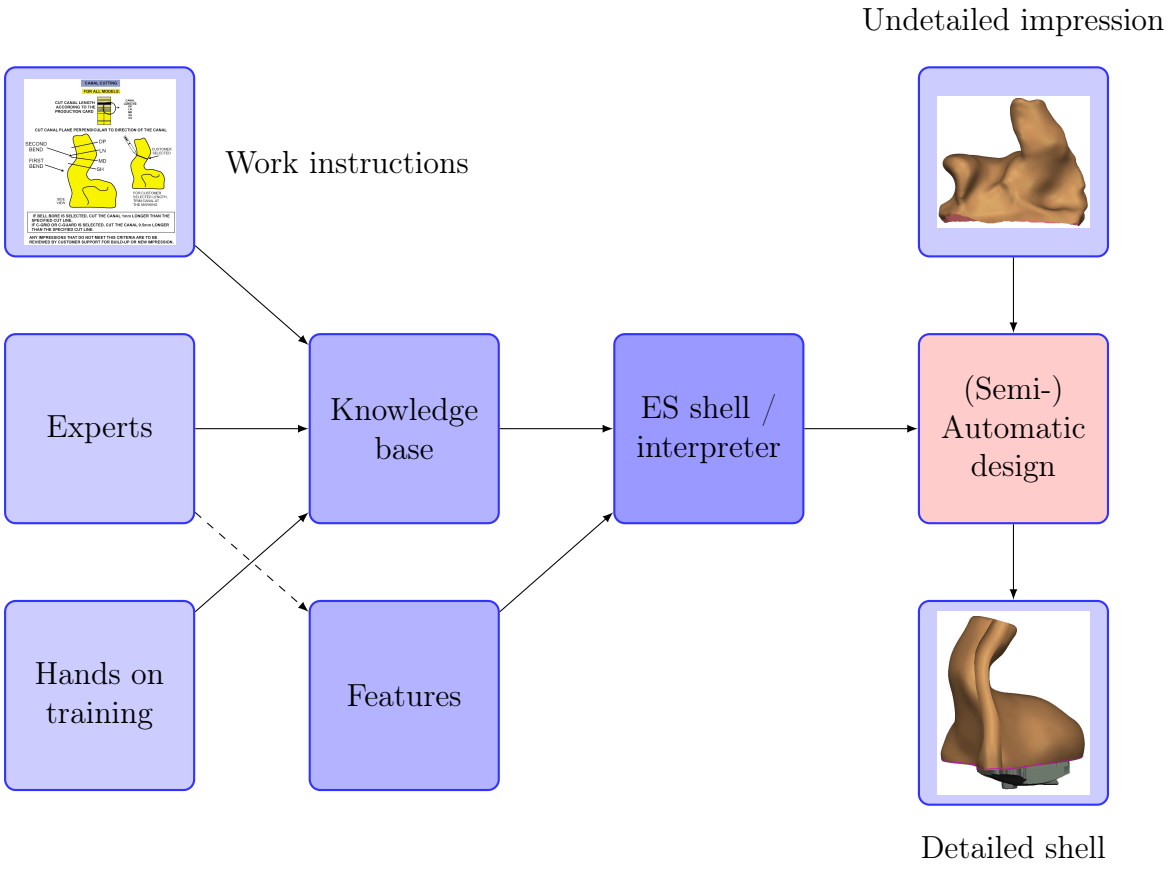


Figure 4.6: The KB contains the knowledge of the work instructions, experts and own experience. In combination with the detected features the ES shell executes the rules and controls the modeling software.

One immediate effect of our work was that it helped in identifying the ambiguities in the written work instructions. This allowed improvements in the current instructions, which are now much more detailed and precise than before. It took several iterations until the first complete KB was acquired. Nevertheless, the performance of the eventual KB was not as good as an expert. The main reason is the huge amount of variability in the ear canals, which may result in two problems.

First, a rule can fail due to potential inaccuracy of a detected feature. Second, a work instruction may turn out to be too general to cope with such variation. To improve this, we again consulted our experts and fine tuned the KB further, which is discussed in detail in Chapter 5.

4.3 Preliminary Experiments and Results

4.3.1 Experiments

For testing the usability and performance of the initial ESAM versions, we used a set of 39 different ear impressions in combination with 13 different device type option combinations. Six experts processed all 39 samples. The operators were advised

to make corrections if necessary. To analyze the performance of our framework we developed a quality scale. This scale enables the expert operators to assign a value for each guide rule for each processed sample. The available values are:

- 0 if the ESAM proposal is unusable.
- 1 if the proposal is usable, but needs modifications.
- 2 if the proposal is acceptable without modifications and
- 3 if the proposal is perfect.

The value 3 was introduced to consider the tendency of the operators to make minor adjustments sometimes ending up at the exact same spot as the ESAM proposal.

4.3.2 Results

The overall quality level was promising. On average ESAM proposed an acceptable solution for each step. For full automation it would be necessary to reach at least a mean value above 2 with a small standard deviation for each rule. In this evaluation, *automation* was reached for about 14 % of the test cases (every script frame was rated ≥ 2). The median value is ≥ 2 for 75 % of the rules, which as well shows that an acceptable performance is reached in most cases. The results are given in Table 4.3.

Rule name	μ	$\pm\sigma$	$\tilde{x}$
Rough Cut to Remove Excess Material	2.21	0.86	2
Taper Canal Tip	1.74	0.98	2
Remove or Reduce Cymba	2.45	0.78	3
Round Cymba Peak	2.43	0.77	3
Local Scooping at Crus Area	1.07	1.14	1
Optional Cuts	1.68	0.87	1
Local Offset at Ear Canal	1.72	1.13	2
Waxguard Cut	2.57	0.86	3
Optional Vent Cuts	2.60	0.76	3
Place Vent Points	2.04	0.93	2
Place Receiver Hole	1.97	1.06	2
CIC Measured Shell Height Cut	1.14	0.81	1
ITC Measured Shell Height Cut	2.10	1.01	2
ITE Measured Shell Height Cut	1.58	0.93	1
Local Fill at Cymba Area	2.25	1.10	3
Place Label Inside of Shell	1.86	1.26	2
Average	1.96	0.95	2

Table 4.3: The table shows the quality values given by the expert operators using ESAM to design a hearing aid device. For the sake of clarity and space, similar frames are grouped together. μ denotes the mean quality value of all operators for a certain process rule, σ the standard deviation and $\tilde{x}$ the median.

These preliminary results need to be interpreted with care. In our feedback sessions with the experts we recognized, that there was a misunderstanding for some rules between what the framework does and what the operator expected. This, in turn, yields to a lower quality level, e.g. optional cuts. Most of the guide rules perform well on average, but need to be improved in terms of the standard deviation, e.g. *Local Fill at Cymba Area*. A minor subset of the rules (*Local Scooping at Crus Area*, CIC *Measured Shell Height Cut* and ITE *Measured Shell Height Cut*) suffers from the fact that the training set did not cover the shell variability found in the validation set.

The quality of the resulting device was comparable to a device done completely manually. There were small differences in size and shape. However, according to the experts a variation in size of about 2 mm can be expected between operators. Even in the case that an operator processes a device twice. The resulting devices manufactured using our framework were always in this range and visually more consistent between different operators.

The overall feedback by the experts was very positive. ESAM is a great help for following the optimal process. This is very well appreciated by the process trainers as well as new operators, which are not familiar with the process yet.

4.4 Conclusions

The developed KRL and the associated ES shell are capable of designing a customized hearing aid. The support of a semi-automatic mode allows the usage of the automation framework in a real production environment by maintaining the necessary quality of the result (due to correction done by the user if necessary). A preliminary evaluation showed an increase in efficiency, consistency, reproducibility and quality of the design, while ensuring the processing of each manufacturing option.

Chapter 5

Rule Learning

In this chapter two rule learning techniques are discussed. Before describing these methods, the used training data is introduced. Afterward, the learning powers of evolutionary algorithms, genetic programming in particular, are explored. The thereby achieved results were presented at the WCCI 2010 [Sick 10b]. The last part contains information about a semi-automatic method. The chapter concludes with a comparison of the results.

5.1 Data and Learning Objectives

5.1.1 The Data

Two data sets for learning are available. The first data set $\mathcal{S}_1$ is composed of 105 ear impressions, each associated with an expert labeled plane defining the so-called *rough cut* plane (Figure 5.1). The second data set $\mathcal{S}_2$ is composed of xml files containing the records of 26449 hearing aid design tasks gathered over a time frame of 22 months.

Data Set $\mathcal{S}_1$

Data set $\mathcal{S}_1$ was acquired with help of an expert operator. The expert labeled on 105 different impressions the *rough cut plane*. Two example cuts are shown in Figure 5.1. Being the first process step, the influence of this single step on the overall outcome is high. In the KB rules, the defined rough cut plane is often used as a reference feature.

Data Set $\mathcal{S}_2$

During the modeling of a HA using ESAM a record of each rule application is stored in an xml file. An excerpt of such a record is given in Listing 5.1. For each executed rule the following data is recorded: rule name, tool name, parameters set by the rule (e. g. plane and reference point for a rounding operation), and the applied parameters. In the example file below, the parameters set by ESAM are stored in the *setup* node and the finally applied parameters in the *commit* node. The rule name is *Rough Cut to Remove Excess Material* and the applied tool is *OpenFillCut*.

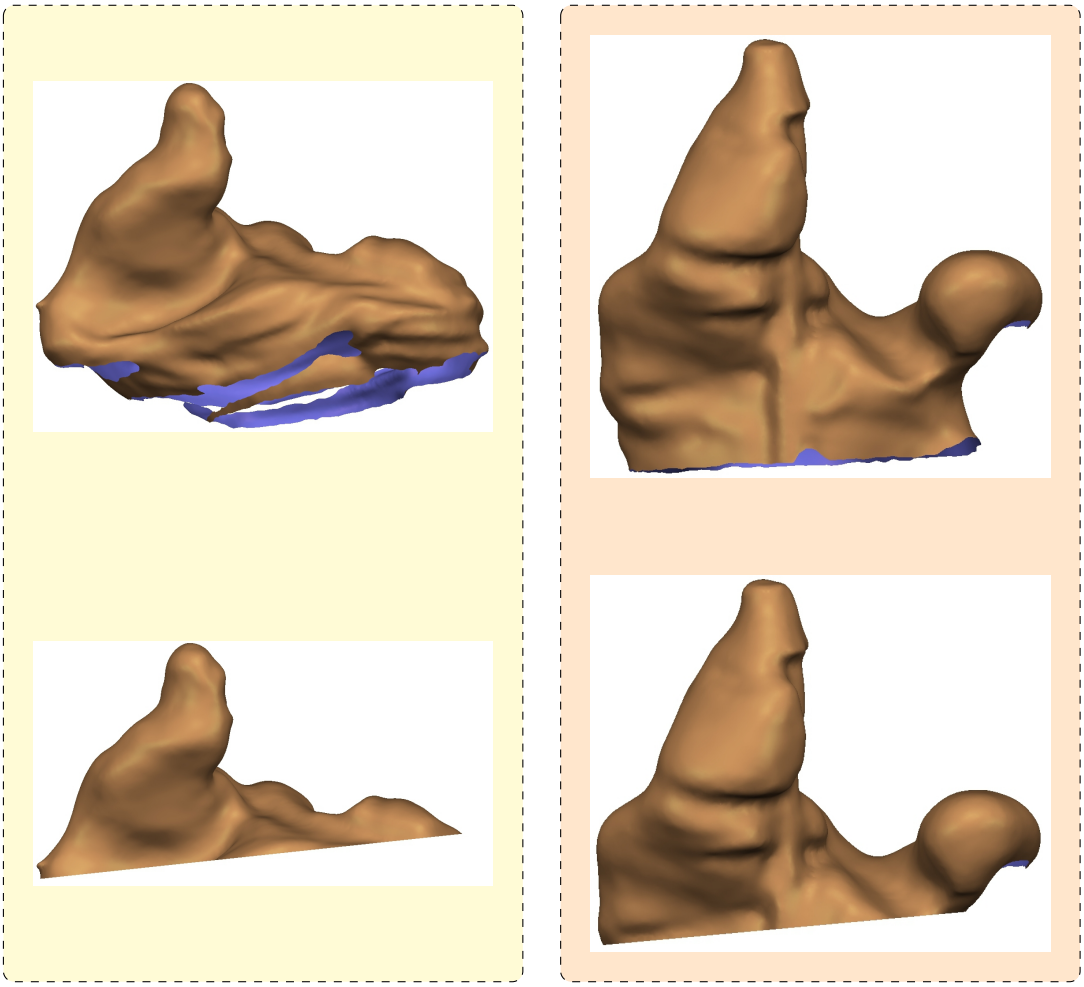


Figure 5.1: Two ear impressions of data set $\mathcal{S}_1$. The upper row shows the original impression and the lower row after application of the expert labeled *rough cut* plane.

```
<ScriptRecord>
  <OpenFillCut Step="Detailing" WizardStep="Rough cut to
      remove excess material">
    <Setup>
      <plane type="PLANEF">
        <x>0.025672</x>
        <y>0.032444</y>
        <z>0.999144</z>
        <c>-14.230454</c>
      </plane>
      <centerOfInterest type="VEC3F">
        <x>-1.522997</x>
        <y>-10.513687</y>
        <z>16.475554</z>
      </centerOfInterest>
```

```
    </Setup>
    <Commit>
      <plane type="PLANEF">
        <x>0.025354</x>
        <y>0.012568</y>
        <z>0.999600</z>
        <c>-13.657396</c>
      </plane>
      <centerOfInterest type="VEC3F">
        <x>-0.502392</x>
        <y>-0.605391</y>
        <z>13.683222</z>
      </centerOfInterest>
    </Commit>
  </OpenFillCut>
</ScriptRecord>
```

Listing 5.1: Example of a design record of data set $\mathcal{S}_2$. The content of the *Rough Cut to Remove Excess Material* rule execution is shown.

The data set $\mathcal{S}_2$ was acquired at different locations and times (Table 5.1). Location refers to three different production facilities which used ESAM. Time refers to three different ESAM versions. ESAM1 was released in April 2009, it was replaced by ESAM2 in October 2009, which in turn was replaced by the current version ESAM3 in April 2010. Hence, the data set allows us to compare the acceptance of ESAM in different facilities and also the performance development over time (versions).

	Location 1	Location 2	Location 3	All locations
ESAM1	329	678	3658	4665
ESAM2	1101	2722	1704	5527
ESAM3	2693	7853	5711	16257
All versions	4123	11253	11073	26449

Table 5.1: Overview of the number of design records for each location and each version contained in data set $\mathcal{S}_2$.

5.1.2 The Learning Objectives

In our work, we focus on two optimization problems. The first one targets the optimization of the *Rough Cut to Remove Excess Material* rule. The rule has a major impact on the final design of a HA, because it is always applied first. Thus it can be analyzed independently of all other rules in the KB. It was picked as a separate optimization problem, because of its importance and designer feedback. The latter criticized it to be not stable enough, especially in case of unusual or bad ear impres-

sions (Figures 3.13 and 3.14). To address this problem, we evaluate the rule learning capabilities of genetic programming using data set $\mathcal{S}_1$ in Section 5.2.

The second learning task is more general and targets all rules in the KB of ESAM. For this we utilize data set $\mathcal{S}_2$ to identify low performing rules and heuristically define suggestions how to improve these a rules. The final rule modification is carried out by the knowledge engineer based on the suggestions. The semi-automatic approach is discussed in Section 5.3.

5.2 Genetic Programming for Rule Learning

5.2.1 Introduction and Motivation for Using Genetic Programming

Evolutionary computation is a field of artificial intelligence. Four fields of evolutionary computation emerged in the past decades [De J 06]. *Evolutionary programming* was introduced by Lawrence J. Fogel [Foge 66]. John H. Holland introduced and popularized the *genetic algorithms* [Holl 92] (first published 1975). In Germany Ingo Rechenberg and Hans-Paul Schwefel developed the *evolutionary strategies* [Rech 73]. Later, in the beginning of the nineties, *genetic programming* (GP) was popularized by John R. Koza [Koza 92] as a method for using natural selection for the creation of new computer programs.

GP has the advantage that it yields programs or in our case rules, which can be analyzed and interpreted. This is one of the reasons why we consider GP for our optimization problem. Since we are dealing with a rule based system, we assume that we know how certain rules should work and be applied. Nevertheless, we might be wrong. One of the excellent properties of GP is that it can be applied, when the interrelationships among the relevant variables are unknown [Poli 08]. In our case, we assume that some of the detected features together define a certain design step. In other words, we assume a certain relationship of these features. GP provides us with an objective method to verify these assumptions.

GP is an iterative process to evolve or grow a population, whereas each individual in the population represents a solution to an optimization problem. The population is developed in a guided random search using parallel processing to identify a local, possibly global, optimum [Foge 05]. Attention should be paid to the fact that GP is based on random processes and hence, a good result cannot be guaranteed. Then again, the randomness enables GP to escape traps, which deterministic algorithms can be captured by. This quality allowed GP (and other evolutionary algorithms) to evolve novel and unexpended ways of solving problems [Poli 08].

Naming conventions: In GP the *population* is comprised of *programs*. The programs are often referred to as *individuals* or *genomes*. The latter emphasizes the analogy to the biological process of DNA recombination.

Recent Work

GP has already been applied as an optimization technique on a wide variety of discrete problems and applications [Poli 08, Will 97]. Examples vary from the analy-

sis of high energy physics data [Link 05] over recognition of handwritten digits on mail [Park 04], improvements in robot control [Nord 98] to solving of the ocean coloring problem [Fonl 01]. Furthermore GP itself was topic of high research interest. Especially the influence and performance of selection, crossover and mutation operators are discussed and compared [Poli 97, Hutt 01, Park 04, Kouc 07, Poli 08, Badr 10].

Rule definition with help of GP has already been successfully applied, for instance by [Andr 94], [Mend 01] and [Potv 04]. These approaches focus on developing rules for classification purposes, like optical-character-recognition (OCR) or data mining. Applying GP in the field of medical prostheses was already applied by Collet et al. [Legr 07]. They addressed the problem of fitting the various parameters of cochlear implants (Section 2.3.2) for a patient. They evaluated an interactive evolutionary algorithm approach with a micro-population and compared the results with an expert. Tsakonas et al. [Tsak 04] proposed an evolving rule-based system for medical decision support. They focused on generating short and simple rules for diagnosis of aphasia's subtypes and the classification of pap-smear examinations. To the best of our knowledge, in the area of design rule learning no works have been published so far.

Basic Algorithm

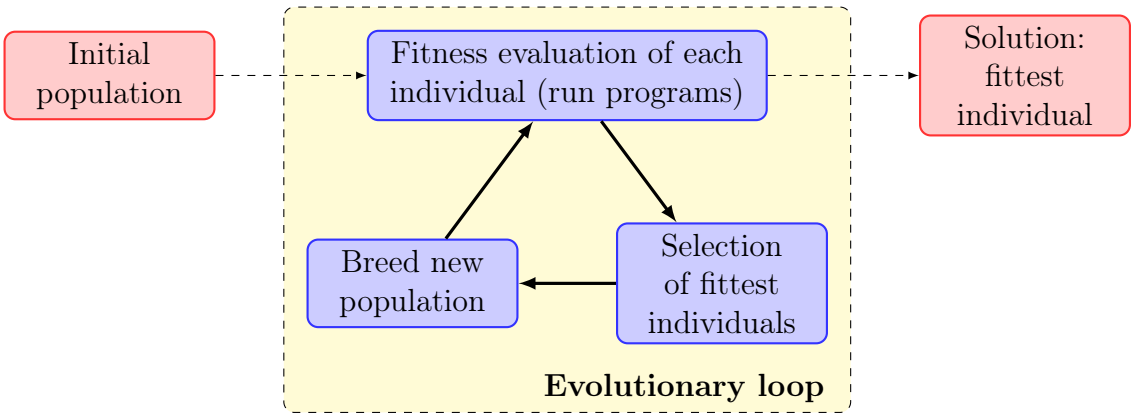


Figure 5.2: GP starts with an initial (random) population, iterates a number of times through the evolutionary loop until presenting a solution. The evolutionary loop contains the main GP steps: selection, crossover and mutation. The latter two are grouped together in *breed new population* in the figure.

GP evaluates each individual of the current population in respect of a fitness (objective) function. Such a fitness function usually transforms the program execution result of an individual into a numeric value which can be compared easily. An essential part of the GP is how to create a new generation based on the computed fitness information.

The basic algorithm is summarized in Algorithm 6 and visualized in Figure 5.2. The three main steps (selection, crossover and mutation) are discussed subsequently in Sections 5.2.2 to 5.2.5.

Algorithm 6: Genetic programming [Poli 08]

1 Randomly create an initial population $\mathcal{P} = \{I_1, I_2, \ldots, I_N\}$ of N programs from the available primitive set;
2 **repeat**
3 | *Run* each program and calculate its fitness $\delta_F(I_i)$;
4 | *Select* a portion of individuals as parents for the next generation;
5 | *Create* new individual programs by applying genetic operators on the parent individuals;
6 **until** *acceptable solution is found or maximum iterations are performed*;
7 **return** *Fittest individual*;

5.2.2 The Primitive Set

Each individual of the population is represented by an abstract syntax tree defined by the grammar introduced in the previous chapter (Figure 5.3). The nodes of the AST are either terminals or functions. The terminal set is represented by variables of the data types specified in Table 4.1 and the features listed in Table 3.1. The function set is comprised of the operators listed int Table 4.2 and a subset of the *internal* functions defined in Section 4.1.4. The functions are mostly among the geometry and math functions and constant parameters defining the HA design task. Furthermore, the function set is enhanced by loop (e. g. while) and branching structures (e. g. if-then-else). All available elements of the terminal and function set are commonly summarized as the *primitive set*. Note: The GP framework does not create functions or procedures.

Closure

Closure is an important property of a GP system [Koza 92, Poli 08]. It requires the function set to be of *type consistency* and *evaluation safety*.

GP operators can combine and join nodes arbitrarily. In order to remain a syntactically correct tree it is necessary to either require all functions to be type consistent, meaning that all will return values of the same type, or that the GP operators take the input and return type of the functions into consideration. The latter is referred to as strongly typed GP [Mont 95]. We use strongly typed GP, because otherwise modifications on the given grammar would have been necessary. A third solution is to ignore type consistency and associate a penalty value to syntactically wrong individuals, but this approach limits the parallel evolving power of GP [Poli 08].

Evaluation safety is required to avoid failures at runtime. For example division by 0 is typically captured by the usage of a protected division operator, which returns 1 in such cases. Similar modifications are applied for other numeric functions, like logarithm, exponential and square root. Furthermore, the execution of loops can be limited by a maximum number of iterations to prevent infinite loops. If such an exception is caught or an infinite loop is stopped, the fitness value of such an individual is penalized.

Sufficiency

Sufficiency means that it is possible to express a solution of the problem using the given primitive set. Therefore, sufficiency can only be guaranteed for problems where either theory or experience shows that a solution can be obtained by combining the given primitives [Poli 08]. In our case, experience tells us that it is possible to define a good solution, but it cannot be concluded that it is possible in all cases. Nevertheless, we assume that the provided primitive set is sufficient for the problem at hand.

5.2.3 Initial Population

The initial population is typically created by a random process. Different strategies targeting the shape of the syntax trees are proposed in the literature [Poli 08]. Examples are the *full* and *grow* methods, and a widely used combination of the two *ramped half-and-half*.

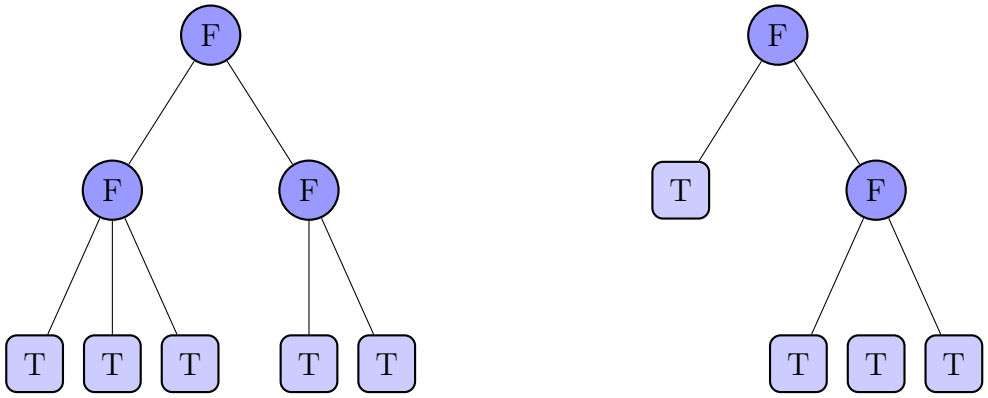


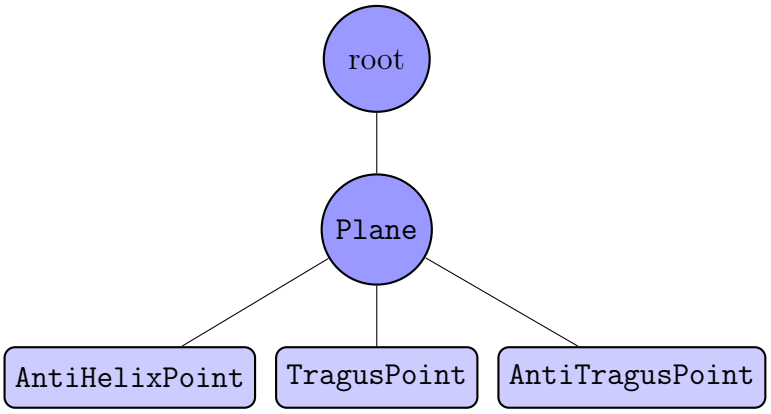
Figure 5.3: Two ASTs with depth 2. The left one is a *full* tree having terminals (T) only at depth 2, while the *grown* tree on the right side contains terminals and functions (F) at all levels except level 0, where only functions are allowed.

In the full method, a syntax tree is grown randomly using only functions until a user specified depth is reached (Figure 5.3). Only after the user specified depth is reached, terminals are used as tree nodes. The grow method is similar, but allows to use terminals from the start, therefore, allowing trees with various shapes. The Ramped half-and-half method uses full for one half of the individuals and grow for the other half to create the initial population. To ensure a greater variety often a range of depth limits is applied (ramped) resulting in a population of AST having a variety of sizes and shapes.

Furthermore, the initial population needs not to be entirely random. It is possible to inject knowledge by using small trees with desired properties as input structures for the tree growing process [Poli 08].

5.2.4 Fitness Evaluation

The goal of our GP optimization is to identify a program (rule) which defines an optimal cutting plane for a wide variety of ear impressions. Thus, we compute the




```
Plane(TragusPoint, AntiTragusPoint, AntiHelixPoint)
```

Figure 5.4: Example of an individual defining the rough cut plane in form of an AST. The functions are denoted by circles while the terminals are indicated by rounded rectangles. Below the syntax tree the script representation is given.

fitness of each individual in the population using the data set $\mathcal{S}_1$. Therefore, we breed programs, which compute a plane using the given primitive set. A minimal example individual is given in Figure 5.4.

In order to compute the fitness value for a certain individual the orientation and location of the defined plane is compared to the one labeled by experts (similar to the feature plane evaluation in Section 3.4.3). Consequently, for each sample in $\mathcal{S}_1$ and each individual, a fitness value is computed based on the equations given below. The resulting fitness values are summarized and averaged for each individual. To speed up the fitness computation it is possible to use a random subset of samples.

Fitness Measure

The orientation measure δ_{or} compares the two normals $\boldsymbol{n}_a$ and $\boldsymbol{n}_b$ of plane a and b.

$$\delta_{\text{or}}(a,b) = 1 - |< \boldsymbol{n}_a, \boldsymbol{n}_b >| \tag{5.1}$$

In Eq. (5.1), $<,>$ denotes the inner product. The localization measure δ_{loc} compares the distance to the origin d_i of both planes.

$$\delta_{\text{loc}}(a,b) = |d_a - d_b| \tag{5.2}$$

The fitness function δ_{F} combines both measures with a weighting factor α ($0 \leq \alpha \leq 1$).

$$\delta_{\text{F}}(a,b) = \alpha \cdot \delta_{\text{or}}(a,b) + (1-\alpha) \cdot \delta_{\text{loc}}(a,b) \tag{5.3}$$

Fitness Ranking

The individual with the smallest fitness value δ_{F} is identified as best individual. If individuals share the same fitness, a *size constrained ranking* is used which incorporates the tree size of the individuals as a second ranking criterion. In case of equal fitness and equal size, the ranking of the involved individuals is defined randomly.

Termination of the Evolutionary Loop

After each completion of the evolutionary loop (Figure 5.2) the fitness of each individual in the population is computed. The execution of the loop is stopped if an exit criterion is fulfilled:

- best fitness value smaller than a predefined value,
- best fitness value and average fitness constant over a certain number of generations or
- maximum number of generations reached.

In all cases the fittest individual is returned as solution of the optimization problem.

5.2.5 Genetic Operators

Selection Operator

The used selection technique has a major impact on the performance of the GP system [Koza 92]. The selection techniques are typically probabilistic to avoid greediness, and therefore, allowing even low fitness individuals to survive. This ensures that even after hundreds of generations a diverse DNA pool exists. In other words, the selection method controls the *evolutionary pressure* applied by the system. If the pressure is too high, the population diversity drops. If it is too low, the system has no pressure to evolve. In this work, we consider three different selection schemes: tournament selection, best half selection and the fitness uniform selection scheme (FUSS).

Tournament Selection The most common selection scheme is tournament selection [Koza 92, Poli 08]. As the name suggests, a number of tournaments are held. For each tournament N individuals are selected randomly from the current population. The winner (fittest individual) of each tournament is selected as potential parent. The selection pressure can be adjusted by the tournament size N.

Best Half Selection The best half scheme simply ranks the individuals by the fitness values and selects the fittest 50 %. In contrary to the above description, it is a greedy scheme. However, it was recommended to us by Stefan Mandl[1] [Mand 05] as a successful and fast scheme.

Fitness Uniform Selection Scheme The fitness uniform selection scheme is a relatively new approach to preserve diversity in a population [Hutt 01]. FUSS biases selection toward sparsely populated fitness values instead of biasing it toward higher fitness. The scheme first computes the interval of the fitness values in the current generation (Figure 5.5). A value in this interval is chosen randomly. The individual whose fitness is most similar to the chosen fitness is considered as potential parent.

[1] Member of the Department of Computer Science, Database and Information Systems, Augsburg University, http://www8.informatik.uni-augsburg.de/en/chairs/dbis/db/staff/mandl, last visited 2012-05-27.

If this fitness value is shared by more than one individual a tree size based ranking is used. It was demonstrated that this approach works well for complex learning tasks [Hutt 01, Legg 04].

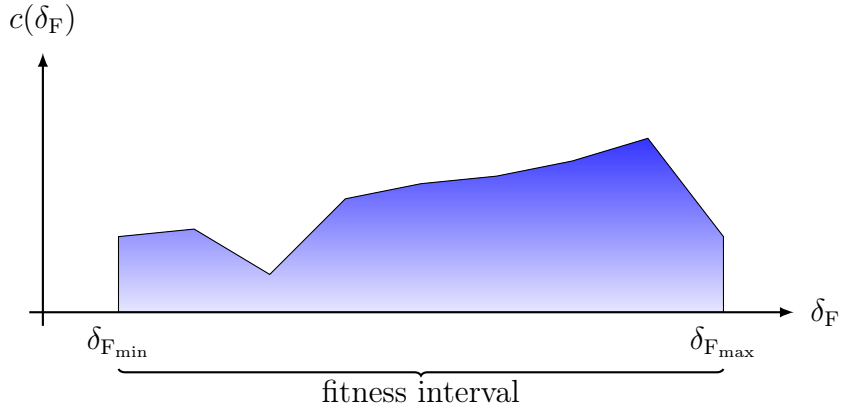


Figure 5.5: FUSS randomly samples the fitness interval $\delta_F \in [\delta_{F_{min}}, \delta_{F_{max}}]$ and selects individuals uniformly. $c(\delta_F)$ denotes the number of individuals with fitness δ_F.

Crossover Operator

The crossover operator denotes the combination of two parent individuals to one offspring (two offsprings are rarely used), see Figure 5.6. Note: It is possible to cross one individual with itself. For our optimization problem, we developed a type-constrained crossover operator [Mont 95]. It only selects crossover points in the ASTs, which fit with respect to the return type of the nodes. Our algorithm randomly selects a crossover point in the first individual. For the second crossover point only nodes which match the return type of the first crossover point are considered. Following Koza's suggestion, we constrain the operator to chose crossover points not uniformly. Similar to his work, only 10 % of the crossover points are terminal nodes and 90 % are function nodes [Koza 92].

Mutation Operator

For sampling the solution space adequately it is important to keep the population diverse. A diverse population is a prerequisite to evolve good solutions, while maintaining the performance of the so far best solutions. Badran et al. [Badr 10] showed that using mutation, especially allowing root node mutation, is crucial for keeping a population diverse without using explicit diversity operations.

Our GP-framework supports *subtree* and *point mutation* [Poli 08]. In subtree mutation, one node of the tree is chosen randomly and replaced by a randomly generated subtree with fitting return type. Subtree mutation will be applied only once per individual. Point mutation (sometimes referred to as node mutation) is more gentle to the tree and can be interpreted as an equivalent of the bit-flip mutation used in genetic algorithms. If point mutation is used, every node in the tree may be target of mutation. If a node is chosen it is flipped, e. g. by replacing a terminal with another terminal.

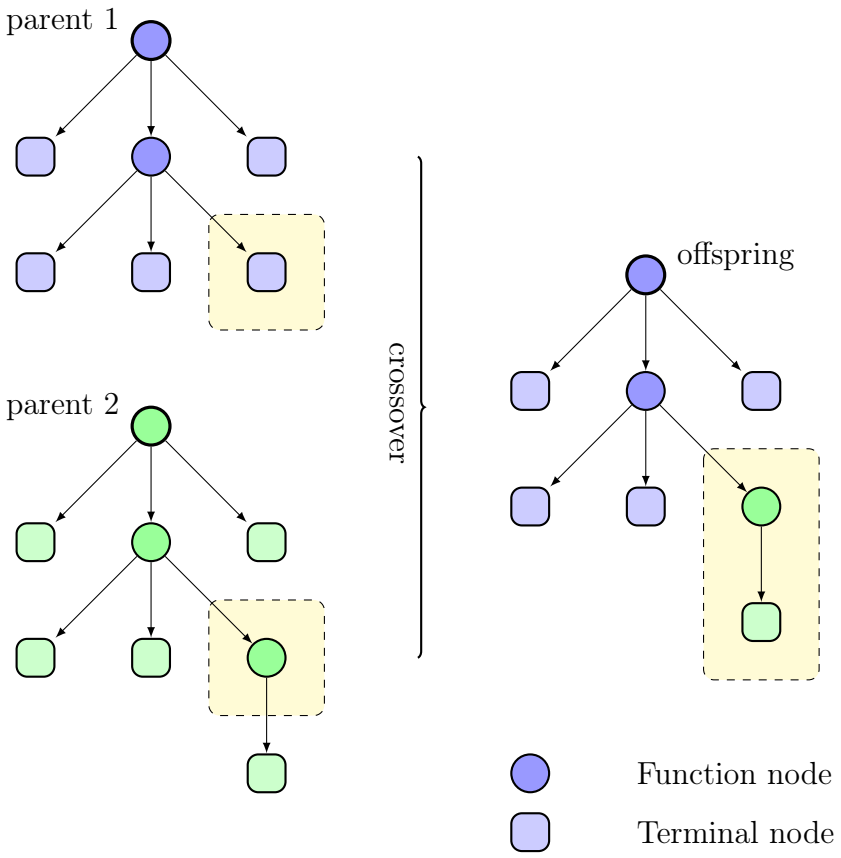


Figure 5.6: Our GP crossover operator selects (type strong) a crossover point in each AST. The offspring is a result of the swapping of subtrees at the crossover point.

If the mutation operator selects the root node for mutation, the individual is initialized again. Thus, both mutation operators support root node mutation.

5.2.6 Genetic Programming Framework

The implemented genetic programming framework is based on the field guide by Poli et al. [Poli 08] and depicted in Figure 5.7. The common control parameters for GP are summarized below, followed by a brief description of the bloat problem.

Control Parameters

Figure 5.7 shows the evolutionary loop enhanced by the details of our implementation. The different options, e. g. point or subtree mutation, are controlled by an xml file. The list of used control parameters and their default values is given in Table 5.2. The stated values are based on experience and were inspired by the literature, e. g. [Koza 92, Poli 08].

Elitism A common technique to preserve promising intermediate results is elitism. If it is enabled, then the best individual is kept regardless of the selection mechanism.

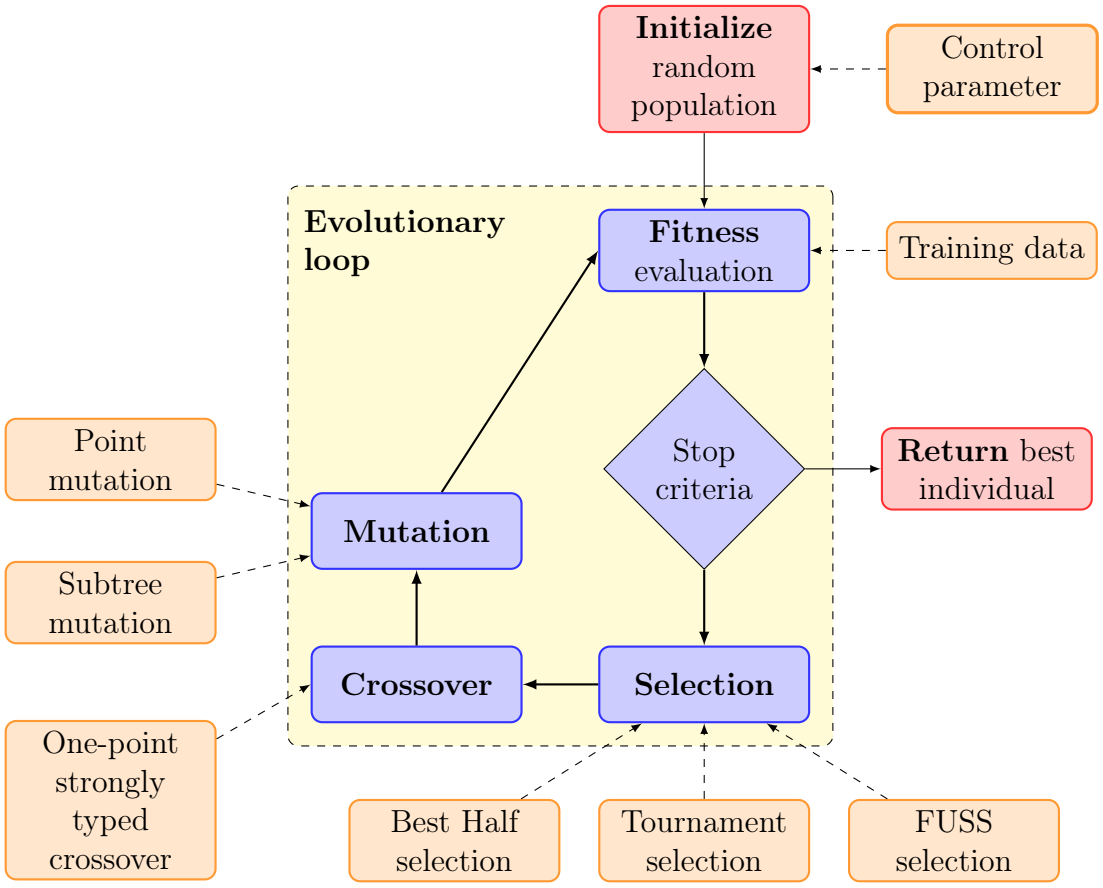


Figure 5.7: Scheme of the genetic programming framework. The evolutionary loop is in the center (blue). Start (initialization) and final result are indicated with red, while the possible parameter selections are colored in orange.

Parameter	Value
Population size	200
Max number of generations	100
Stop threshold (fitness)	$\delta_F \leq 10^{-6}$
Fitness weighting factor	$\alpha = 0.9$ (see Eq. 5.3)
Initial tree size	350
Max tree size	1000
Selection strategy	Tournament 2 (see Section 5.2.5)
Crossover probability	0.5
Crossover non-uniform selection	10 % terminals and 90 % functions
Mutation method	Subtree mutation (see Section 5.2.5)
Mutation probability	0.5
Elitism	enabled

Table 5.2: Default parameter values for the GP-framework. The values are based on experience and literature suggestions [Koza 92, Poli 08].

Thus, elitism limits the randomness factor of GP. Then again, one fixed individual (per generation) does not have a high impact on the overall randomness of the system.

Tree Size and Bloat A common method for efficient GP evaluation is to limit tree growth to prevent bloat. Bloat denotes tree branches with useless code, which increase the size of the tree, and consequently the execution time of the individual, without any benefit to the result.

Restricting the tree size can be achieved in multiple ways. One way is to border the tree depth. We agree with Parkins [Park 04] that this is a poor choice, because it limits the power of crossover and mutation operators and forces us to select a suitable depth. A more suitable way is to fix the number of allowed nodes in the tree. This preserves the strength of the genetic operators, while stopping the tree to grow infinitively. In our GP framework, the constraint is enforced as a special type of *size enforcing mutation*. This was inspired by the work of Crawford-Marks [Craw 02]. Instead of applying subtree or point mutation, randomly selected branches of the individuals are deleted (replaced by a fitting terminal) in cases where the tree size exceeds the maximum value.

Generational versus Steady-State

In our work, we consider the classical generational GP model. It employs two populations: parents and offspring. In the steady-state model the offspring is merged with the current population, and thereby, replacing older members. It can be considered as an online update of the generation, because the new members are part of the selection process. The update of a generation continues until all old members are replaced. We use the generational approach in our work, because the steady-state model is known to suffer from the negative effects of *genetic drift* [Kinn 93, Mont 95].

5.2.7 Evaluation of the GP-Framework

An interesting fact about data set $\mathcal{S}_1$ is that the coordinate system is different for each sample, because of the acquisition procedure [Gao 09]. One claim about GP is, that there is no need to preprocess the data [Poli 08]. This might not be true for any GP-problem, but in our case the data preprocessing would only involve rotation and translation correction. We assume, that our GP-framework can handle this and consequently did no preprocessing.

To simplify the optimization task, we only considered the left ear case. From the remaining 65 samples, 35 are used for training and 30 for validation. In [Sick 10b], we extensively analyzed the influence of the different parameters listed in Table 5.2. The main results are summarized below.

Population size The population size was evaluated in the range $[50, 500]$ yielding 200 as a good trade off between final fitness value and execution speed.

Maximum number of generations Our experiments verified the statement of Poli et al. [Poli 08] that the best solutions are usually generated in the early generations. Therefore, the algorithm is stopped after at most 100 generations.

Initial tree size The individuals were randomly created using the *grow* method. The initial tree size was evaluated in the range $[50, 500]$. It could be shown that it has a high influence on the diversity of the population. The value of 200 nodes in an initial grown tree was performing best.

Maximum tree size Huge trees have on the one hand a high potential for useful code. On the other hand they often contain large amounts of bloat, which slows down the evolution process. 1000 nodes as maximum tree size was identified as a good trade off between execution and fitness performance.

Stopping criteria The algorithm is stopped, if the fitness of the best individual is below the threshold 10^{-6}. However, in our experiments the maximum number of generations stopped the algorithm more often.

Fitness weighting factor α The orientation of the cutting plane is of major interest. Therefore, the factor was set to 0.9.

Selection strategy The best performing selection strategy was tournament selection with tournament size two. It was superior than tournaments with larger size and also performed better than the other two introduced selection strategies.

Resulting Rule

After evaluation of the control parameters based on the fitness values, we manually analyzed the rules defined by the best individuals for ten GP runs. On average approximately 70 % of the program did not influence the target plane (bloat). The two major findings are:

1. Compared to the currently used plane in ESAM1 it should be updated by a small shift along the plane normal.
2. A small rotation of the target plane should be done. The rotation is defined by the vector from tragus concavity to anti-tragus concavity. The rotation essentially decreases the remaining material below the helix-peak.

Ear side	$\mu_0(\pm\sigma_0)$	$\mu_1(\pm\sigma_1)$	Improvement in %
left	0.333 (0.235)	0.259 (0.266)	22.22
right	0.246 (0.078)	0.158 (0.067)	35.77

Table 5.3: Fitness evaluation with and without rule update. μ indicates the mean fitness value and σ the standard deviation. Subscript 0 denotes the original rule and subscript 1 the updated rule.

To verify these findings we compared the performance of the expert system before and after including the new rules into the knowledge base. Tab. 5.2.7 shows that the fitness improved about 22 % on the left samples and 35 % on the right samples. The latter surprised us positively since we only used samples of the left side during our evaluation.

5.2.8 Application Evaluation and Conclusions

After the positive test results of the GP evaluation on data set $\mathcal{S}_1$, it was shown to, and evaluated by, the process experts. During these evaluation sessions, we had to acknowledge that the idea of using GP for rule learning did fail for our optimization problem. The experts consistently rated (personal communications) the original rule higher than the updated one. Also, only using the shift or the rotation exclusively did not improve the ratings. If just the shift was applied, they were concerned about too less material at the inter-tragal notch point. Applying the rotation only resulted in more space at the inter-tragal notch, but reduced the helix (cymba) height too much. The original rule provided the better trade off and was always preferred by experts. Hence, we did not continue to evaluate the GP learning approach for other rules of the KB.

Nevertheless, one positive result of the GP evaluation was the evidence provided by the evolved rules about our feature assumptions. Most of the investigated rules used the three feature points: tragus concavity, anti-tragus concavity and anti-helix concavity to define the initial cutting plane, which matches our implementation. Some rules used the bottom contour plane as initial plane, which has a similar orientation in most cases, but fails for bad scans with large areas of excess material.

5.3 Semi-automatic Rule-learning

The goal of the semi-automatic approach, is to identify high, medium and low performing rules in a first step and improve the low performers in the second step. This can be achieved by analyzing data set $\mathcal{S}_2$. The classification of the rules based on performance enables the knowledge engineer to focus on rules with low performance. To further support the knowledge engineer an estimation of the necessary rule change is extracted from the data.

5.3.1 Rule Types

Note: In order to facilitate the reading, we may repeat facts concerning the HA design process and its rules already presented in the previous chapters. A rough scheme of the design process is given in Figure 5.8 (and 2.13). It distinguishes between plane, point, area and *other* process steps. The latter are not of interest in the following, because they are either only used for showing a modeler a certain color mapping or allowing component placement, which is only considered partially in ESAM so far. The other three types are always connected to a plane, point or area set by the expert system. Furthermore, a couple of plane related process steps are marked as manual steps. Manual steps were introduced to cope with missing functionality of the CAD software or agreement of the expert designers. For example, it was planned to define the *Round Shell Around Faceplate Component* in terms of splines. However, the CAD software did not support this feature in time. Thus, it was replaced by a manual step, which preselects the rounding tool as a workaround.

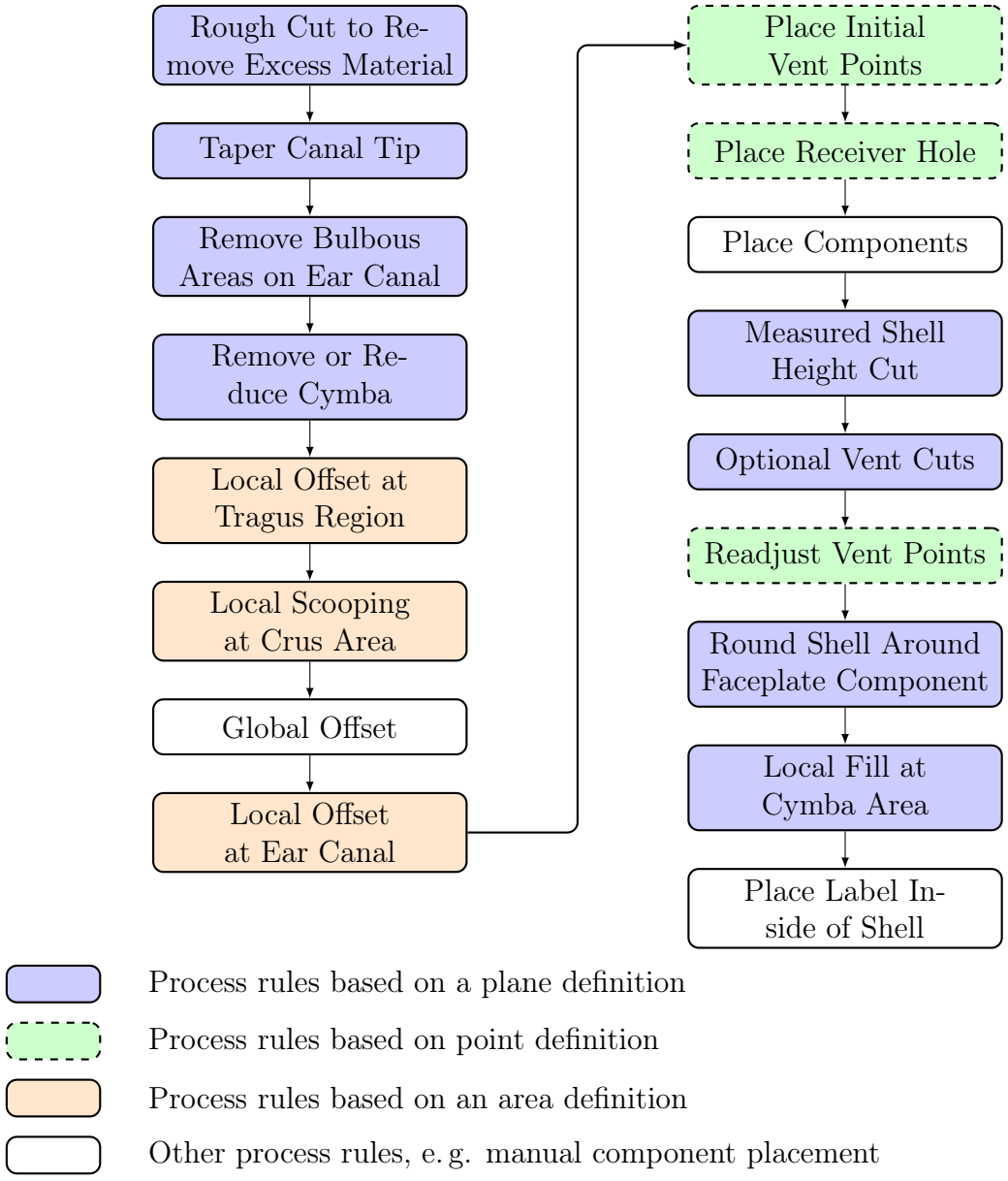


Figure 5.8: Workflow of a hearing aid design task. The different steps are marked as plane, point or area based process rules. The order is not entirely mandatory, but depends on the design goal: either the smallest possible instrument or the best possible vent. Depicted is the vent-centric workflow (two vent steps). The shown workflow is not complete, meaning that a couple of steps are grouped in one step for the sake of clarity, e. g. *Remove or Reduce Cymba*. Furthermore, a certain amount of process steps is device type (CIC, ITC, ITE) dependent, e. g. *Local Scooping at Crus Area* is only valid for ITE devices.

5.3.2 Classification of Rules based on Performance

The classification of rules in high, medium and low performing rules is based on the amount of user interaction in the process step and the difference between ESAM

suggested and finally applied operation parameters. At first the used terms are defined.

Definition 5. Completeness rate δ_c: A stored guide rule is considered as complete, if setup and commit node exist (Listing 5.1).

Definition 6. Acceptance rate δ_a: If the difference between a recorded setup and commit node is smaller than an expert defined threshold, it is acceptable. The acceptance rate is only calculated for complete rule records. The used thresholds are in sync with the ones used for the feature detection evaluation (see Section 3.4.3).

- planes: plane orientation difference $\leq 15°$ and plane location difference $\leq 3\,\text{mm}$
- points: Euclidean distance $\leq 3\,\text{mm}$
- areas: size difference $\leq 30\,\%$

Definition 7. User interaction rate δ_u: The average number of user interactions applied during rule execution. Note: A high number of user interactions shows on the one hand that the ESAM suggestion was not sufficient and on the other hand that the required selection was difficult to achieve.

Definition 8. The rules are classified in low, medium and high performing rules as defined in Table 5.4.

	$\delta_a \leq 0.7$	$0.7 < \delta_a \leq 0.85$	$\delta_a > 0.85$
$\delta_u \leq 1.0$	low	medium	high
$1.0 < \delta_u \leq 2.0$	low	medium	medium
$\delta_u > 2.0$	low	low	low

Table 5.4: Rule performance classification matrix.

The chosen values are based on discussions with expert operators. In their opinion, a rule should be accepted at least in 70 % of the cases, before it can be considered as acceptable. The user interaction values are derived from the consideration that an automatic system should require less than 1 interaction per rule. If the user interaction exceeds 2 on average, it is an indicator that the preselection done by ESAM has only a small benefit for the user.

Definition 9. A design record is considered as ESAM-complete, if the *Place Label Inside of Shell* rule is applied last.

In other words, ESAM was used to model the device.

Definition 10. A design record is considered as ESAM-valid, if it contains at least ten rule recordings.

The definition covers all ESAM-complete records and all records, where there was a clear indention to use ESAM.

Rule Performance

The completeness rate, acceptance rate and average number of user interactions per guide rule are listed in Table 5.5. The performance values discussed in the following were computed by the Rule Analyzer (RA), which is a tool for the analyzing of design records, which we developed.

Rule	δ_c	δ_a	δ_u	performance
Rough Cut to Remove Excess Material	0.94	0.82	1.52	medium
Taper Canal Tip	0.87	0.54	1.90	low
Remove or Reduce Cymba	0.97	0.78	0.84	medium
Local Offset at Tragus Region	0.99	0.18	0.08	low
Local Offset at Ear Canal	0.98	0.77	0.45	medium
Place Initial Vent Points	0.99	0.91	1.31	medium
Place Receiver Hole	0.99	0.88	2.70	low
Measured Shell Height Cut	0.96	0.65	2.71	low
Measured Shell Height Cut (ITC)	0.95	0.62	22.60	low
Optional Vent Cuts	0.77	0.86	0.85	high
Readjust Vent Points	0.99	0.98	2.27	low
Local Fill at Cymba Area	0.97	0.78	0.18	medium
Place Label Inside of Shell	0.95	0.61	0.53	low

Table 5.5: Performance of the ESAM1 system. Similar guide rules (using similar design rules) are grouped together. Note: δ_a and δ_u are computed for each complete guide rule, regardless if the record is ESAM-complete.

The results listed in Table 5.5 indicate seven low performing, five medium performing rule groups and one high performing rule group. Based on these performance values combined with feedback from the designers, we concentrate our learning efforts on the following rule groups.

- *Taper Canal Tip*
- *Place Receiver Hole*
- *Measured Shell Height Cut*
- *Place Initial Vent Points* together with *Readjust Vent Points*
- *Local Fill at Cymba Area*

We neglect the labeling rule for now, because the data can not provide the information if the placement failed due to the position suggested by the rule or if the label string was too long to be fitted inside the shell. The latter happens regularly for CIC devices and requires that the user shortens the label string about one or two characters. Also, the *Local Offset at Tragus Region* rule experienced little modifications, because the rule is not mandatory and rarely applied. Thus, feedback was limited, therefore, we focused our efforts on other rules.

5.3.3 Drawing Conclusions from User Modifications

After identification of the low performing rules, we consulted with the process experts to clarify what exactly fails in the rule suggestion. This is followed by a detailed evaluation of the user modifications for each interesting rule using data set $\mathcal{S}_2$. The combined information could be used by the knowledge engineer to improve the rules.

The difficulty thereby is the missing feature information (they are not stored in the design record) and that no common coordinate system exists. Thus, it is possible that the user rotated a plane about a certain angle, but the motivation, e. g. because of a certain feature point or to generally increase a certain area, cannot be extracted easily. To cope with this situation, we use relative changes between different process steps, e. g. the plane defined during the *Rough Cut to Remove Excess Material* rule application can be used to identify if the user elongated or shortened the canal by moving the canal taper plane up or down in respect to the rough cut plane.

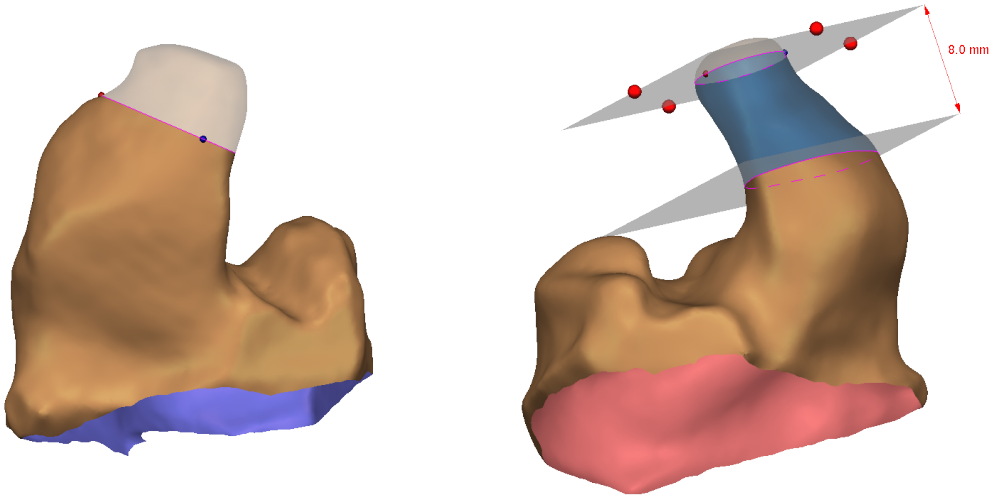
One other major drawback of learning methods in the HA design environment is the *shot at a moving target problem*. Each and every iteration of ESAM contains changes necessary to cope with the changed needs of the design process. These changes may involve the exchange of rules, the creation of new rules, the modification or the removing of rules. A typical example is the modification of the meaning of an option. For example, the ear canal is usually tapered based on a length option. The length option is defined relative to the first and second bend. In order to design smaller hearing aids the canals are designed now longer than in the past, because some of todays components are small enough to be placed inside the ear canal. In order to cope with this ever changing situation, the learning results are evaluated and interpreted by the knowledge engineer before implementing the modifications in the KB.

Improving the Position of the Canal Taper Plane

The detailed evaluation of the *Taper Canal Tip* rule is listed in Table 5.6. An example, of the canal taper operation is shown in Figure 5.9.

Record property	ESAM1	ESAM2	ESAM3
Number of rule occurrences	1688	1539	10791
Number of complete rule occurrences	1502	1365	9859
Completeness rate δ_c	0.89	0.89	0.91
Acceptance rate δ_a	0.54	0.45	0.53
User interaction rate δ_u	1.29	1.26	0.97
Average normal angle error in degree	7.67	7.34	9.89
Average location error in mm	1.89	1.93	1.78
Plane too low	0.01	0.01	0.01
Plane too high	0.64	0.68	0.66

Table 5.6: Result of the RA for the *Taper Canal Tip* rule for the different ESAM versions.



(a) Canal taper operation using a single plane.

(b) Canal taper operation using two planes, which define an area where conical rounding is applied.

Figure 5.9: Two examples of the *Taper Canal Tip* rule.

A constant problem for all versions is that the plane is placed too high by the rule. This is especially unfortunate, because the expert modelers with whom we discuss rule changes complain about the opposite case. Nevertheless, we were able to reduce the location error slightly with the current version (ESAM3).

More severe is the error in terms of orientation. The position and orientation of the canal taper plane is mainly defined by the two feature planes: first bend and second bend. The position is further adjusted by a canal length option given for each HA order. This option is transformed into a factor $s \in [0, 1]$, where $s = 0$ would result in the height of the first bend plane and $s = 1$ in the height of the second bend plane. In Figure 5.10 the input features for the rule are displayed.

The orientation $\boldsymbol{n}_{\text{ct}}$ of the canal taper plane is defined by the normals of the two bends ($\boldsymbol{n}_{\text{fb}}$ and $\boldsymbol{n}_{\text{sb}}$) and the canal centerline normal at the plane position $\boldsymbol{n}_{\text{cl}}$, see Eq. (5.4).

$$\boldsymbol{n}_{\text{ct}} = \frac{\boldsymbol{n}_{\text{fb}} + \boldsymbol{n}_{\text{sb}} + 2 \cdot \boldsymbol{n}_{\text{cl}}}{4} \tag{5.4}$$

The above equation relies heavily on the bend features and the centerline. The results in Table 5.6 show that this only was accepted in roughly 50 % of the cases. The main reasons for this low performance are due to the moderate performance of the feature detector for these features (see Table 3.15). For example, the mean orientation error of the second bend is equal to the orientation threshold of 15°. In addition, due to the impression taking procedure (see Section 2.4.4) the centerline is usually stable until the second bend. Past the second bend the deformations at the canal tip introduce artifacts in the centerline. In order to improve the performance, the rule defining the normal was simplified in the subsequent ESAM versions:

$$\boldsymbol{n}_{\mathrm{ct}} = \boldsymbol{c}_{\mathrm{sb}} - \boldsymbol{c}_{\mathrm{fb}}. \tag{5.5}$$

Eq. (5.5) uses the location of both bends, but ignores the often unreliable orientations. The points $\boldsymbol{c}_{\mathrm{fb}}$ and $\boldsymbol{c}_{\mathrm{sb}}$ denote the center points of the first bend and second bend. The centerline was excluded as well, because of the tip deformation problematic. Furthermore, the height factor s was modified in accordance with the design experts. As a result, the plane may be located past the second bend for certain order options.

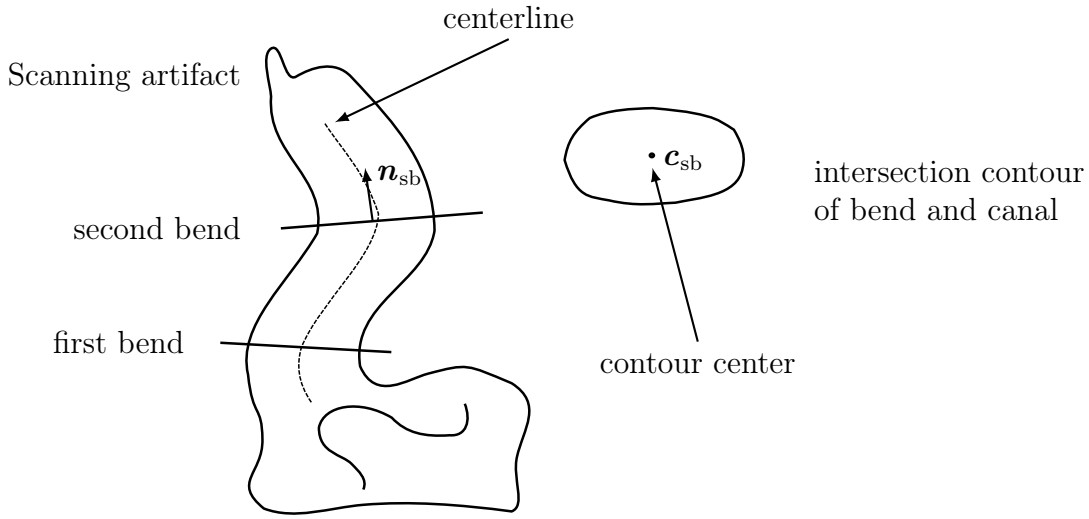


Figure 5.10: Features employed by the *Taper Canal Tip* rule. On the left a side view of an ear impression is shown and on the right the intersection contour of a bend plane with the canal. Depending on the impression the length of the canal can vary significantly. In the drawing a very long canal is shown, usually it ends shortly after the second bend and the centerline ends there as well. The centerline after the second bend is usually unreliable, because the shape of the tip is often distorted by artifacts.

Table 5.6 shows the evolution of the rule in the three successive ESAM versions. While we received positive feedback for the rule modifications by our key experts, the measured performance actually dropped for ESAM2. It recovered for ESAM3 having a similar performance as ESAM1. One explanation for the performance drop is given by the disappointment of the users. After getting used to correct the taper plane more or less every time using ESAM1, they did not change this behavior immediately after the release of ESAM2. Nevertheless, the current rule improved the location result and reduced the required user interaction. Hence, the modifications applied to the rule had a positive impact for ESAM3, but the impact is small.

In order to improve the rule further, we plan to re-engineer the centerline detection. If the reliability of the centerline is ensured, we are confident to improve the orientation of the suggested taper plane.

Improving the Position of the Receiver Hole

The detailed evaluation of the *Place Receiver Hole* is listed in Table 5.7. An example of the rule task is shown in Figure 5.11.

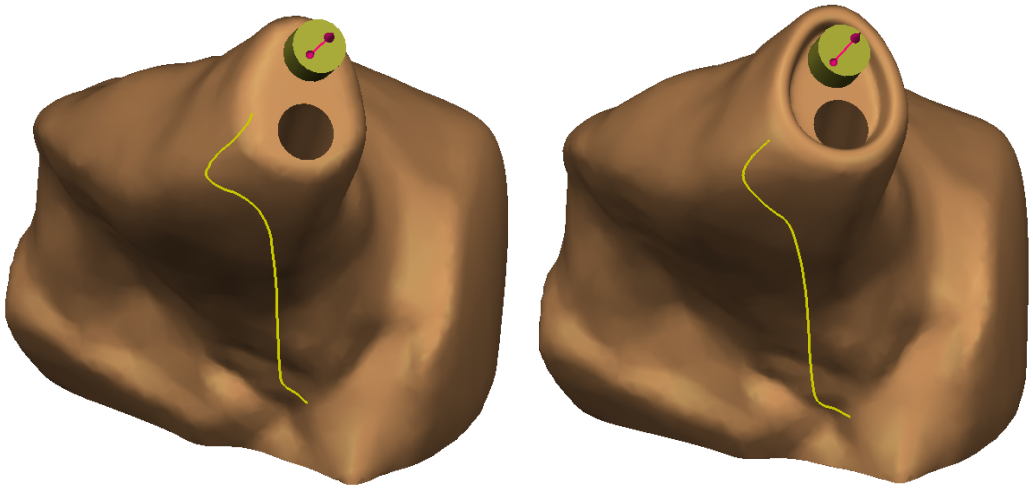


Figure 5.11: The receiver hole has to be placed next to the vent inlet. It has to be ensured that it does not intersect with the vent inlet and the tip contour. The available tip area can be small, especially if a wax guard structure is in place as shown on the right.

Record property	ESAM1	ESAM2	ESAM3
Number of rule occurrences	1961	1565	10832
Number of complete rule occurrences	1941	1560	10772
Completeness rate δ_c	0.99	1.0	0.99
Acceptance rate δ_a	0.88	0.86	0.82
User interaction rate δ_u	2.70	3.77	3.29
Average distance error in mm	2.47	1.95	1.98
Receiver too close to vent	0.10	0.28	0.23
Receiver too far from vent	0.38	0.60	0.27

Table 5.7: Result of the RA for the *Place Receiver Hole* rule for the different ESAM versions.

The rule performance in terms of completion and acceptance rate is high for all versions. It has to be noted that the acceptance threshold of 3 mm is rather large compared to the overall tip area, which mostly has a diameter of about 10 mm. The opposite applies for the average number of user interactions. In this case the counted user interactions include the placement of the receiver component in the canal. Hence, the average number is high. In order to analyze the user interactions applied to the suggested position, we used the vent location as reference. If the distance between the two centers increased about more than 0.5 mm, we conclude that the receiver was placed too close to the vent. In the opposite case that it was placed to far away from the vent. The severeness of both cases is difficult to judge, because information what caused the reposition, e. g. receiver position was colliding with the vent or was placed on the tip contour, is not available. Results from interviews and our experience show that the majority of changes is meant to increase the cosmetic appearance, which

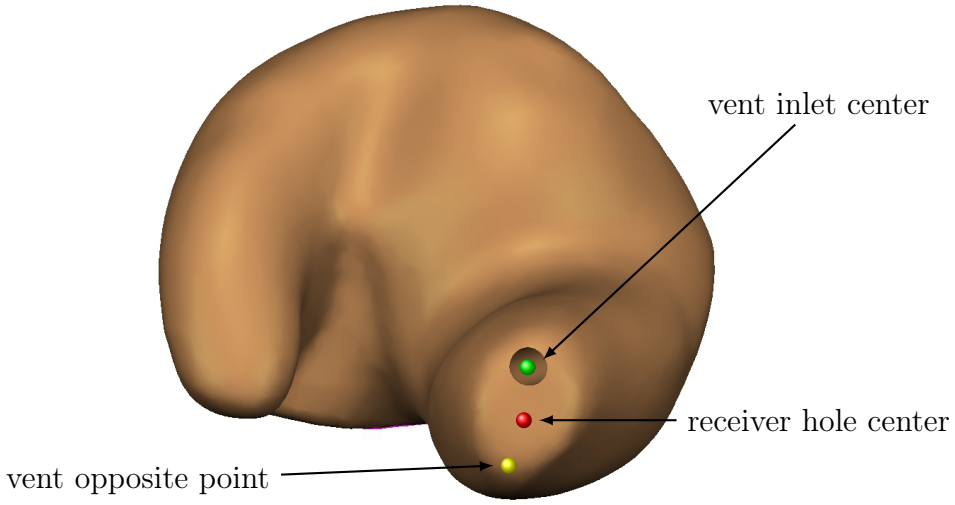


Figure 5.12: The receiver hole has to be placed together with the vent inlet on the tip of the canal. The vent inlet is placed first, therefore, the receiver hole has to be fitted in the remaining space ensuring a minimal distance to the vent of at least vent wall thickness, vent radius and a small offset for collision avoidance. A cosmetically appealing receiver hole position is the center of the remaining area.

is supported by the fact that the average reposition distance is smaller than the acceptance level threshold.

The ESAM1 rule for placing the receiver hole on the tip initialized the hole position with the center of the canal tip contour. If this position already fulfilled the minimal distance to the vent the algorithm finished immediately. In the next step, the receiver hole was positioned at the average of the given vent point, the current receiver hole position and the vent opposite point. The latter is visualized in Figure 5.12. The rule performed well on average, but the average error in mm is high considering the size of the tip area. An explanation is the usage of shifts based on the feature points, which could push the position beyond the tip area or away from the principle components, if the feature points are detected not accurate enough.

In ESAM2, we modified the algorithm to be independent of the opposite vent point in order to reduce the average distance error and user interactions. Especially in case of narrow tips or if special waxguard structures (Figure 5.11) were integrated the found position was cosmetically not appealing in ESAM1. The modified algorithm is solely based on the tip area and the current vent location as shown in Figure 5.13. In the previous rule, the receiver hole was placed too far away in approximately 40 % of the cases, which we think is related to the shifting procedure and the include offset to avoid collisions. The region-based rule did not improve this behavior, but seemed to require more diverse changes. In general the updated rule performs better concerning the distance error (standard deviation reduced from 6.45 mm to 2.28 mm).

To reduce user interactions, we integrated additional checks to ensure validity of the used feature area in ESAM3. The area could leak out from the tip in certain situations, which essentially ruined the computed receiver position. Theses checks solved the problem of the high amount of positions which were computed too far away

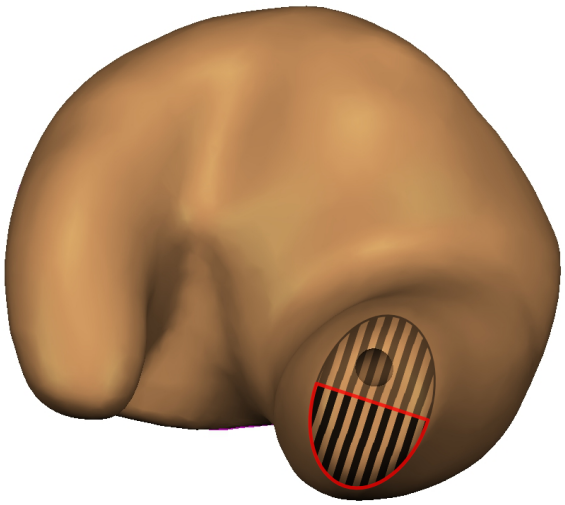


Figure 5.13: The opaque black stripes mark the tip area. The darker red framed area, is the reduced target area utilized in the ESAM2 receiver hole placement rule.

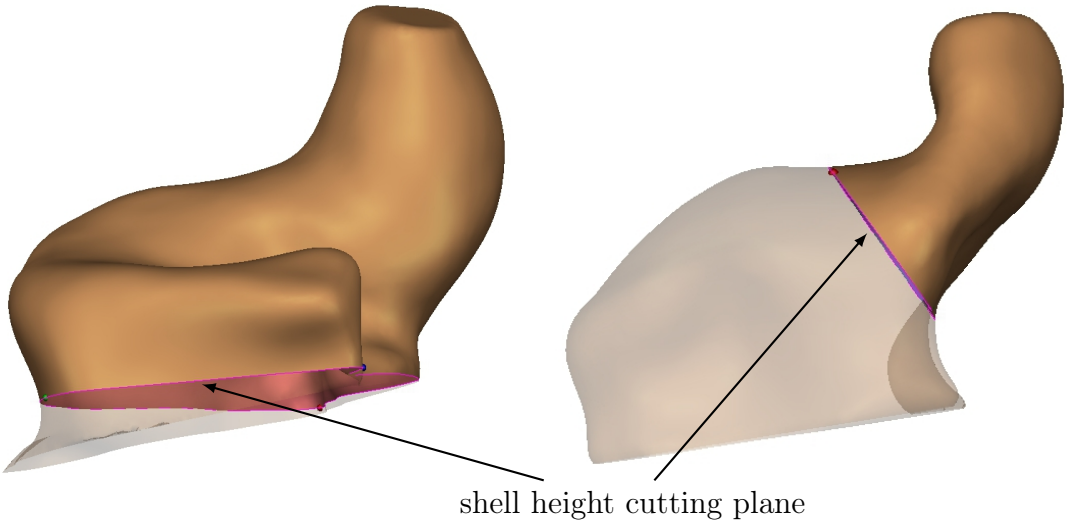


Figure 5.14: Examples for the *Measured Shell Height Cut* rule. On the left an ITE and on the right a CIC is shown.

from the vent, which went down from approximately 60 % to 27 %. As a consequence, the user interactions rate decreased slightly.

Improving the Position of the Measured Shell Height Cuts

The *Measured Shell Height Cut* is different for the different HA sizes: ITE, ITC and CIC (see Section 2.4.4). Examples for the rule application are given in Figures 5.14 and 5.15. Thus, Table 5.8 lists the results for each device type separately.

Measured Shell Height Cut for ITEs In case of ITEs the *Measured Shell Height Cut* rule shows a good performance. The average error is low and the acceptance rate

Record property	ITE	ITC	CIC
Number of rule occurrences	469	989	344
Number of complete rule occurrences	462	944	339
Completeness rate δ_c	0.98	0.95	0.98
Acceptance rate δ_a	0.93	0.62	0.26
User interaction rate δ_u	0.21	22.59	0.71
Average normal angle error in degree	4.48	-	16.72
Average location error in mm	0.92	3.26	4.84

Table 5.8: Result of the RA for the different specifications of the *Measured Shell Height Cut* for ESAM1.

is very high. Therefore, the rule was only modified minimally in the following versions. The results for ESAM2 and ESAM3 show a similar rule performance. For ESAM3, $\delta_a = 0.97$, $\delta_u = 0.07$, and the location and orientation error is reduced to 0.61 mm and 0.56° respectively.

The rule is based on three distinctive feature points: tragus concavity, anti-tragus concavity and anti-helix concavity, which define the initial cutting plane. The plane is adjusted based on features and measurements indicating the shape and height of helix and concha. Furthermore, the current open contour at the bottom side of the impression is used as a boundary condition.

Measured Shell Height Cut for ITCs The rule performance for ITCs shows a completely different behavior. The reason for that is the so-called hinge tool applied for this design task. The tool works like a hinge, where the axis is defined with two points - the hinge points and the third point is used to define a plane which can hinged rather than shifted. Furthermore, the electronics are snapped to the hinge plane (Figure 5.15). Thus, the design task involves placing the component as deep as possible into the concha bowl and setting the correct hinge points.

Since the angle of the shell height cutting plane is constrained by the faceplate and concha shape, it is not reasonable to analyze the angular difference between setup and commit node. The most important features for the hinge tool are the two hinge points placed on the tragus side. They should be located on the intersection between the inter-tragal notch ridge respectively the crus-ridge and a plane defined by the tragus concavity and the up direction of the impression. The latter is extremely difficult to define, because it depends on the orientation of the canal, the height of the concha bowl and the helix height. The performance results for the ESAM versions are given in Table 5.9.

The initial version of the *Measured Shell Height Cut* rule for ITCs was mainly based on the point features canal-concha intersection, canal-crus intersection and concha-peak in order to define the required up direction. However, as the results in Table 3.13 show that especially the first two have a low detection quality. Consequently, the orientation of the hinge axis was often off resulting in a low acceptance rate of 62 %. Furthermore, the number of user interactions is very high, because the faceplate lowering was done in this design step as well. The lowering requires to

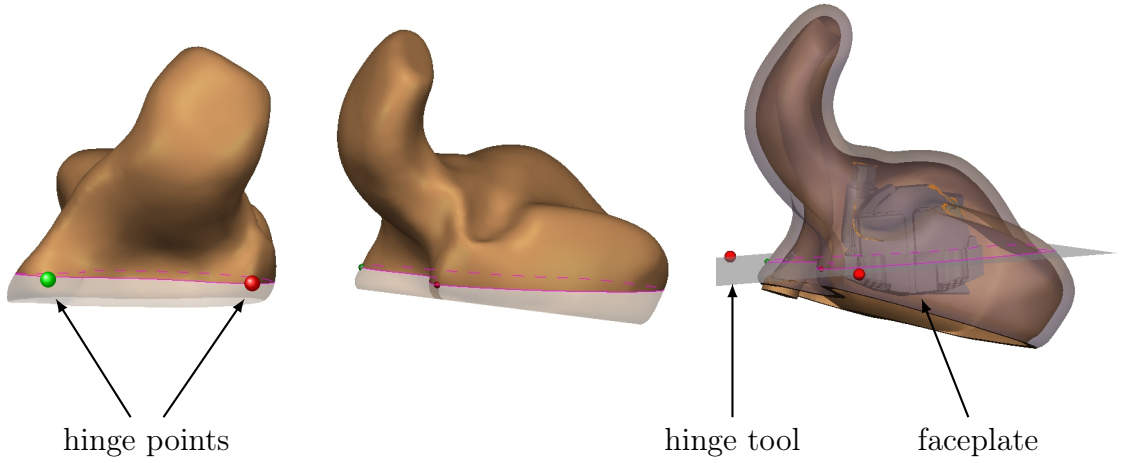


Figure 5.15: The hinge tool used in case of ITC devices for the faceplate position includes the definition of the measured cut. The two hinge points at the tragus side define the hinge axis. The third point on the concha bowl can be dragged to lower the faceplate. The rule places the three points and executes a lowering algorithm after approval by the user. The result is shown on the right. In contrast to the measured cuts for CICs and ITCs the hinge point shifts applied by the user are analyzed instead of the plane location and orientation.

Record property	ESAM1	ESAM2	ESAM3
Number of rule occurrences	989	1074	5538
Number of complete rule occurrences	944	1064	5446
Completeness rate δ_c	0.95	0.99	0.98
Acceptance rate δ_a	0.62	0.74	0.86
User interaction rate δ_u	22.60	0.41	0.20
Average distance error in mm	3.26	2.60	1.23
Hinge points moved down (closer to bottom)	0.13	0.21	0.07
Hinge points moved up (away from bottom)	0.25	0.18	0.15
Hinge points moved in different directions	0.43	0.41	0.14
Hinge points stable	0.19	0.20	0.64

Table 5.9: Result of the RA for the *Measured Shell Height Cut* rule for ITCs for the different ESAM versions. The average distance error describes the average shift done by the user for both hinge points. The knowledge extraction focused on the relationship between hinge points and bottom contour. It was analyzed, if the hinge points were moved up or down.

hinge the plane and to shift and rotate the faceplate avoid collisions. In the following ESAM versions, the lowering was separated from the definition of the hinge points, which explains the huge decrease in user interactions.

After learning that canal-concha intersection and canal-crus intersection were limiting the performance of the design rule, we worked on improving these features resulting in a better performance in ESAM2. The problems caused by the features

can be seen in the fact that often (43 %) the hinge points had to be readjusted up and down, meaning that the axis orientation was not correct. In the remaining 57 % of the cases the orientation was acceptable and the points were shifted up or down together.

For ESAM3 the rule was reworked using additional features to improve the location of the hinge points. The additional features are based on results of previous steps, e. g. *Rough Cut to Remove Excess Material* providing a hint for the up direction and *Remove or Reduce Cymba* as an indicator for the hinge axis direction. Finally, the achieved performance is adequate to perform the design task more or less automatically, because the δ_u indicates that only every fifths order is modified by the user and the applied modification is small. Also, the indicator for opposite movement of the hinge points was reduced significantly from 41 % to 14 %.

Measured Shell Height Cut for CICs Figure 5.14 shows the position of the measured cut for CIC devices. The cutting plane is roughly defined with the canal-concha intersection, canal-crus intersection and low of aperture feature points. As discussed for the ITC rule, canal-concha intersection and canal-crus intersection are low quality features and consequently the initial rule performance was poor. Table 5.10 list the rule performance for all ESAM versions.

Record property	ESAM1	ESAM2	ESAM3
Number of rule occurrences	397	493	2854
Number of complete rule occurrences	376	473	2803
Completeness rate δ_c	0.95	0.96	0.98
Acceptance rate δ_a	0.26	0.26	0.87
User interaction rate δ_u	4.57	4.60	1.16
Average normal angle error in degree	16.42	15.39	2.11
Average location error in mm	4.73	3.56	1.18
Relationship to *Rough Cut to Remove Excess Material* plane			
Average angle in degree	52.75	49.23	52.79
Angle increased ($>$ 5 degree)	0.21	0.20	0.02
Angle decreased ($>$ 5 degree)	0.44	0.46	0.63
Relationship to *Remove or Reduce Cymba* plane			
Average angle in degree	78.74	78.98	83.03
Angle increased ($>$ 5 degree)	0.37	0.28	0.26
Angle decreased ($>$ 5 degree)	0.28	0.31	0.40

Table 5.10: Result of the RA for the *Measured Shell Height Cut* rule for CICs for the different ESAM versions. In order to extract knowledge from the data, we analyzed the angles between the applied rough cut at the bottom and the applied cymba removal cut.

The second KB contained no modifications for this rule, because initial experiments showed that the rule performance should increase, because of the feature point improvements as already discussed for the ITC case. Unfortunately, this was not the

case. The rule performance did increase minimally in terms of the average error, but in general was constant.

Hence, for ESAM3 the rule was redesigned, to be independent of canal-concha intersection and canal-crus intersection. This was achieved similarly to the redesign of the ITC rule by including information gathered in the previous design steps. For example, the intersection of the crus-ridge and the cutting plane of the *Remove or Reduce Cymba* can be used as a replacement for the canal-crus intersection feature. The orientation provided by the rough cut plane of the initial rule helped to indicate the up direction of the impression, which could be used together with the centerline to verify the orientation of the measured cut plane. The normal of the resulting plane was a combination of the plane based on point features, the normal of the *Rough Cut to Remove Excess Material* plane and a normal interpolated from the centerline. The changes resulted into an improved rule performance. The acceptance rate more than tripled and the average errors went down to values, which are in the typical range of user differences. Comparing the average angles (rough cut plane, cymba removal plane) extracted from the data, it can be seen that the relationship between rough cut plane and measured cut is similar in ESAM1 and ESAM3. However, ESAM3 has a better performance. Hence, the change about four degrees on average in the relationship between measured cut and cymba removal plane, seems to have a major impact on the performance.

Improving the Vent Position

The design process illustrated in Figure 5.8 contains two vent related rules. The *Place Initial Vent Points* rule places the vent at an initial position before the shell is fully detailed, just to provide the designer with the information how much space the vent requires in the shell. The second rule *Readjust Vent Points*, defines the final location of the vent.

The initial rule performance was medium with a high acceptance rate. Similar to the receiver hole placement, the acceptance rate is optimistic due to the threshold of 3 mm, which is roughly one third of the canal tip size. The user interaction rate indicates that roughly both vent points were moved every time, which should not be necessary. The detailed results for all ESAM versions are listed in Table 5.11.

Record property	ESAM1	ESAM2	ESAM3
Number of rule occurrences	1752	2050	11940
Number of complete rule occurrences	1727	1998	11624
Completeness rate δ_c	0.99	0.97	0.97
Acceptance rate δ_a	0.98	0.88	0.99
User interaction rate δ_u	2.27	2.14	1.28
Average distance error in mm	0.36	4.46	0.35
Average distance of vent bottom to cymba rounding / removal plane in mm	10.12	11.15	10.45

Table 5.11: Result of the RA for the *Readjust Vent Points* rule for the different ESAM versions.

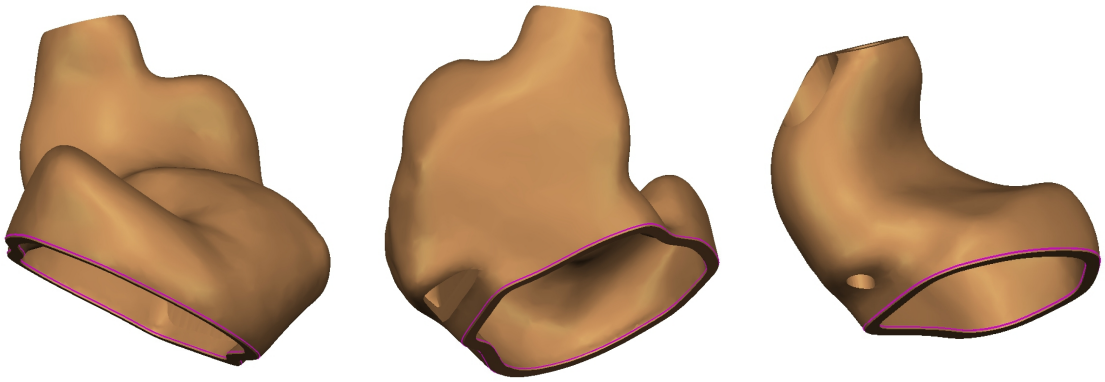


Figure 5.16: The left impression features a rectangular cut at the canal tip, while the middle impression is cut at both sides and finally the right impression has a slanted cut at the tip and a rectangular one at the bottom side. These types of cuts are very common. Their use is to increase the wearing comfort by reducing occlusion effects.

The initial vent placement is based on the intersection of the ridges: inter-tragal notch ridge, inter-tragal notch ridge with the canal tip. Depending on the device type the vent is located at the one or the other intersection point (plus and offset to fully placed on the tip). The initial rule for the vent adjustment used the information from the previous vent rule and only modified the vent points if vent cuts were applied. Due to the changes applied in between the two rules, the initial vent points have to be projected to the current shell. The top point is projected to the closest point on the shell, while for the bottom point an additional check is done to snap it to the bottom opening. In case of vent cuts, the information provided by the cuts (usually given as planes) is used to guide the projection. Example vent cuts are illustrated in Figure 5.16.

In ESAM2, the rule was not modified, because the performance was adequate for a first version. However, the results show that the performance dropped significantly, especially the distance error increased. This was mainly caused by two other changes. The first was a modification in the *Place Initial Vent Points* to better handle wax guard structures on the tip. The second was a modification in the same rule to improve the location of the bottom vent point for ITC devices. This change used the plane defined in the *Remove or Reduce Cymba* to ensure that the bottom vent point is close to the crus area. Especially the latter placed the bottom vent point in a location which is strongly affected by the *Measured Shell Height Cut* and the associated smoothing and rounding of the shell. As a result, the projection in the *Readjust Vent Points* rule yields unsatisfiable positions in some cases.

As a consequence, the initial vent rule was updated to ensure a better position of the vent top and bottom point. The top point rule is similar to the receiver hole placement and based on the tip area. For the bottom point special rules are used depending on the device type. For CICs the bottom point is placed on the major axis of the bottom opening and then moved as close as possible to the crus-ridge. If the bottom contour is non elliptical, the tragus line is used as a further constraint (Figure 5.17). For ITEs it is ensured that the bottom point is close to the inter-tragal

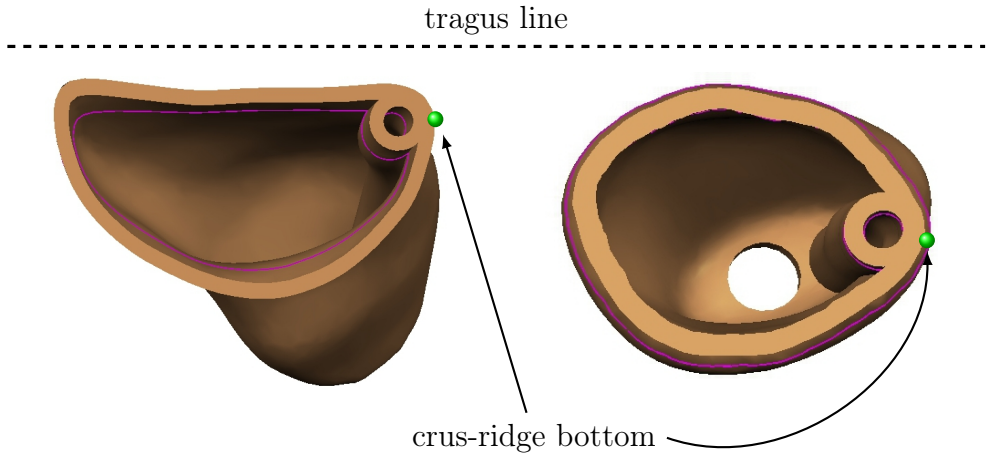


Figure 5.17: For CIC devices the bottom opening is analyzed using PCA. The vent bottom point is placed on the major opening axis closest to the crus-ridge bottom point. Since the opening contour is not always elliptical, the tragus line is used as an additional constraint.

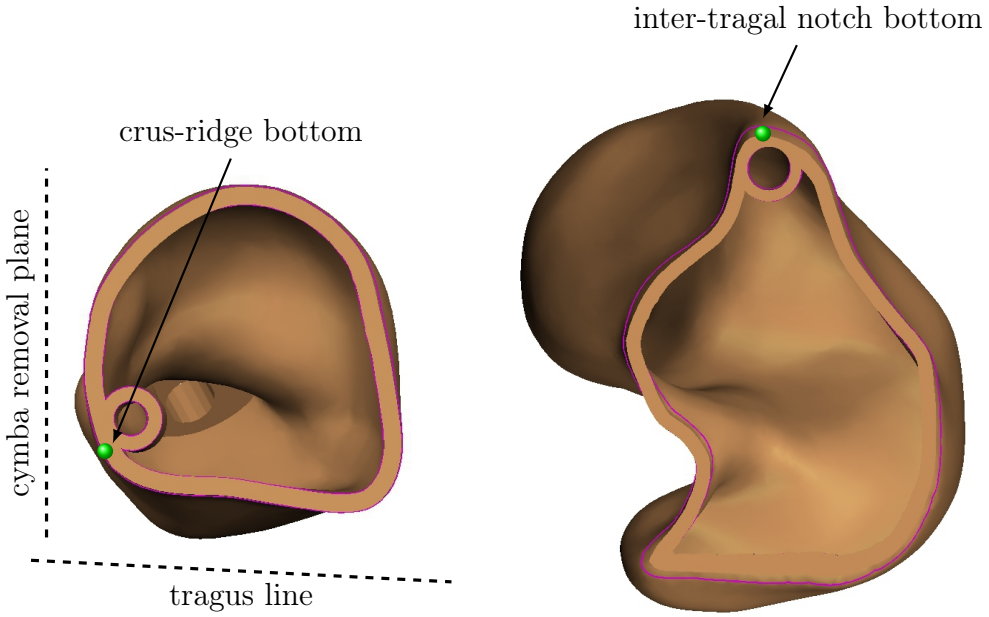


Figure 5.18: The vent bottom point for ITC devices is guided by the crus-ridge bottom point and constrained to be close to the tragus line and the cymba removal plane. The ITE vent bottom point is only constrained by the intersection of inter-tragal notch ridge and bottom contour.

notch ridge (Figure 5.18). Finally, for ITCs the bottom point is constrained by the *Remove or Reduce Cymba* plane and the so-called tragus line (Figure 5.18). The applied changes resulted in a perfect acceptance rate with a low user interaction rate. The average correction distance of 0.35 mm is below the expected differences of two users and indicate that the user interaction does not change the result significantly.

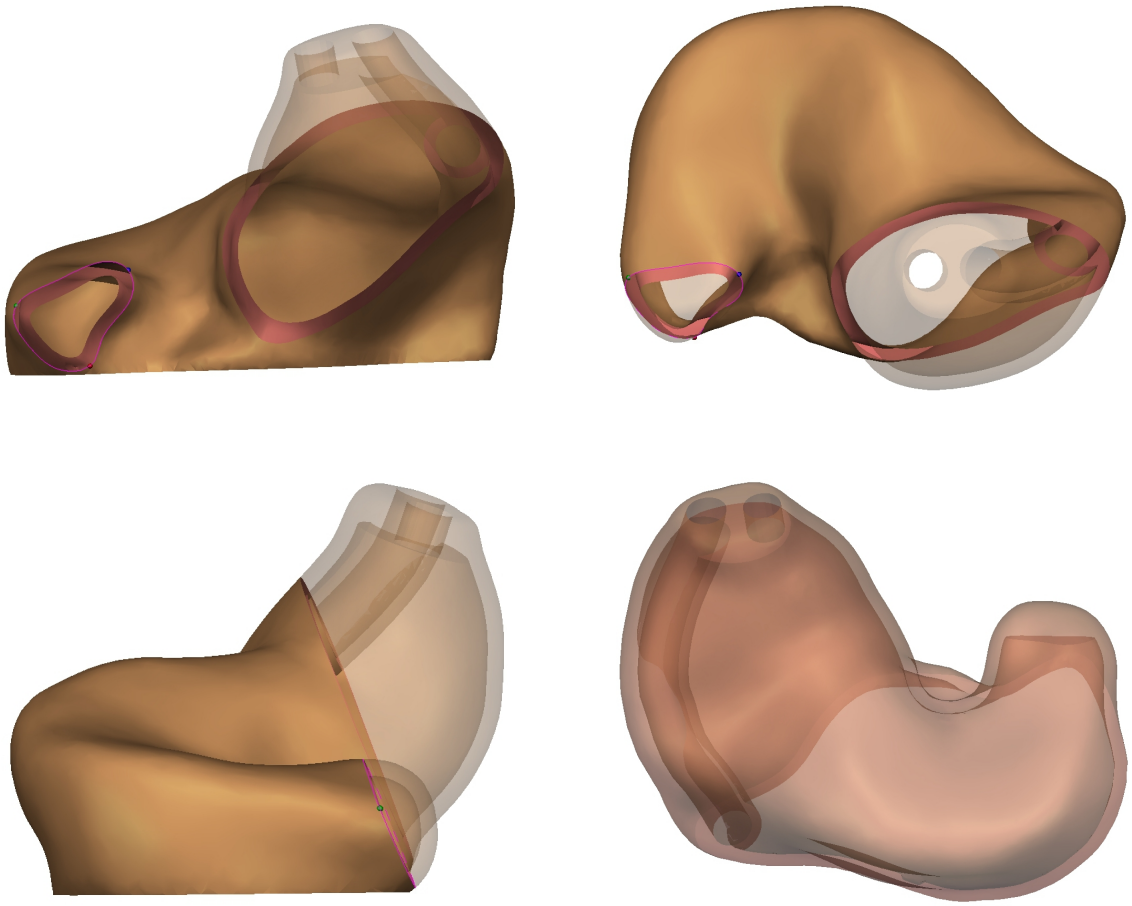


Figure 5.19: The first *Local Fill at Cymba Area* rule defines a plane which is used to fill the cymba peak with additional material. The transparent area indicates the defined plane. The plane usually intersects with the cymba and the ear canal. In order to select the correct area, a reference point is used, which is based on the helix-peak feature point. The picture on the lower right side shows the result of the rule application. The inner wall of the impression is enlarged by the selected area.

Improving the Cymba Filling

The results for the *Local Fill at Cymba Area* rule shown in Table 5.5 is the average of two successive rules. The first rule targets to fill the cymba (sometimes referred to as helix) peak (Figure 5.19). The second rule aims to fill the whole cymba area with material by offsetting the inner wall (Figure 5.20).

The workflow diagrammed in Figure 5.8 shows that the cymba filling rules are applied at the end of the design process. Therefore, all previous steps can influence the performance of these rules. Both rules were marginally changed in the different ESAM versions. Thus, the improvements in performance is due to the general improvements of the feature detection, the improvements applied to predecessor rules and the increased trust of the designers into the rule performance.

Filling the Cymba Peak The rule defines a plane, which shall ensure the filling of the cymba peak area as shown in Figure 5.19. The plane orientation $\boldsymbol{n}_{\text{sf}}$ is defined as a combination of the bottom-opening plane normal $\boldsymbol{n}_{\text{b}}$ and the crus-valley plane normal $\boldsymbol{n}_{\text{c}}$:

$$\begin{aligned} \boldsymbol{n}_{\text{dir}} &= \boldsymbol{n}_{\text{b}} \times \boldsymbol{n}_{\text{c}}, \\ \boldsymbol{n}_{\text{sf}} &= \boldsymbol{n}_{\text{b}} + 2 \cdot \boldsymbol{n}_{\text{dir}}. \end{aligned} \tag{5.6}$$

The location of the plane is defined by helix-peak point $\boldsymbol{p}_{\text{hp}}$ together with a shift along $\boldsymbol{n}_{\text{sf}}$:

$$\boldsymbol{p}_{\text{sf}} = \boldsymbol{p}_{\text{hp}} - 2.0 \cdot \boldsymbol{n}_{\text{sf}}. \tag{5.7}$$

The 2 mm shift is based on user observations and ensures that the whole cymba peak is filled, without intersecting with the bottom-opening plane. The performance values for the rule are given in Table 5.12.

Record property	ESAM1	ESAM2	ESAM3
Number of rule occurrences	469	402	3793
Number of complete rule occurrences	450	390	3758
Completeness rate δ_c	0.96	0.97	0.99
Acceptance rate δ_a	0.80	0.87	0.90
User interaction rate δ_u	0.29	0.17	0.07
Average normal angle error in degree	6.99	5.08	3.61
Average location error in mm	1.24	0.84	0.57

Table 5.12: Result of the RA for the first *Local Fill at Cymba Area* rule for the different ESAM versions.

Despite the fact that only minor modifications were applied to the rule, e. g. update of the reference point in order to ensure selection of the cymba area and not the canal area, the rule performance increased steadily for each ESAM version. Since the performance was already good in ESAM1, we assume that a strong impact on the performance measures is the increased trust of the user into the rule suggestion. Instead of modifying the selected plane marginally as done in ESAM1 for one third of the planes, the users reduce their changes to times in which the rule suggestion is failing.

Filling the Cymba The rule in ESAM3 equals the one of ESAM1, but the performance measures improved. On the one hand, the improvement is a result of the continuous improvements applied to ESAM, e. g. feature detection improvement. On the other hand, the system was trusted by the users to suggest the correct plane, and, therefore the tiny modifications applied in the first versions ceased. The development of the rule performance is given in Table 5.13.

The required plane shown in Figure 5.20 is computed as follows. The plane normal $\boldsymbol{n}_{\text{sf2}}$ is the average of the bottom-opening plane normal and the crus-valley plane normal

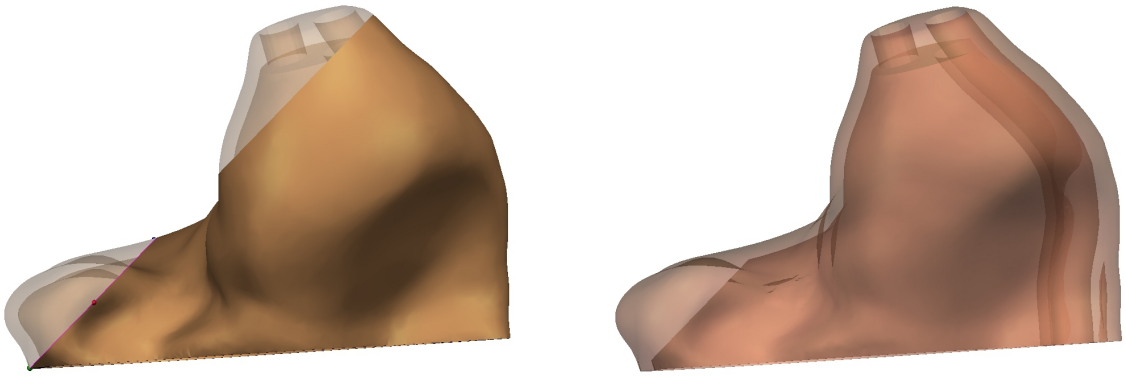


Figure 5.20: The *Local Fill at Cymba Area* rule defines a plane in the cymba reaching, which is used to fill the cymba with material. The left picture shows the defined plane crossing the cymba starting from the bottom contour and leaving the shell close to the crus concavity area. The plane is perpendicular to the viewing direction which correlates to the tragus line depicted in Figure 5.17. The root cause for the cymba filling is to provide the acoustician with the option to manually remove material in case the patient is uncomfortable with the aid in this area.

$$\boldsymbol{n}_{\text{sf2}} = \boldsymbol{n}_{\text{b}} + \boldsymbol{n}_{\text{c}}. \tag{5.8}$$

The plane location is defined by shooting the helix-peak point along the crus-valley plane normal away from the impression. Afterward, the shifted point is projected onto the bottom-opening plane $\boldsymbol{e}_{\text{b}}$. Finally, the closest point to the bottom opening contour $\mathcal{C}_{\text{b}}$ is identified.

$$\begin{aligned} \boldsymbol{p}_{\text{sf2}} &= \boldsymbol{p}_{\text{helix-peak}} + 20 \cdot \boldsymbol{n}_{\text{c}} \\ \boldsymbol{p}_{\text{sf2}} &= f_p\left(\boldsymbol{e}_{\text{b}}, \boldsymbol{p}_{\text{sf2}}\right) \\ \boldsymbol{p}_{\text{sf2}} &= f_c\left(\mathcal{C}_{\text{b}}, \boldsymbol{p}_{\text{sf2}}\right) \end{aligned} \tag{5.9}$$

In Eq. (5.9), function f_p projects the point given as second argument on the plane given as first argument. The second function f_c, identifies the closest point of the contour given a point as the second argument.

Contour point and normal define the filling plane. The last step is to move the plane about a small offset, e. g. 0.5 mm away from the bottom-opening plane. This ensures that the filling material is not connected to the faceplate in the final assembly. It could cause problems if it is necessary to remove the faceplate in a potential repair scenario.

Noteworthy is the decrease in orientation and location error of the suggested plane. On average, the suggested plane is almost perfect for ESAM3, because the error values are in a range expected by different users.

Record property	ESAM1	ESAM2	ESAM3
Number of rule occurrences	467	399	3777
Number of complete rule occurrences	456	390	3743
Completeness rate δ_c	0.98	0.98	0.99
Acceptance rate δ_a	0.75	0.74	0.89
User interaction rate δ_u	0.08	0.04	0.03
Average normal angle error in degree	10.45	9.51	4.17
Average location error in mm	3.95	1.58	0.67

Table 5.13: Result of the RA for the second *Local Fill at Cymba Area* rule for the different ESAM versions.

Rule	δ_c	δ_a	δ_u	performance
Rough Cut to Remove Excess Material	0.86	0.74	11.67	low
Taper Canal Tip	0.89	0.51	**1.52**	low
Remove or Reduce Cymba	0.95	0.74	**0.33**	medium
Local Offset at Tragus Region	0.97	**0.21**	0.48	low
Local Offset at Ear Canal	0.98	0.75	**0.20**	medium
Place Initial Vent Points	0.99	0.91	**1.11**	medium
Place Receiver Hole	0.99	0.81	3.54	low
Measured Shell Height Cut	**0.98**	**0.93**	**0.60**	high
Measured Shell Height Cut (ITC)	**0.98**	**0.86**	**0.20**	high
Optional Vent Cuts	**0.91**	0.86	**0.28**	high
Readjust Vent Points	0.97	**0.99**	**1.28**	medium
Local Fill at Cymba Area	**0.99**	**0.90**	**0.05**	high
Place Label Inside of Shell	0.76	0.36	1.22	low

Table 5.14: Performance of the ESAM3 system. Similar process (using similar rules) are grouped together. The values in bold font indicate performance improvements compared to ESAM1 (Table 5.5)

5.4 Conclusions

The previous sections showed that the semi-heuristic approach had a positive impact on the rule performance, while the GP method had only a limited use. To be fair, it must be noted that the data set $\mathcal{S}_1$ is smaller and had limited information compared to $\mathcal{S}_2$.

We showed that it was possible to improve the rule performance using a semi-automatic approach. The analysis of the data identified low performing rules and provided hints what needs to be improved, by exploring the relationships between rules. The manual implementation of the changes by the knowledge engineer accounts for the moving target situation. For example, new vents and design restrictions were introduced.

A disadvantages of the semi-heuristic learning is that the analysis is not detailed enough, due to the missing common coordinate system and feature detection results,

which renders drawing conclusions a complex and error prone task. It is possible to substitute the missing information by relationships and dependencies between features and rules, as shown in the previous discussions. Nevertheless, the extracted knowledge is based on assumptions and has to be analyzed with care. Furthermore, using the *Local Fill at Cymba Area* example it could be shown that the ESAM system in general improved with each version and the acceptance level improved accordingly.

Table 5.14 is the counterpart to Table 5.5 and shows in an overview the performance levels for all rule groups using ESAM3. On average, we could decrease the required amount of user interactions, which in turn increases the automation level of ESAM. Since there are still low performing rules left, there is still the need to make further improvements in order to achieve the goal of full automation.

Chapter 6

Expert System for Automatic Modeling

Finally, all parts discussed in the previous three chapters are wrapped together to the ESAM framework in a formal way. At first, an overview of the system architecture is given followed by a formal definition of the system. Subsequently, the development and deployment of the system is discussed. The evaluation section answers the question, if the goals defined in Chapter 1 were achieved and to what degree. In the end, ESAM is compared to similar works and systems, and conclusions are drawn. The ESAM framework was partly presented and discussed in [Sick 09, Sick 11a].

6.1 The ESAM Framework

As described in Chapter 2, the design of medical prostheses involves the modification of a given anatomical template (represented by a surface) by an expert designer, driven by a set of features. The designer thereby relies on rules, which are either available as work instructions or have been acquired by personal experience.

The design task can, therefore, be defined as that of a constrained shape transformation. Given an unprocessed shape (surface) S_{source}, represented as a triangulated mesh, the goal is to transform it into a target shape S_{target}, by a transformation $T_{\mathcal{R}}$. $T_{\mathcal{R}}$ is described as a sequence of transformations $T_{r_1}, T_{r_2}, \ldots$ defined by a set of rules $\mathcal{R} = \{r_1, \ldots, r_n\}$:

$$S_{\text{target}} = T_{\mathcal{R}}\left(S_{\text{source}}\right). \tag{6.1}$$

Typically, the transformation depends on constraints or options $\mathcal{O}$ (see Chapter 4), as well as anatomical features $\mathcal{F}$ (see Chapter 3) detected on the source shape. The features are utilized by the rules to determine the eventual transformation. Eq. (6.1) may, therefore, be extended to include this explicit dependence:

$$S_{\text{target}} = T_{\mathcal{R}}\left(S_{\text{source}}; \mathcal{F}, \mathcal{O}\right). \tag{6.2}$$

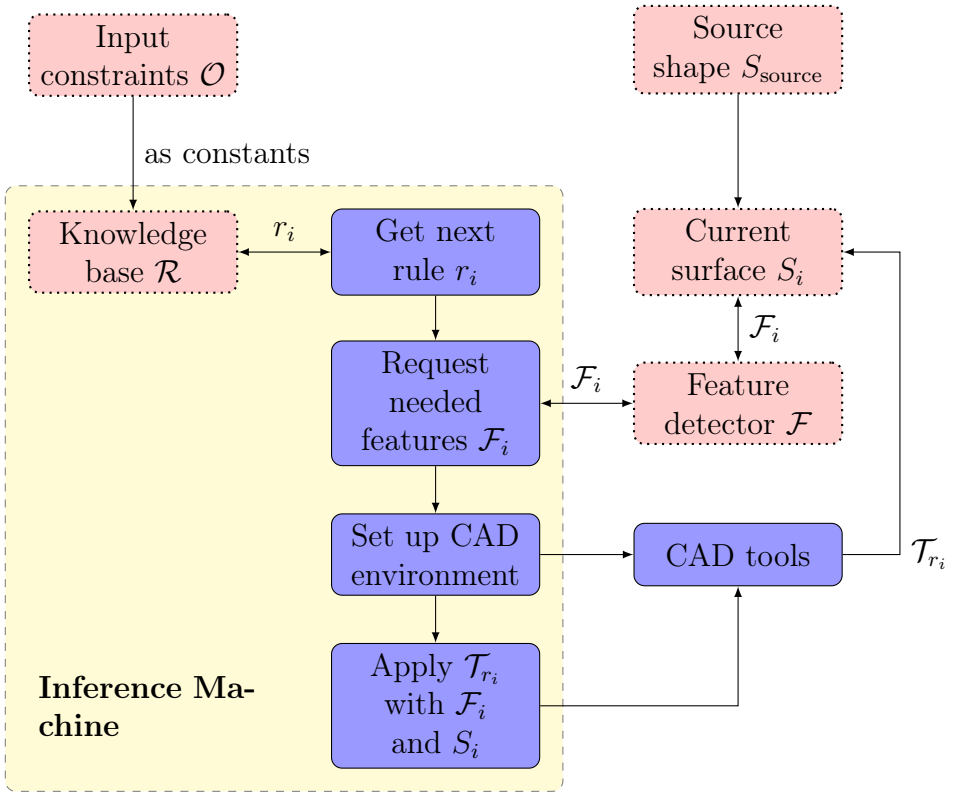


Figure 6.1: Workflow scheme of ESAM. For each prosthesis design task a set of constraints $\mathcal{O}$ and a source shape S_{source} is given as input. These inputs will be provided to the knowledge base $\mathcal{R}$ respectively to the feature detector $\mathcal{F}$. The inference machine executes all applicable rules, which yields application of CAD operations and consequently the modification of the source shape. During rule execution the feature detector updates the features if necessary. The blocks marked in blue are considered general, while the red (dotted) blocks are specific for the design application in mind.

Since the transform $T_{\mathcal{R}}$ is derived from rules, Eq. (6.2) can be rewritten as an application of a sequence of the pertinent individual rules:

$$\begin{aligned} S_0 &= S_{\text{source}}, \\ S_{i+1} &= T_{r_k}\left(S_i; \mathcal{F}_i, \mathcal{O}\right), \quad i = 0, \ldots, j-1; \quad r_k \in \mathcal{R}, \\ S_{\text{target}} &= S_j, \end{aligned} \tag{6.3}$$

where S_i denotes the source shape after the application of i rules. $\mathcal{F}_i$, in turn, denotes the features detected on the current surface S_i. Eqs. (6.2) and (6.3) form the foundation of the ESAM framework. They formalize the general automation problem, by including application specific data and constraints. A graphic representation of Eq. (6.3) is given in Figure 6.1.

6.2 ESAM Development and User Acceptance

6.2.1 ESAM Development

So far, the ESAM development included three major phases. Phase 1 – ESAM1 is the initial system containing the first complete knowledge base and the initial feature detection algorithms. During the following phases – ESAM2 & ESAM3 the KB, the feature detection and the CAD tools were improved continuously. The split in three phases is based on major KB updates. During these three phases not only the KB was modified, but also the design process, e. g. reordering of design steps, including new process steps and integrating new or updated CAD tools. These product based changes consequently required rule updates or new rules. Especially the reordering of design steps can influence the performance of a specific rule strongly, since it was designed with a different assumption in place. Thus, a simple comparison between the different phases is not possible nor reasonable. The moving target problem was already briefly discussed in the previous chapter.

6.2.2 Evolution of the Knowledge Base

Knowledge Base Size

In Table 6.1 the development of the KB is illustrated by the number of guide rules and lines of code. The numbers are generated from the dataset $\mathcal{S}_2$ introduced in Section 5.1.1.

ESAM version	# guide rules	# lines of code	# complete design records / %
ESAM1	43	2319	1724 / 36.96
ESAM2	49	2615	1565 / 28.32
ESAM3	53	3395	10832 / 66.63

Table 6.1: Development of the number of guide rules, the total number of lines of code and the ESAM-completion rate for all three major ESAM versions. In the last column only the number / percentage of completed records is given. The total number of design records were presented in Table 5.1.

Table 6.1 shows that the number of guide rules gradually increased for each version. However, due to the fact that the design process orders the rules linearly (see Figure 4.4) the KB can be managed well by a knowledge engineer. The modifications between KB2 and KB3 were more substantial then in the phase before yielding better completion rate. The strongly changing finish rate (ESAM-complete design records, see Definition 9) is discussed in Section 6.3.1.

Distribution of Guide Rule Types

In Figure 4.4 and the tables of the previous chapter, the rule types were defined as: point, plane, area and other. This characterization is based on the rule suggestion. In this section, we define a second characterization of the rules based on their automation level.

Definition 11. Rule types based on automation level.

- *Manual:* guide rules comprise two scenarios. One the one hand, there are the quality confirmation rules, which do not require any rule suggestion, but a visual inspection by the user. An example is the *Check Quality of Design* rule and its associated process step. The guide rule displays the designed shell aligned with the original impression and a color mapping, to provide the designer with a possibility to verify the design, see Figure 6.2. On the other hand, there are rules which require component placement and interactions. For example, after the *Place Receiver Hole* a specialized CAD function is called, which pushes the receiver as deep as possible into the ear canal. After this automatic placement the user has the possibility to manually optimize the position further. Hence, in such a case the rule is reduced to a function call including the set up of the CAD environment, which may be followed by manual adjustments. Finally, a couple of design steps are based on the cosmetic feeling of the designer. For example, the *Taper Canal Tip* is followed by a manual rule to provide the designer with the possibility to apply cosmetic modifications on the ear canal.

- *Semi-automatic:* Most of the guide rules are of the semi-automatic type meaning the rule suggestion works well in many cases, but requires confirmation.

- *AutoApply:* The *autoapply* rules are comprised of rules with very high performance. Thus, the user confirmation is skipped. Final goal of ESAM is to improve all semi-automatic rules until they can be considered as autoapply type rules.

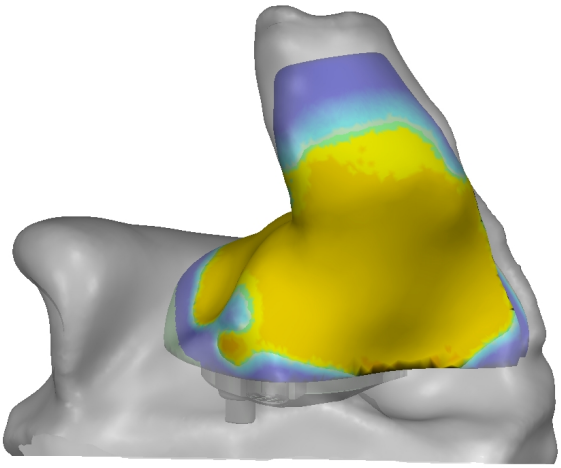


Figure 6.2: The manual guide rule *Check Quality of Design* displays the designed HA shell aligned with the original impression enhanced by a color mapping. The color mapping indicates if the designed shell has a positive (yellow/red) or negative (blue) offset compared to the original impression. This information is used by the designer to decide if the HA fits nicely in the ear canal without causing pain or being too loose.

The distribution of the three types is given in Table 6.2. In ESAM1 the proportion of manual and autoapply rules is alike and low (11 %). Unfortunately, in the next version, the number of manual rules increased. This was caused by the introduction of new design process steps, which lacked feature support at this point of time. Then again, the number of autoapply steps could be increased (14 %) and, thereby, increased the overall automation of ESAM.

ESAM version	# guide rules	# manual rules / %	# autoapply rules / %
ESAM1	43	5 / 11.63	5 / 11.63
ESAM2	49	8 / 16.23	7 / 14.28
ESAM3	53	6 / 11.32	7 / 13.32

Table 6.2: Development of manual and autoapply guide rules for the three major ESAM versions.

In the following version, ESAM3, we could reduce the number of manual rules again and kept the number of autoapply rules constant. This was made possible by enabling feature detection support for the crus area, which caused the rise of manual steps in the previous version.

The development of ESAM is still ongoing. In order to achieve full automation, it is mandatory to prevent the need for manual guide rules. So far, experience showed that it is possible to reduce the number of manual rules by adjusting KB and feature detection. However, the necessary process changes implied by supporting new designs or products may again increase the number of these rules. A further goal is to steadily increase the number of autoapply rules. Thereby, the system may achieve full automation in the future.

6.3 ESAM Verification

In the next sections and chapters, the term evaluation does not refer to a clinical evaluation with patients, but to a verification using expert designed measures and production results. A clinical verification was not possible in the scope of this work.

The general evaluation of ESAM is organized in five parts. First, we discuss the user acceptance of ESAM. The second part deals with the general performance of the guide rules. This is followed by a comparison of the design consistency for the manual and for the ESAM supported process. Fourth, the quality in terms of size is evaluated. Finally, the average time needed to design a customized HA is analyzed.

6.3.1 ESAM User Acceptance

In order to evaluate ESAM, the previously discussed dataset $\mathcal{S}_2$ is used. This time the evaluation is not focused on a single rule, but on the overall performance.

A major issue during the deployment was to convince the designers of the benefits of ESAM. In the beginning, ESAM was refused by a large number of users, because it requires additional training, it constrains the user to follow the process and it makes mistakes. In addition, a general resistance against changes and the fear to

be replaced by the system did reduce the acceptance rate further. However, after a training period we could convince more and more users to see the benefits of ESAM and to use it. This can be seen in Figure 6.3, which shows the increasing proportion of design tasks finished with ESAM3 compared to ESAM1. The increased completion rate indicates that the designers more and more accepted the system and used it to design the devices.

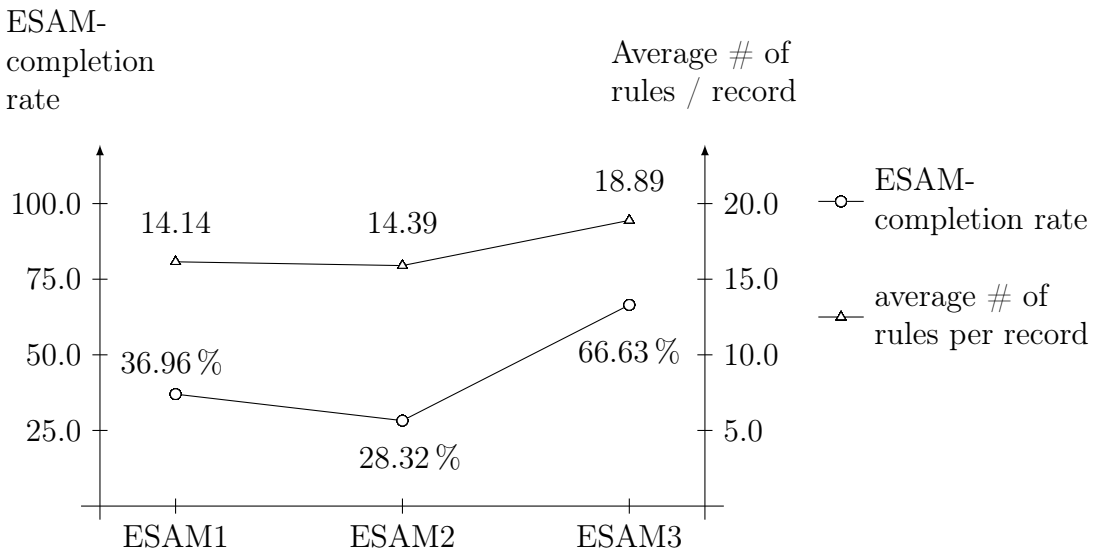


Figure 6.3: Visualization of the ESAM-completion rate (Definition 9) together with the average number of rules contained in a design record.

6.3.2 General Rule Performance

The final goal of ESAM is to automatically design medical prostheses, in our example case HA shells. To evaluate the automation level of ESAM, we analyzed the acceptance rate for all guide rules, independently of the completion status of the record. Furthermore, we divided the samples based on the target device type (CIC, ITC, ITE) to figure out, if the system performs better or worse for a certain device type. The results are shown in Table 6.3.

HA device type	ESAM1		ESAM2		ESAM3	
	δ_a	fraction %	δ_a	fraction	δ_a	fraction
ITE	0.8462	30.57	0.8134	30.45	0.7787	33.29
ITC	0.7929	49.60	0.7954	47.46	0.7866	43.62
CIC	0.7889	19.83	0.7832	22.09	0.8177	23.09

Table 6.3: Guide rule performance analysis for the different ESAM versions based on the acceptance rate δ_a.

The results in Table 6.3 show that the automation status achieved is good and similar for all device types. The modifications applied to the KB did not have a substantial

effect on δ_a. The performance for ITC devices is constant, for CIC devices it improved, but for ITE devices it dropped slightly. The drop is performance is also closely related to the increase in completion. In ESAM1, the designer often stopped using ESAM after coming across a low performing suggestion. In contrast to that, the designers correct and continue using ESAM in the later versions, which influences the acceptance rate. This is in accordance with the process changes applied during the KB evaluation, which mostly focused on ITEs and CICs. Analyzing the data further identified three kinds of behavior.

Approximately 85 % of the rules are stable in their performance level having a mean error and standard deviation below the thresholds defined in Definition 6. The remaining 15 % of the rules show a high mean error and/or standard deviation exceeding the thresholds. We could identify the following reasons, why the automation fails or is at least not good enough in these cases.

1. Poorly acquired impressions, with gaps, extreme amount of excess material, and noise, resulted in degradation of the performance of the feature detector. Since the entire system depends on these features, the overall performance dropped in such cases.

2. It is possible to misuse the framework by not following the process. If a step requires canal tip rounding, but the user applies rounding elsewhere on the surface, it can invalidate assumptions defined in the knowledge base.

3. The level of acceptability varies across users, as each user has his own understanding and cosmetic feeling.

4. An audiologist may specify special instructions. If they cannot be translated into the constraint set $\mathcal{O}$, the designer has to adjust the guide rule in order to fulfill these specific constraints.

Finish Rate

The user has the choice to ignore ESAM suggestions or disable ESAM support at any time during the process. Thus, we investigated the exit rule or in other words, which rule was performed last in a design record. In the ideal case this would always be the *Place Label Inside of Shell* rule. Analyzing the exit rules allows to draw conclusions about the typical reason for leaving the system. With this information, it is possible to directly approach the users and discuss the reasons for leaving the system so often during that particular rule.
Table 6.4 shows that in case of ESAM1 usage the system is mostly used from start to finish. The remaining 19.45 % are distributed evenly over the other rules without showing a significantly frequent other exit point. Thus, in ESAM1 the finish rate is good if the user starts to work with ESAM.

Comparing the different types of rules, shows that the manual rules are the most likely exit points. This could be confirmed during discussion with the designers. Their explanation was that in process steps which require manual interactions, they tend to fall back in the old behavior and keep on modeling manually.

Last applied rule	ESAM-valid (2114)	Total (4664)
Place Label Inside of Shell	81.55 %	36.96 %
Round Shell Around Faceplate Component (manual)	03.45 %	01.56 %
Faceplate Integration (manual)	01.56 %	00.71 %
Rough Cut to Remove Excess Material	01.56 %	00.71 %
Round Helix Peak	01.37 %	00.62 %

Table 6.4: Ranking of exit rules for ESAM1. Goal is to finish always with the *Place Label Inside of Shell* rule. The ESAM-valid column shows the exit rate for design records containing at least ten rule records (Definition 10). The last column shows the exit rate for all available design records.

Last applied rule	ESAM-valid (2350)	Total (5526)
Place Label Inside of Shell	66.59 %	28.32 %
Faceplate Integration (manual)	02.76 %	07.83 %
Receiver Collision Correction (manual)	02.76 %	01.18 %
Readjust Vent Points	02.17 %	00.92 %
Remove Bulbous Areas on Ear Canal (manual)	01.23 %	00.52 %

Table 6.5: Ranking of exit rules for ESAM2. The ESAM-valid column shows the exit rate for design records containing at least ten rule records (Definition 10). The last column shows the exit rate for all available design records.

Table 6.5 confirms the drop in user acceptance as already discussed in Section 6.3.1. As before, the manual rules are the most likely exit points. This becomes even more apparent than before. In the previous phase, only two of the most likely exit points were manual steps.

Last applied rule	ESAM-valid (15643)	Total (16257)
Place Label Inside of Shell	69.24 %	66.63 %
Rough Cut to Remove Excess Material	10.53 %	10.13 %
Define Quality Plane (manual)	04.49 %	04.32 %
Faceplate Integration (manual)	04.06 %	03.91 %
Receiver Collision Correction (manual)	02.20 %	02.12 %

Table 6.6: Ranking of exit rules for ESAM3. The ESAM-valid column shows the exit rate for design records containing at least ten rule records (Definition 10). The last column shows the exit rate for all available design records.

Finally, Table 6.6 lists the most likely exit points for ESAM3. The finish rate increased slightly compared to the previous version. The overall finish rate increased strongly showing a higher acceptance by the users. Thus, the modifications applied on the KB were appreciated by the users. Nevertheless, the high value for *Rough Cut to Remove Excess Material* rule shows that the user stop the usage of ESAM early in the process for 10 % of the design tasks. In other words, 10 % of the designers could

not be convinced to work with ESAM. The *Define Quality Plane* rule was introduced in ESAM3 and continues the tendency that manual rules are most likely to be an exit point.

6.3.3 Design Consistency

One of the major goals of ESAM stated in Chapter 1 was to improve the design quality and design consistency. To evaluate the consistency of the designs, we acquired two sample sets: $\mathcal{S}_{\text{Man}}$ and $\mathcal{S}_{\text{ESAM}}$.

1. Sample set $\mathcal{S}_{\text{Man}}$ consists of 30 manually designed HA shells. For each device type (CIC, ITC, ITE) ten shells were designed. Each shell was designed three times by the *same* user, resulting in 90 surfaces: $\mathcal{S}_{\text{Man}} = \{S_1, \ldots, S_{90}\}$.

2. Sample set $\mathcal{S}_{\text{ESAM}}$ consists of the same 30 impressions, but designed three times by different designers with ESAM.

As a preprocessing step, all corresponding surfaces were registered using the algorithm described in Section 3.3.1. The below described similarity measures were then computed for each corresponding group of ear shells.

Mesh coverage δ_{mc} is an indicator how well the shapes overlap. To compute δ_{mc}, the number of corresponding points whose distance is below a threshold is divided by the number of corresponding points. The measure is almost zero for shapes which are similar in size and shape, but do not match perfectly.

$$\delta_{\text{mc}}(S_i, S_j) = \frac{1}{N} \sum_{x \in S_i} f(\tau(x, S_j)), \tag{6.4}$$

The function $\tau(x, S)$ computes the distance between the closest point of surface mesh S and point x. f is a counting function, being 1 iff $0 \leq \tau(x, S) \leq \theta$ and 0 otherwise. θ is a threshold which is set to 2 mm in our experiments. Finally, N denotes the number of corresponding points, which may consist of all mesh points or a subset, e. g. 10 % of all points.

Normalized sum of squared error δ_{se} is similar to δ_{mc}, but instead of counting points, the squared error is accumulated and averaged. δ_{se} is equal to the definition given in Section 3.3.2, Eq. (3.26).

Median distance $\delta_{\tilde{m}}$ is defined as the median of the squared errors between the corresponding points of both surface meshes.

$$\delta_{\tilde{m}}(S_i, S_j) = \underset{x \in S_i}{\text{median}} \ \|\tau(x, S_j)\|_2 \tag{6.5}$$

Consistency Evaluation

Table 6.7 shows that using ESAM leads to superior consistency, even though $\mathcal{S}_{\text{Man}}$ were generated by the same user. Comparing both data sets, shows that all three error measures are reduced about approximately 50 %.

For CICs the improvement is less, but still approximately 30 %. The reason for this is the strong influence of the two manual process steps: *Remove Bulbous Areas on Ear Canal* and *Round Shell Around Faceplate Component*. The CICs are more affected by them, because they are smaller in general and the impression typically consist of a longer canal. The later requires more manual smoothing and therefore yields less consistent results.

The improvement in case of ITCs is huge. For δ_{se} the improvement is approximately 49 % and for δ_{mc} the improvement is approximately 59 %. This substantial progress in consistency is mainly due to a consistent placement of the hinge axis. Thus, the final faceplate position is much more consistent for ESAM modeled devices compared to manually modeled ones.

Similar to the ITCs, the ITEs modeled with ESAM are much more consistent compared to their manual twins. In case of ITEs, the application of the *Round Cymba Peak* and *Remove or Reduce Cymba* rules results in a very similar shape of the cymba region for all shells.

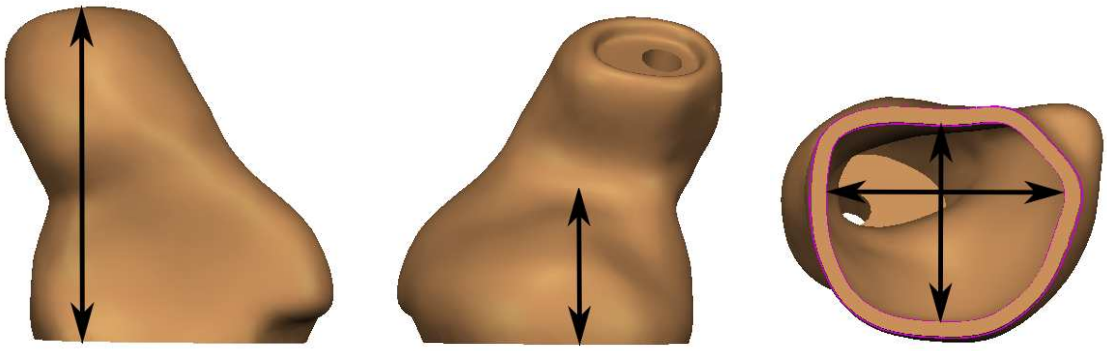
Sample set	δ_{mc}	δ_{se}	$\delta_{\tilde{m}}$	# samples
$\mathcal{S}_{\text{Man}}$	0.0097	0.2083	0.1247	90
$\mathcal{S}_{\text{Man-CIC}}$	0.0083	0.1670	0.1163	30
$\mathcal{S}_{\text{Man-ITC}}$	0.0128	0.2414	0.1399	30
$\mathcal{S}_{\text{Man-ITE}}$	0.0068	0.2147	0.1166	30
$\mathcal{S}_{\text{ESAM}}$	0.0043	0.1072	0.0663	90
$\mathcal{S}_{\text{ESAM-CIC}}$	0.0079	0.1146	0.0716	30
$\mathcal{S}_{\text{ESAM-ITC}}$	0.0053	0.1243	0.0834	30
$\mathcal{S}_{\text{ESAM-ITE}}$	0.0034	0.0868	0.0475	30

Table 6.7: Results of the consistency analysis. The set designed with ESAM shows superior (smaller) values than the manual design for all consistency measures.

6.3.4 Design Quality

It is difficult to objectively evaluate the quality of a hearing aid design. We simplified the problem, by ignoring the cosmetic appearance, and focusing on the size of the resulting device, since compactness forms a major design requirement for ITE HAs. We defined six size related measures:

- The *mesh size* is defined as the inner surface area of the designed shell.
- The *free volume* is related to the mesh size. It is defined as the empty space space inside the shell. In other words, everything which is not filled by the



(a) Shell height measured from *top* to *bottom*.

(b) Body height measured from *bottom* to the *low of aperture*.

(c) Bottom opening measured as sum of the length of the major axis of the bottom opening contour.

Figure 6.4: Visualization of the developed shell measures for design quality comparisons.

components is counted. Thus, the compactness of the component packing is measured.

- The *shell height* is measured as the perpendicular distance from the bottom opening to the farthest point (Figure 6.4(a)).
- The *body height* is measured as the perpendicular distance from the bottom opening to the lowest point on the aperture (Figure 6.4(b)). This measure is more significant than the shell height, since the canal length does not influence the visibility of the device.
- The *canal length* is the distance between the lowest point on the aperture and the highest point on the shell.
- The *bottom opening* is defined as the sum of two perpendicular lines at the bottom opening (Figure 6.4(c)). The perpendicular lines correspond to the major axis of the opening contour. This roughly measures how much space is available to fit the faceplate, see Figure 2.13(f).

Results of the shell measurements for the samples sets are shown in Table 6.8 and continued in Table 6.9. In general, the measurements are similar for both data sets, which is the expected result, because the modeling result is reviewed by an expert in both cases. Furthermore, the data sets are small and therefore have to be interpreted with care. For example, applying the t-test for the body height of CICs (4.87 mm and 3.29 mm) has no significance for $\alpha = 0.05$[1].

[1] For an independent two-sample t-test, the hypotheses are

$$H_0 : \mu_{\text{ESAM}} = \mu_{\text{Man}},$$
$$H_1 : \mu_{\text{ESAM}} \neq \mu_{\text{Man}}.$$

Sample set	Mesh size in mm^2	Free volume in mm^3	Shell height in mm
$\mathcal{S}_{\text{Man}}$	656.58	1114.35	15.55
$\mathcal{S}_{\text{Man-CIC}}$	513.96	709.28	15.05
$\mathcal{S}_{\text{Man-ITC}}$	551.20	823.81	15.92
$\mathcal{S}_{\text{Man-ITE}}$	904.59	1809.95	15.67
$\mathcal{S}_{\text{ESAM}}$	665.01	1165.32	16.01
$\mathcal{S}_{\text{ESAM-CIC}}$	502.12	681.73	14.71
$\mathcal{S}_{\text{ESAM-ITC}}$	509.78	733.96	16.06
$\mathcal{S}_{\text{ESAM-ITE}}$	972.26	2048.02	17.16

Table 6.8: Comparison of the ESAM and manually modeled shells. The shell measures are computed for all shells and for each device type.

Sample set	Body height in mm	Canal length in mm	Body opening in mm
$\mathcal{S}_{\text{Man}}$	8.06	11.10	27.25
$\mathcal{S}_{\text{Man-CIC}}$	4.87	11.99	19.94
$\mathcal{S}_{\text{Man-ITC}}$	9.33	10.17	25.32
$\mathcal{S}_{\text{Man-ITE}}$	9.98	11.16	36.29
$\mathcal{S}_{\text{ESAM}}$	7.67	12.22	27.49
$\mathcal{S}_{\text{ESAM-CIC}}$	3.29	13.67	19.48
$\mathcal{S}_{\text{ESAM-ITC}}$	8.76	11.08	25.94
$\mathcal{S}_{\text{ESAM-ITE}}$	10.67	12.01	36.36

Table 6.9: Comparison of the ESAM and manual modeled shells. The shell measures are computed for all shells and for each device type.

An advantage for the ESAM designed devices is the reduction in body height, because the instruments are less visible from the outside. This goes hand in hand with the increase in canal length and shell size, because more space in the canal area is needed to fit the components. As expected, the design quality is similar, because of the user verification. Nevertheless, there are some indications that the ESAM modeled shells are less visible and therefore more appealing for the patient.

6.3.5 Design Speed

To evaluate the design time, we made a time study together with our industry partner using 90 design tasks. The sample set comprises a high variation in ear impressions as

If $N = 30$ and $\alpha = 0.05$ then the t-statistic is

$$\|t\| = 0.7654$$

and the t-table [Bron 01] shows

$$t(\frac{1-\alpha}{2}, N + N - 2) \approx 2.0.$$

Hence, $\|t\| < t(0.975, 58)$ and the hypothesis H_0 is assumed to be valid with a confidence of 95 %.

well as in the selected options. Furthermore, the influence of monaural[2] or binaural modeling was taken into account. The results of the study are presented in Table 6.10.

Design mode	Manual	ESAM	Time reduction
Monaural	652.0 s	582.0 s	70.0 s
Binaural	990.0 s	615.0 s	375.0 s

Table 6.10: Design time study comparing the manual and the semi-automated design process based on 90 design tasks.

The results show a considerable time saving in case of ESAM usage. Especially, in case of binaural design tasks the ESAM system is far superior compared to the manual workflow. Here, the support for a synchronized workflow shows a potential reduction in time of 38 %. One drawback of the presented results so far is the small number of samples and a non consistent distribution of monaural and binaural cases. The time study for the manual designs contained 62 monaural and 28 binaural cases, while the other for ESAM contained 45 for each.

In order to verify the results, we extracted data from the production database from a period before introducing ESAM and compared it with a recently extracted data set (January 2011). The drawback in this case is that it is not possible to identify if the user was working binaural or not. Furthermore, given the results of Section 6.3.1 roughly 33 % of the current data set was designed manually. The results of the data extraction are given in Table 6.11.

	Design time	Time reduction	Sample size
Manual (old data)	705.1 s	-	8375
ESAM3	491.1 s	214 s	6832

Table 6.11: Design time for customized HA modeling with and without ESAM.

The time savings are even larger than measured in the previous time study. The average processing time is reduced about 30 %. However, it has to be taken into account that in the period between 2009 and 2011 the design process changed. Furthermore, the CAD software and the used hardware were updated. It is not possible to precisely extract the influence of these factors on the current processing time, but we assume that the larger portion of the time reduction is due to the usage of ESAM, which is supported by the results of the previous time study.

6.4 Comparison to Other Works

The field of rapid prototyping or CAD modeling in production including automation is a field of high interest and discussed, e. g. in [Gebh 03, Hopk 05]. The specific problem of medical prostheses design, especially customized hearing aid design automation, is

[2]Monaural modeling denotes the design of one HA shell and binaural modeling denotes the design of left and right HA shell simultaneously.

researched by Rasmus Paulsen et al., Greg Slabaugh, Alexander Zouhar et al. and us. With the latter, we discussed ideas and methods, but unfortunately worked with different data sources. We briefly introduce the methods proposed by these scientists and compare their results with ours.

6.4.1 Statistical Shape Models of the Human Ear Canal

Methods

Paulsen et al. published several works about statistical shape models for hearing aids including the dissertation of Paulsen [Paul 02, Paul 03, Paul 04b, Paul 04a, Dark 07]. Their work can be divided in two parts. First, they worked on the creation of a statistical shape model of the human ear canal. Thereby, they evaluated different methods to compute the shape correspondences including landmarks set by an expert, ICP based methods and Markov Random Field Regularization.

Second, the automated design of CIC HAs was researched. The research focused on the optimal placement of the faceplate in the aperture region of the ear canal. The developed objective function included a collision term, a term denoting the residual size of the faceplate, a term denoting the distance between faceplate and tympanic membrane, an alignment term expressing the ease of fitting the device into the ear. The terms are combined linearly with the help of weights chosen by trial and error. Their experiments have shown considerable variation between automatic and expert faceplate positioning. Nevertheless, the results were promising, but on a very limited sample set of eight impressions.

Another research topic resulted in the *one-size-fits-most* hearing aid shell. Therefore, a minimum shell was created based on the statistical shape model results and a sample set of 29 ear impressions. However, the fitting quality was not evaluated quantitatively, only the statement "*It turned out that a large group of people can use the minimum shell, if it is used in combination with a soft earplug.*" was given [Paul 04a].

Comparison

Paulsen et al. simplified the complexity of customized hearing aid design, by focusing on CIC instruments or in other words on the ear canal excluding the outer ear. In addition, the *minimum shell* CIC aims to be wearable by many patients instead of being customized for one patient. The beauty of the approach is the simplicity of the solution given a statistical shape model in comparison to our framework consisting of a feature detection unit and the knowledge base. Since no evaluation of the minimum shell CIC is given, we can only refer to our evaluation. ESAM3 showed an acceptance rate of 81.77 % for CIC devices (Table 6.3). Furthermore, due to the currently semi-automatic approach the missing 18 % are corrected by the user, thereby ensuring the fitting quality of the device. Another aspect neglected by Paulsen et al. is the venting and electric components problematic. Based on the hearing loss, different vent styles and sizes and different chips are suitable. Therefore, one single *minimum shell* model would not be sufficient in order to provide a solution for each patient. In our approach, the vents are position by the ESAM based on given option codes and the components are prepositioned.

The presented method for faceplate placement shows a good performance for the given sample set, but neglects the influence of vent position and size and is prone to have a tilted faceplate resulting in HA with low cosmetic quality. In ESAM the faceplate for CICs is prepositioned in the aperture region based on the *Measured Shell Height Cut* information. This ensures that the faceplate has the correct orientation and is not tilted. Furthermore, the subsequently called faceplate lowering function considers the vent and does not tilt the faceplate.

In the current state, ESAM is the superior approach to design HA automatically, respectively semi-automatically. ESAM is more flexible in the application scenarios and considers size and form components and vents. Furthermore, the evaluation of ESAM contains quantitative and qualitative results based on extensive data sets.

6.4.2 Generating Shapes by Analogies

Zouhar, Slabaugh and Unal published several works on the topic of hearing aid registration, shape matching and custom design automation [Unal 08, Zouh 06, Zouh 09, Zouh 10, Balo 10b, Balo 10a, Unal 11].

In their most recent work [Unal 11], the described automation framework is very similar in the application: repeatable, consistent automatic design of customized prosthetic devices, but with an opposite approach. They emphasize, that their shape registration and estimation approach are better suited to the applications, because they are less affected by noise than extracting features to device rules for a rule-based shape design.

Their approach is described by a two class relationship. One class is the undetailed raw ear impressions and the other the detailed ones. For the later steps, the shapes are converted to a voxelized signed distance function (SDF) representation using a 100^3 grid. At first all undetailed shapes have to be registered in order to be in a common coordinate system. As registration method an ICP based algorithm is used [Zouh 06], which actually utilizes features extracted from the impression surfaces. The detailed shapes were registered using a variational approach using the sum of squared distances as energy function. Next, shape regression was performed using a subset of the available modes identified using principle component analysis. Using as simplification the assumption that there is a linear behavior between the two classes resulting in the following equation:

$$\mathcal{P} \circ \boldsymbol{W}^{\boldsymbol{u}} = \boldsymbol{W}^{\boldsymbol{d}}, \tag{6.6}$$

where a row in $\boldsymbol{W}$ contains the shape information of one sample projected into a lower manifold using the principle components. $\boldsymbol{W}^{\boldsymbol{u}}$ denotes the undetailed shapes and $\boldsymbol{W}^{\boldsymbol{d}}$ the detailed ones. $\mathcal{P}$ describes the mapping between both classes chosen to be linear least squares optimization problem. The problem is enriched by using so called auxiliary shapes to ensure a good fit of the estimated detailed shape and the given ear geometry. The last step in the framework is to estimate a top and bottom cut. The former corresponds to the canal-taper plane described in previous chapters and the latter to the faceplate position. The planes are constructed by a collection of points which are likely to belong to a cutting region. A PCA results in the orientation of the plane which is further optimized by using the cut orientations of the according

detailed shape. The final stage is a morphing of the shape to the boundaries defined by the two planes.

Comparison

In comparison to our work, the presented frameworks by Slabaugh, Unal and Zouhar are limited and simplified. They exclude all dependencies to the electronics, the venting, neglect possible special design requests and restrict themselves to the most simple device class (CICs).

The presented results are detailed and convincing, but based on small test set, e. g. 34 shapes in [Unal 11]. Furthermore, the shown shapes indicate that bad scans and extreme shapes were not part of the testing database. They use the Dice measure[3] to express the quality of their results. The Dice measure shows a 85 % ± 5 % overlap with the ground truth (leave-one-out experiments). The average distance between deformed and ground truth impression is 0.43 mm ± 0.25 mm is small, but the maximum distance between the shapes is 3.6 mm ± 0.9 mm, which is rather large and would be not acceptable in a real world scenario.

The elegance of the approach is the immediate result and its sound mathematical basis. Since it lacks a number of required features in order to applied for customized HA design, a hybrid system as mentioned in [Unal 11] maybe a valid approach. In such a system, the shape estimation would be the starting point, which must be followed by manual adjusting the cutting planes to fit venting and components.

In terms of transparency, the the rule-based approach has advantages, because it is easy to comprehend and allows in-between adjustments if necessary. Furthermore, the biggest advantage of ESAM is the flexibility to be adjusted to new requirements, e. g. new vent shapes and special requests, and that it can be applied for all HA devices and not only CICs. In terms of reproducibility, the framework by Unal et al. would have an advantage, but only if the shape base would be kept constant, which is unlikely in a fast changing industry as HA manufacturing.

We think that a combination of our approach and shape estimation approach would be beneficial, be if for a design starting point or an enhanced feature mapping.

6.4.3 Specialized CAD / Modeling Software for Customized Design

In CAD-based HA design, two major software packages are available: 3Shape Shell Designer [3Sha 10] and Rapid Shell Modeling by Materialise [Mate 11]. Both software packages are similar and support the designer by providing virtual tools to model the device and to merge vents and suspension systems to fix electronic components. A limited automation support, usually based on templates, is available as well. This is implemented similar to ESAM, e. g. by entering a certain design step the appropriate

[3]The Dice measure between two shapes A and B represented in the voxel domain is defined as:

$$\delta_{\text{Dice}} = 2\frac{\#(A \cap B)}{\#A + \#B},$$

where # denotes the number of voxels on and inside of shape A.

tool is selected in very limited cases a suggestion is placed on the model. According to our experience, the suggested selection has a low quality, but helps the speeds up the user, because the correction is often faster than placing the tool from scratch.

A relatively new software package is the Cyfex Secret Ear Designer [Cyfe 11]. It heavily relies on templating and the algorithms used are similar to the works by Paulsen et al. and Unal et al. The approach works well, for so-called earmolds, which are even simpler than CICs, but fails for more complex devices.

In general, the competing CAD frameworks provide all similar modeling support in terms of tools and standard functionality. ESAM excels when it comes to automation and supporting repeatability and consistency of the designs.

6.5 Conclusions

In this chapter, we wrapped together the results of the previous three chapters, giving an overview of the complete architecture and a formal description. Furthermore, the deployment of the system in a real world manufacturing environment was discussed, including the problem of user acceptance. After a difficult beginning, ESAM could be successfully used as a design tool in HA manufacturing. The completion rate of ESAM1 was 36.96 %, which could be increased to 66.63 % for ESAM3.

The benefits of ESAM compared to the original manual process were shown by comparing consistency, quality and processing time with and without ESAM. The results generally favored ESAM as the superior approach. The consistency could be improved by at least 10 %. Design quality in terms of device size is comparable or slightly better than in the manual approach. The design time was decreased substantially. The average time saving is 214 s which equals a reduction of about 30 %.

The comparison to similar research works and specialized CAD packages showed that ESAM is currently the most complete and practical solution for design automation in the field of customized HA design. Other works usually focused on simplified problems and evaluated their results only on a small scale. Thus, it is not possible to judge, if the proposed methods are suitable for real world applications.

The presented framework is versatile in its architecture and can be used in a semi- and fully automatic mode to design customized medical prostheses. In order to apply ESAM for other prostheses as HAs, the application specific parts: knowledge base and feature detection need to be adjusted. The latter may, depending on the application in mind, require substantial effort. The developed KRL is able to address a wide range of CAD tasks without further adaptations.

Chapter 7

Conclusions

This chapter summarizes the main contributions and results of this work and describes future directions for research in the field of automatic design of customized medical prostheses. The summary is focused on our developed automation framework Expert System for Automatic Modeling (ESAM). It consists of an expert system shell, a knowledge base, a feature detection unit and CAD software. The framework was verified in a real world environment using the example of customized in-the-ear hearing aids.

7.1 Summary

In Chapter 1, the goal of this thesis was defined as the study of artificial intelligence methods, rule-based expert systems in particular, to provide an automatic solution for the design of customized medical prostheses. This goal was subdivided in three parts:

- development of a framework to automatically design customized prostheses,
- verification of the framework in a real world environment, and
- application of learning techniques to further improve the framework.

For the first part, we extensively studied rule-based expert systems and based on these studies we developed our own expert system shell and knowledge representation language. The expert system shell employs forward chaining for inferencing, because of its benefits for controlling and monitoring. The developed knowledge representation language, the *script*, provides a powerful and expressive language to encode the necessary expert knowledge in facts and rules. Nonetheless, the language was kept simple in order to facilitate the definition of rules by the domain experts involved. The expert system was combined with state-of-the-art feature detection algorithms and a CAD software, yielding the ESAM framework.

ESAM utilizes two types of feature detection methods. On the one hand, surface analyzing algorithms are used, which work on the instance level. On the other hand, registration of samples in combination with feature transfer is used to improve the robustness of the results. The former method involved the development of generic

algorithms functioning on mesh surfaces to identify interesting patterns and features. The labeling of these features is based on the hierarchical structure of the feature relationships. The second, registration based approach required the development of a robust alignment method. Our solution was to combine a rough registration based on the centerline of the ear canal, with a following fine registration using the surface. The iterative closest point algorithm was applied for both steps, once using the point-to-point and once using the point-to-plane metric. An intermediate, but nevertheless significant result of our registration based method was the definition of a representative set of ear impressions. So far, such a set has not been available in literature. We identified a representative set of ten impressions, which was acquired by clustering 473 unique ear impressions. The size of the representative set was identified by using agglomerative hierarchical clustering. We confirmed this result using expectation maximization clustering and Chinese restaurant clustering. To point out the dependency to the representative set, we refer to the second feature detection approach as clustering-based feature detection, while the former is referred to as surface-based feature detection. Comparing both methods showed that the clustering-based method is superior for most features. The mean deviation for point features is smaller by about 30 %, which translates to an average improvement of about 1.2 mm. For the plane features the results need to be improved for both methods. The surface-based method is superior concerning the detection of the ear bends, e. g. the second bend plane orientation error is more than doubled for the clustering based method. In case of the crus-valley plane both methods perform similar with a small advantage for the clustering based feature detection. The average orientation error is smaller about 27 %, while the locatiom error is smaller about 4 %.

The development of ESAM involved three major versions. Each version contained improvements in the knowledge base, the feature detection algorithms and the CAD software. For verification, ESAM was deployed to the manufacturing floor of a major hearing aid manufacturer. The successful employment of ESAM for this challenging use case confirms our hypothesis that AI methods are suitable for automatic design of customized prostheses. Each major version of ESAM was verified in the production environment. In order to ensure the quality of the designed and manufactured in-the-ear hearing aids, the system was used in a semi-automatic mode. This mode allows an operator to monitor and approve the rule suggestions and furthermore generates user interaction data, which can be analyzed afterward. The evaluation of more than 26000 hearing aid shell design records showed that after a learning and acceptance phase for the operators the usage of ESAM increased, yielding an ESAM-completion rate of 67 % for the current system, ESAM3. With an average rule acceptance rate of 76 % ESAM3 displayed a good performance. Nevertheless, improvements are still required for the fully automatic mode in order to support new use cases or very uncommon samples. For the evaluation, the design consistency as well as the design quality and the design time were measured. The comparison of the traditional manual method with the ESAM based approach showed significant benefits for the latter. The design consistency increased by approximately 10 % with regard to a similarity measure. Hence, repeatability of designs is better supported by ESAM. We developed six shell measures to compare the design quality between manual and ESAM supported workflow. The shell measures evaluate the design with

focus on size and visibility. There were no significant differences between the two approaches with respect to the shell measures. Hence, quality wise ESAM did not fall short either. This was expected, because in the semi-automatic mode the design process was still monitored by an expert modeler. Nevertheless, for the smaller device types the ESAM-designed results tend to be smaller. In regard to the design time, a reduction of about 30 % could be achieved, which is a significant improvement in this labor intensive business.

Concerning the third part, the application of learning techniques to further improve ESAM, two types of rule learning techniques were studied and evaluated in this work. The first one, genetic programming, was employed to evolve new rules based on training data. The evaluation of the evolved rules showed good performance on the test set, but failed to meet the expectations of the domain experts. Also, the exact rule extraction was difficult due to the high amount of bloat present in the genetic programming result. As a consequence, the genetic programming approach was replaced by a heuristic method, which is based on the data generated by ESAM during modeling. These files were processed to analyze the relationships between different rules as well as the intention and severity of user modifications during modeling. Furthermore, the general performance (acceptance and quality) of ESAM and its rules could be evaluated. The thereby gained knowledge was used to update the rules in the knowledge base and substantially improved the performance of ESAM.

7.2 Conclusions and Outlook

In this thesis we developed a versatile framework for the automatic design of customized medical prostheses. It was verified in a real world environment using the example of in-the-ear hearing aids and proved to be superior to the manual design approach. ESAM is currently used in a semi-automatic mode, because the individuality of the human body at times requires the correction of a rule suggestion to ensure the design quality. Even though the workflow in hearing aid manufacturing could not be fully automated, our work certainly changed the way how hearing aids are designed today. Operators are still needed for the validation of designs and design steps, but the system presented supports the inexperienced operators to achieve better results, eases the work for the expert designers and reduces the occurrence of design errors.

The automatic design of customized medical prostheses still is an area of high research interest. A topic regularly discussed in literature is the application of templating techniques or active shape models, which found their way into commercially available CAD packages. The templating techniques applied are similar to what we proposed in the clustering-based feature detection. Instead of features, though, it is the location of certain tools or components that is copied from the template to the current shape. We think that along with the representative set of ear impressions identified during our clustering study these ideas could be extended and should be integrated in future versions of ESAM.

We believe that ESAM is an intermediate step on the road to a fully automated system. Currently, manual validation of the results is still necessary and the maintenance of the knowledge base is mostly done by a knowledge engineer. We believe that

the next steps will be targeting the integration of templating techniques on the one hand, and the development of learning tools to maintain and update the knowledge base on the other hand. The combination of the solid basis we provided with ESAM and templating methods may yield a fully automatic design system in the future.

Appendix A

Knowledge Representation Language – *The Script*

In Chapter 4, the KLR is introduced and discussed as a mean of defining the design workflow for medical prostheses modeling. The KLR was realized the tools flex [1] and bison [2] [Donn 03]. Shortened examples of the used flex and bison files are given below to illustrate the implementation.

A.1 Lexical Analyzer

Please note that flex file is simplified to contain only the information necessary to identify the used tokens.

```
%option c++
%option yylineno
%option prefix="CZScript"
%option noyywrap

%{
#include "ScriptBlock.h"
#include "ScriptExpression.h"
#include "ScriptParser.h"
#include "ScriptInterpreter_yacc.hpp"

#define YY_INPUT(buf,result,max_size) \
        if ((result = LexerInput((char*)buf)) < 0) \
                YY_FATAL_ERROR("input in flex scanner failed");
%}

Digit           [0-9]
Letter          [_a-zA-Z]
DigitOrLetter   [0-9_a-zA-Z]
Number          0|([1-9]{Digit}*)
HexDigit        [0-9a-fA-F]
Exp                     [eE][+-]?{Number}
WhiteSpace      [ \t\r]

%e  2300
%p  100000
%n  1000

%%

'.'     {
        NextTokenLocation();
        yylval->ivalue = yytext[1];
        return T_CHAR;
};

{Number} {
```

[1] The Fast Lexical Analyzer, http://flex.sourceforge.net/, last visited 2011-01-06.
[2] GNU bison, http://www.gnu.org/software/bison/, last visited 2011-01-06.

```
        NextTokenLocation();
        yylval->ivalue = atol(yytext);
        return T_INT;
};

({Number}"."[0-9]*{Exp}?)|("."[0-9]+({Exp})?)| ({Number}{Exp}) {
        NextTokenLocation();
        yylval->dvalue = atof(yytext);
        return T_REAL;
};

\"[^\"\r\n]*[\r\n] {
        YYLTYPE loc;
        loc.first_line   = m_Location.first_line + 1;
        loc.last_line    = m_Location.first_line + 1;
        loc.first_column = m_Location.first_column - 1;
        loc.last_column  = m_Location.first_column - 1;
        m_pState->SetError(loc, _T("String constant is not closed."));
        NextTokenLocation();
        return T_EOL;
};

\"[^\"\r\n]*\" {
        int len = yyleng - 2;
        if (yyleng > 4097) {
                YYLTYPE loc;
                loc.first_line   = m_Location.first_line + 1;
                loc.last_line    = m_Location.first_line + 1;
                loc.first_column = m_Location.first_column - 1;
                loc.last_column  = m_Location.first_column - 1;
                m_pState->SetError(loc, _T("String constant truncated to 4095 characters"));
                len = 4095;
        }
        NextTokenLocation();
        // Convert to Unicode
        yytext[len + 1] = 0;
        MultiByteToWideChar(CP_UTF8, 0, yytext + 1, -1, yylval->svalue, len + 1);
        return T_STRING;
};

\"[^\"]* {
        YYLTYPE loc;
        loc.first_line   = m_Location.first_line + 1;
        loc.last_line    = m_Location.first_line + 1;
        loc.first_column = m_Location.first_column - 1;
        loc.last_column  = m_Location.first_column - 1;
        m_pState->SetError(loc, _T("String constant is not closed"));
        NextTokenLocation();
};

"<=" { NextTokenLocation(); return T_LE; }
">=" { NextTokenLocation(); return T_GE; }
"==" { NextTokenLocation(); return T_EQ; }
"<>" { NextTokenLocation(); return T_NE; }

{Letter}({DigitOrLetter}){0,4094} {
        NextTokenLocation();
        // check keyword
        int iKeyword = m_pState->GetKeyword(yytext);
        if (iKeyword != -1)
                return iKeyword;
        // Convert to Unicode
        MultiByteToWideChar(CP_UTF8, 0, yytext, -1, yylval->svalue, yyleng + 1);
        const ShellFeatureInfoStruct* pFeatureInfo = m_pState->GetFeatureInfo(yylval->svalue);
        if (pFeatureInfo != 0) {
                yylval->ivalue = pFeatureInfo->nFeatureID;
                switch (pFeatureInfo->nFeatureType) {
                case FEATURE_TYP_POINT: return T_FEATURE_POINT;
                case FEATURE_TYP_AREA:  return T_FEATURE_AREA;
                case FEATURE_TYP_AREACOLLECTION: return T_FEATURE_AREACOLLECTION;
                case FEATURE_TYP_PLANE: return T_FEATURE_PLANE;
                case FEATURE_TYP_SCALAR: return T_FEATURE_SCALAR;
                case FEATURE_TYP_CURVE: return T_FEATURE_CURVE;
                default:
                        _ASSERT(false);
                }
        }
        const std::string* pParameter = m_pState->GetParameter(yytext);
        if (pParameter != 0) {
                MultiByteToWideChar(CP_UTF8, 0, pParameter->c_str(),-1, yylval->svalue, pParameter
                    ->size() + 1);
                return T_PARAMETER;
        }

        const CZScriptSymbol *pSymbol = m_pState->GetSymbol(yylval->svalue, yytext);
        if (pSymbol != 0) {
                yylval->sym = pSymbol;
                if (pSymbol->IsFunction())
                        return T_FUNCTION;
                return T_VARIABLE;
        }

        return T_NEWSYMBOL;
};
```

```
(#.*)?\r?\n {
        NextTokenLocation();
        // comment to the end of the line
        return T_EOL;
};

{WhiteSpace}    { NextTokenLocation(); }
/* eat the whitespaces which were not used until now */

[\+\-\*/\<\>\=\(\)\[\]\.\,:]    { NextTokenLocation(); return yylval->ivalue = *yytext; };

.                               {
        YYLTYPE loc;
        loc.first_line   = m_Location.first_line + 1;
        loc.last_line    = m_Location.first_line + 1;
        loc.first_column = m_Location.first_column - 1;
        loc.last_column  = m_Location.first_column - 1;
        m_pState->SetError(loc, _T("Ignoring invalid character: \"%c\""), *yytext);
        NextTokenLocation();
        yylval->ivalue = *yytext;
};

%%
```

A.2 Parser

The bison parser file below is shortened to avoid lengthen of the thesis unnecessarily, but still contains the information necessary to identify how the parser works in conjunction with the tokens provided by the flex file.

```
%pure-parser
%error-verbose
%verbose
%name-prefix="ScriptInterpreter_"
%locations
%parse-param { CZScriptParser &parser }
%parse-param { CZScriptParserState &state }
%lex-param   { CZScriptParser &parser }

%{
#include "ScriptParser.h"
#include "ScriptParserState.h"
#include "ScriptBlock.h"
#include "ScriptExpression.h"
#include "ScriptFunction.h"
#include "ScriptStatement.h"
%}

%union {
  int        ivalue;
  int        newVar;
  double     dvalue;
  wchar_t    svalue[SCRIPT_MAXSYMBOLLEN];
  CZScriptBlock          *block;
  const CZScriptSymbol   *sym;
  CZScriptParserExpression expr;
}

%token T_IF T_THEN T_ELSIF T_ELSE T_ENDIF
%token T_WHILE T_ENDWHILE T_DO
%token T_FOR T_FROM T_TO T_DOWNTO T_STEP T_DO T_ENDFOR
%token T_REPEAT T_UNTIL
%token T_RETURN T_EXIT T_BREAK
%token T_OPOPEN T_OPEXEC T_OPERATION
%token T_CONST T_REF T_STATIC
%token T_BOOLTYPE T_CHARTYPE T_INTTYPE T_REALTYPE T_STRINGTYPE T_POINTTYPE T_VECTORTYPE
    T_PLANETYPE T_MATRIXTYPE
%token T_FORWARD T_PROCDEF T_PROCEND T_FUNCDEF T_FUNCEND
%token T_UMINUS '+' '-' '*' '/' T_INTDIV T_MOD T_DEFINED T_FEATURESHOW T_FEATUREHIDE
%token T_EQ T_NE '>' '<' T_GE T_LE
%token T_NOT T_OR T_XOR T_AND
%token '=' '(' ')' ',' '[' ']'

%token <ivalue> T_TRUE T_FALSE T_CHAR T_INT T_EOL
%token <dvalue> T_REAL
%token <svalue> T_STRING T_NEWSYMBOL T_PARAMETER
%token <sym>    T_FUNCTION T_VARIABLE
%token <ivalue> T_FEATURE_POINT T_FEATURE_AREA T_FEATURE_AREACOLLECTION T_FEATURE_PLANE
    T_FEATURE_SCALAR T_FEATURE_CURVE

/// lowest precedence
%right T_ELSE
%nonassoc '='
```

```
%left T_OR
%left T_XOR
%left T_AND
%left T_EQ T_NE '<' '>' T_LE T_GE
%left T_BOR
%left T_BXOR
%left T_BAND
%left '+' '-'
%left '*' '/' T_INTDIV T_MOD
%nonassoc T_NOT T_BNOT T_UMINUS
%left '.'
%right '['

%type <svalue> symname funcdefopen procdefopen statementinvalid
%type <svalue> notallowed_else notallowed_endwhile notallowed_endfor notallowed_until
%type <svalue> notallowed_procend notallowed_funcend notallowed_opexec
%type <svalue> statement_mainerr statement_iferr statement_forerr statement_whileerr
    statement_repeaterr
%type <svalue> statement_operationerr statement_procdeferr statement_funcdeferr
%type <expr>   expr lval exprlist exprlistnamed funcall proccall for_expr_to for_expr_do
%type <sym>            function procedure
%type <ivalue> feature vardeftype argmodifier argmodifiers opmodifiers

%start program
%%
program: statementlist_main ;
statementlist_main: statement_main | statementlist_main statement_main;
statementlist_if: statement_if | statementlist_if statement_if;
statementlist_for: statement_for | statementlist_for statement_for;
statementlist_while: statement_while | statementlist_while statement_while;
statementlist_repeat: statement_repeat | statementlist_repeat statement_repeat;
statementlist_operation: statement_operation | statementlist_operation statement_operation;
statementlist_procdef: statement_procdef | statementlist_procdef statement_procdef;
statementlist_funcdef: statement_funcdef | statementlist_funcdef statement_funcdef;
statement_main:
        statement
|       statement_mainerr error T_EOL {
                state.SetError(@1, _T("Main block: %s"), $1);
        }
;
statement_if:
        statement
|       statement_iferr error T_EOL {
                state.SetError(@1, _T("IF block: %s"), $1);
        }
;
statement_for:
        statement
|       statement_forerr error T_EOL {
                state.SetError(@1, _T("FOR block: %s"), $1);
        }
;
statement_while:
        statement
|       statement_whileerr error T_EOL {
                state.SetError(@1, _T("WHILE block: %s"), $1);
        }
;
statement_repeat:
        statement
|       statement_repeaterr error T_EOL {
                state.SetError(@1, _T("REPEAT block: %s"), $1);
        }
;
statement_operation:
        statement
|       statement_operationerr error T_EOL {
                state.SetError(@1, _T("Operation block: %s"), $1);
        }
;
statement_procdef:
        statement
|       statement_procdeferr error T_EOL {
                state.SetError(@1, _T("PROCEDURE definition block: %s"), $1);
        }
;
statement_funcdef:
        statement
|       statement_funcdeferr error T_EOL {
                state.SetError(@1, _T("FUNCTION definition block: %s"), $1);
        }
;
statement:
        T_EOL
|       conditional
|       opblock
|       varassignment
|       vardefinition
|       T_RETURN          T_EOL { AllocReturnEmpty(state, @1); }
|       T_RETURN expr T_EOL { AllocReturnExpression(state, $2, @2); }
|       T_BREAK       T_EOL { AllocBreak(state, @1); }
|       T_EXIT            T_EOL { AllocExit(state, @1); }
|       proccall {
                // This is a dummy assignment to empty symbol. Good for procedure or function
                    calls.
```

```
                AllocEmptyAssignment(state, $1, @1);
        }
|       funcdefinition
|       statementinvalid error T_EOL {
                state.SetError(@1, _T("Invalid expression starting with \"%s\""), $1);
        }
;
conditional:
        if_elsif_else_endif
|       for_do_endfor
|       while_do_endwhile
|       repeat_until
;
if_elsif_else_endif:
        condition_if    statementlist_if        elsifs {
                state.PopBlock();
        }
|       condition_if                                    elsifs {
                state.PopBlock();
        }
;
condition_if:
        T_IF expr T_THEN {
                CZScriptStatementBranch *pBranch        = new CZScriptStatementBranch();
                CZScriptBlock                   *pBlock         = state.PushNode(new CZScriptBlock
                    );
                state.PushStatement(pBranch);
                state.PushBlock(pBlock, pBranch, _T("IF"), @1, false);
                if ($2.GetType() != AMD_BOOL8)
                        state.SetError(@2, _T("Boolean condition expected"));
                else
                        pBranch->PushExpression($2.GetExpression(), @2.first_line);
                pBranch->PushBlock(pBlock);
        }
|       T_IF error T_EOL {
                CZScriptStatementBranch *pBranch        = new CZScriptStatementBranch();
                CZScriptBlock                   *pBlock         = state.PushNode(new CZScriptBlock
                    );
                state.PushStatement(pBranch);
                state.PushBlock(pBlock, pBranch, _T("IF"), @1, false);
                pBranch->PushBlock(pBlock);
                state.ExplainError(_T("IF block opening: invalid expression"));
        }
;
condition_elsif:
        T_ELSIF expr T_THEN     {
                if ($2.GetType() != AMD_BOOL8) {
                        state.SetError(@2, _T("Boolean condition expected"));
                } else {
                        CZScriptStatementBranch *pBranch        = dynamic_cast<
                            CZScriptStatementBranch*>(state.GetBlockParent());
                        CZScriptBlock                   *pBlock         = state.PushNode(new
                            CZScriptBlock);
                        state.PopBlock();
                        state.PushBlock(pBlock, pBranch, _T("ELSIF"), @1, false);
                        pBranch->PushExpression($2.GetExpression(), @2.first_line);
                        pBranch->PushBlock(pBlock);
                }
        }
|       T_ELSIF error T_EOL {
                state.ExplainError(_T("ELSIF block opening: invalid expression"));
        }
;
condition_else:
        T_ELSE {
                CZScriptStatementBranch *pBranch        = dynamic_cast<CZScriptStatementBranch*>(
                    state.GetBlockParent());
                CZScriptBlock                   *pBlock         = state.PushNode(new CZScriptBlock
                    );
                state.PopBlock();
                state.PushBlock(pBlock, pBranch, _T("ELSE"), @1, false);
                pBranch->PushBlock(pBlock);
        }
;
elsifs:
                                                                                        else
|       condition_elsif          statementlist_if      else
|       condition_elsif                                                          else
|       condition_elsif elsif statementlist_if  else
|       condition_elsif elsif                                    else
;
else:
                                                                                T_ENDIF
|       condition_else  statementlist_if       T_ENDIF
|       condition_else                                   T_ENDIF
;
elsif:
                statementlist_if      condition_elsif
|                                               condition_elsif
|       elsif statementlist_if  condition_elsif
|       elsif                                   condition_elsif
;
// for, while, repeat are similar
opblock:
        opopen1 T_OPEXEC '(' symname ',' exprlist ')' T_EOL {
```

```
                OperationClose(state, $4, @4);
        }
|       opopen1 T_OPEXEC '(' symname ')' T_EOL {
                OperationClose(state, $4, @4);
        }
|       opopen1 statementlist_operation T_OPEXEC '(' symname ',' exprlist ')' T_EOL {
                OperationClose(state, $5, @5);
        }
|       opopen1 statementlist_operation T_OPEXEC '(' symname ')' T_EOL {
                OperationClose(state, $5, @5);
        }
|       opopen2 symname ',' exprlist ')' T_EOL {
                OperationClose(state, $2, @2);
        }
|       opopen2 symname ')' T_EOL {
                OperationClose(state, $2, @2);
        }
|       opopen1 error ')' T_EOL {
                state.PopBlock();
                state.PopExpressionList();
                state.SetInOperation(false);
                state.ExplainError(_T("OpFinalize"));
        }
|       opopen2 error ')' T_EOL {
                state.PopBlock();
                state.PopExpressionList();
                state.SetInOperation(false);
                state.ExplainError(_T("OpFinalize"));
        }
;
opopen1:
        T_OPOPEN '(' symname ',' expr ',' opmodifiers ')' T_EOL {
                if (state.IsInOperation())
                        state.SetError(@1, @8, _T("Already inside operation. Nested operations not
                                allowed."));
                if (state.GetFunctionFrame() != 0)
                        state.SetError(@1, @8, _T("Operations inside functions not allowed."));
                OperationOpen(state, @1, $3, @3, $5, @5, $7, @7);
        }
|   T_OPOPEN error T_EOL {
                // push something to be popped by the upper level error handler
                state.PushBlock(state.PushNode(new CZScriptBlock), 0, _T("Operation"), @1, false);
                state.PushExpressionList();
                state.ExplainError(_T("Operation block opening"));
        }
;
opopen2:
        T_OPERATION '(' symname ',' expr ',' opmodifiers ',' {
                if (state.IsInOperation())
                        state.SetError(@1, @8, _T("Already inside operation. Nested operations not
                                allowed."));
                if (state.GetFunctionFrame() != 0)
                        state.SetError(@1, @8, _T("Operations inside functions not allowed."));
                OperationOpen(state, @1, $3, @3, $5, @5, $7, @7);
        }
|   T_OPERATION error ',' {
                // push something to be popped by the upper level error handler
                state.PushBlock(state.PushNode(new CZScriptBlock), 0, _T("Operation"), @1, false);
                state.PushExpressionList();
                state.ExplainError(_T("Operation command"));
        }
;
opmodifiers:
        symname { $$ = OperationModifier(state, $1, @1); }
|       opmodifiers '+' symname { $$ = $1 | OperationModifier(state, $3, @3); }
;
vardefinition:
        vardeftype vardeflist T_EOL {
                state.SetLastVarDefModifiers(0);
        }
|       vardefmods vardeftype vardeflist T_EOL {
                state.SetLastVarDefModifiers(0);
        }
|       vardeftype error T_EOL {
                state.ExplainError(_T("Variable definition"));
                state.SetLastVarDefModifiers(0);
        }
|       vardefmods error T_EOL {
                state.ExplainError(_T("Variable definition"));
                state.SetLastVarDefModifiers(0);
        }
;
vardeftype:
        T_BOOLTYPE              { $$ = state.SetLastVarDefType(AMD_BOOL8); }
|       T_CHARTYPE              { $$ = state.SetLastVarDefType(AMD_CHAR); }
|       T_INTTYPE               { $$ = state.SetLastVarDefType(AMD_SINT32); }
|       T_REALTYPE              { $$ = state.SetLastVarDefType(AMD_FLT32); }
|       T_STRINGTYPE    { $$ = state.SetLastVarDefType(AMD_STDSTRING); }
|       T_POINTTYPE             { $$ = state.SetLastVarDefType(AMD_VEC3F); }
|       T_VECTORTYPE    { $$ = state.SetLastVarDefType(AMD_VEC3F); }
|       T_PLANETYPE             { $$ = state.SetLastVarDefType(AMD_PLANEF); }
|       T_MATRIXTYPE    { $$ = state.SetLastVarDefType(AMD_MATRIX4X4F); }
;
vardefmods:
        vardefmodifier
```

```
|       vardefmods vardefmodifier
;
vardefmodifier:
        T_CONST                 { state.SetLastVarDefModifiers(state.GetLastVarDefModifiers() |
            SYMBOL_CONST); }
|       T_STATIC                { state.SetLastVarDefModifiers(state.GetLastVarDefModifiers() |
    SYMBOL_STATIC); }
;
vardeflist:
        vardef
|       vardeflist ',' vardef
;
vardef:
        symname {
                if ((state.GetLastVarDefModifiers() & SYMBOL_CONST) != 0)
                        state.SetError(@1, _T("Constant must have a value assigned."));
                else
                        state.AddSymbolExplicit($1, @1, state.GetLastVarDefType(), state.
                            GetLastVarDefModifiers());
        }
|       symname '=' expr {
                const CZScriptSymbol *pSymbol = state.AddSymbolExplicit($1, @1, state.
                    GetLastVarDefType(), state.GetLastVarDefModifiers());
                if (pSymbol != 0) {
                        CZScriptParserExpression expr;
                        AllocExpressionVariableRef(state, pSymbol, @1, expr);
                        AllocAssignment(state, expr, @1, $3, @3, false);
                }
        }
|       symname '[' expr ']' {
                if ((state.GetLastVarDefModifiers() & SYMBOL_CONST) != 0)
                        state.SetError(@1, _T("Constant must have a value assigned."));
                else {
                        const CZScriptSymbol *pSymbol = state.AddSymbolExplicit($1, @1,
                                state.GetLastVarDefType(), state.GetLastVarDefModifiers() |
                                    SYMBOL_VECTOR1);
                        if (pSymbol != 0)
                                AllocArrayResize(state, pSymbol, $3, @3);
                }
        }
|       symname '[' ']' {
                if ((state.GetLastVarDefModifiers() & SYMBOL_CONST) != 0)
                        state.SetError(@1, _T("Constant must have a value assigned."));
                else
                        state.AddSymbolExplicit($1, @1, state.GetLastVarDefType(), state.
                            GetLastVarDefModifiers() | SYMBOL_VECTOR1);
        }
|       symname '[' ']' '=' expr {
                const CZScriptSymbol *pSymbol = state.AddSymbolExplicit($1, @1,
                        state.GetLastVarDefType(), state.GetLastVarDefModifiers() | SYMBOL_VECTOR1
                            );
                if (pSymbol != 0) {
                        CZScriptParserExpression expr;
                        AllocExpressionVariableRef(state, pSymbol, @1, expr);
                        AllocAssignment(state, expr, @1, $5, @5, false);
                }
        }
;
// functions and procedures removed
varassignment:
        T_NEWSYMBOL     '=' expr T_EOL {
                const CZScriptSymbol *pSymbol = state.AddSymbolImplicit($1, $3.GetType(), 0);
                CZScriptParserExpression expr;
                AllocExpressionVariableRef(state, pSymbol, @1, expr);
                AllocAssignment(state, expr, @1, $3, @3, true);
        }
|       lval            '=' expr T_EOL {
                AllocAssignment(state, $1, @1, $3, @3, true);
        }
|       T_NEWSYMBOL     '=' error T_EOL {
                state.ExplainError(_T("Variable Assignment: invalid expression"));
        }
|       lval            '=' error T_EOL {
                state.ExplainError(_T("Variable Assignment: invalid expression"));
        }
;
lval:
        T_VARIABLE                                                      {
                if (! AllocExpressionVariableRef(state, $1, @1, $$))
                        YYERROR;
        }
|       lval    '[' expr ']'                            { AllocExpressionArrayItemRef(state, $1,
    @1, $3, @3, $$); }
|       lval    '[' expr ',' expr ']'       { AllocExpressionArrayItemDim2(state, $1, @1, $3, @3,
    $5, @5, true, $$); }
;
expr:
    '(' expr ')'        { $$ = $2; }
|       T_TRUE              { $$.Set(AMD_BOOL8,  state.PushNode(new CZScriptExpressionConst<
    BOOL8>(true)),  SYMBOL_CONST); }
|       T_FALSE             { $$.Set(AMD_BOOL8,  state.PushNode(new CZScriptExpressionConst<
    BOOL8>(false)), SYMBOL_CONST); }
|       T_CHAR              { $$.Set(AMD_SINT8,  state.PushNode(new CZScriptExpressionConst<
    SINT8>($1)),    SYMBOL_CONST); }
```

```
|       T_INT                   { $$.Set(AMD_SINT32, state.PushNode(new CZScriptExpressionConst<
    SINT32>($1)),   SYMBOL_CONST); }
|       T_REAL                  {
                if (state.GetRealArithmeticsPrecisionExtended())
                        $$.Set(AMD_FLT64, state.PushNode(new CZScriptExpressionConst<FLT64>($1)),
                            SYMBOL_CONST);
                else
                        $$.Set(AMD_FLT32, state.PushNode(new CZScriptExpressionConst<FLT32>((float
                            )$1)), SYMBOL_CONST);
        }
|       T_STRING                { $$.Set(AMD_STDSTRING, state.PushNode(new CZScriptExpressionConst
    <STDSTRING>($1)), SYMBOL_CONST | SYMBOL_REF); }
|   T_NEWSYMBOL                 {
                state.SetError(@1, _T("Undefined symbol \"%s\""), $1);
                YYERROR;
        }
|       lval                    {
                if ($1.GetExpression() == 0) {
                        _ASSERT($1.GetType() == AMD_TYPE_UNKNOWN);
                        YYERROR;
                }
                $$ = $1;
        }

|       '-' expr %prec T_UMINUS { if (! AllocExpressionUMinus(state, $2, @2, $$)) YYERROR; }
|       expr '+' expr   { if (! AllocExpressionPlus    (state, $1, @1, $3, @3, $$)) YYERROR; }
|       expr '-' expr   { if (! AllocExpressionMinus   (state, $1, @1, $3, @3, $$)) YYERROR; }
|       expr '*' expr   { if (! AllocExpressionMultiply(state, $1, @1, $3, @3, $$)) YYERROR; }
|       expr '/' expr   { if (! AllocExpressionDivide  (state, $1, @1, $3, @3, $$)) YYERROR; }
|   expr  T_EQ   expr   { if (! AllocExpressionCompare(state, OP_EQ, $1, @1, $3, @3, $$)) YYERROR;
    }
|   expr  T_NE   expr   { if (! AllocExpressionCompare(state, OP_NE, $1, @1, $3, @3, $$)) YYERROR;
    }
|   expr  '>'    expr   { if (! AllocExpressionCompare(state, OP_GT, $1, @1, $3, @3, $$)) YYERROR;
    }
|   expr  '<'    expr   { if (! AllocExpressionCompare(state, OP_LT, $1, @1, $3, @3, $$)) YYERROR;
    }
|   expr  T_GE   expr   { if (! AllocExpressionCompare(state, OP_GE, $1, @1, $3, @3, $$)) YYERROR;
    }
|   expr  T_LE   expr   { if (! AllocExpressionCompare(state, OP_LE, $1, @1, $3, @3, $$)) YYERROR;
    }
|   T_NOT  expr %prec T_UMINUS { if (! AllocExpressionNot(state, $2, @2, $$)) YYERROR; }
|   expr  T_AND  expr { if (! AllocExpressionBoolean(state, OP_AND, $1, @1, $3, @3, $$)) YYERROR;
    }
|   expr  T_OR   expr { if (! AllocExpressionBoolean(state, OP_OR,  $1, @1, $3, @3, $$)) YYERROR;
    }
|   expr  T_XOR  expr { if (! AllocExpressionBoolean(state, OP_XOR, $1, @1, $3, @3, $$)) YYERROR;
    }
|       funcall
|       funcall '[' expr ']'                    { AllocExpressionArrayItemVal(state, $1, @1, $3,
    @3, $$); }
|       funcall '[' expr ',' expr ']'   { AllocExpressionArrayItemDim2(state, $1, @1, $3, @3, $5,
    @5, false, $$); }
|   T_FEATURE_SCALAR { if (! AllocExpressionFeatureAccess(state, FEATURE_TYP_SCALAR, $1, @1, $$))
    YYERROR; }
|       T_FEATURE_POINT  { if (! AllocExpressionFeatureAccess(state, FEATURE_TYP_POINT,  $1, @1,
    $$)) YYERROR; }
|   T_FEATURE_PLANE  { if (! AllocExpressionFeatureAccess(state, FEATURE_TYP_PLANE,  $1, @1, $$))
    YYERROR; }
|   T_FEATURE_CURVE  { if (! AllocExpressionFeatureAccess(state, FEATURE_TYP_CURVE,  $1, @1, $$))
    YYERROR; }
|   T_FEATURE_AREA       { if (! AllocExpressionFeatureAccess(state, FEATURE_TYP_AREA,   $1, @1,
    $$)) YYERROR; }
|   T_FEATURE_AREACOLLECTION { if (! AllocExpressionFeatureAccess(state,
    FEATURE_TYP_AREACOLLECTION, $1, @1, $$)) YYERROR; }
|   T_DEFINED     '(' feature ')' { if (! AllocExpressionFeatureDefined(state, $3, @3, $$))
    YYERROR; }
|       T_PARAMETER              { if (! AllocExpressionParamAccess(state, $1, @1, $$)) YYERROR; }
;
feature:        T_FEATURE_SCALAR | T_FEATURE_POINT | T_FEATURE_PLANE | T_FEATURE_CURVE |
    T_FEATURE_AREA | T_FEATURE_AREACOLLECTION;
symname:        T_NEWSYMBOL
|                       T_FUNCTION      { wcscpy($$, $1->GetNameW().c_str()); }
|                       T_VARIABLE      { wcscpy($$, $1->GetNameW().c_str()); }
function:
        procedure               { $$ = $1; }
|       T_POINTTYPE             { state.PushExpressionList(); $$ = state.GetSymbol(_T("Point")); }
|       T_VECTORTYPE    { state.PushExpressionList(); $$ = state.GetSymbol(_T("Vector")); }
|       T_PLANETYPE             { state.PushExpressionList(); $$ = state.GetSymbol(_T("Plane")); }
;
procedure:
        T_FUNCTION      { state.PushExpressionList(); $$ = $1; }
|       T_VARIABLE      { state.PushExpressionList(); $$ = $1; }
|       T_NEWSYMBOL { state.PushExpressionList(); $$ = state.AddSymbolUndefined($1); }
;
exprlist:
        expr                            {
                state.GetExpressionList().m_aExpressions.push_back($1);
                state.GetExpressionList().m_aPositions.push_back(@1);
        }
|       exprlist ',' expr       {
                state.GetExpressionList().m_aExpressions.push_back($3);
                state.GetExpressionList().m_aPositions.push_back(@3);
        }
;
```

```
exprlistnamed:
        symname                                                 '=' expr        {
                state.GetExpressionList().m_aNames.push_back($1);
                state.GetExpressionList().m_aPositions.push_back(@3);
                state.GetExpressionList().m_aExpressions.push_back($3);
        }
|       exprlist     ',' symname      '=' expr     {
                state.GetExpressionList().m_aNames.push_back($3);
                state.GetExpressionList().m_aPositions.push_back(@5);
                state.GetExpressionList().m_aExpressions.push_back($5);
        }
;
// end of simplified parser code
```

List of Figures

1.1 Structure of a rule-based expert system [VDI 92]. 3
1.2 Knowledge Base development cycle showing the interaction of knowledge engineer and domain experts. 4

2.1 Evolution of leg prostheses. 11
2.2 Examples of exoprostheses. 13
2.3 Cochlear implants. 15
2.4 Different types of cosmeses. 16
2.5 Examples of different kinds of implants. 17
2.6 Anatomy of the human skull, courtesy of M. Ruiz Villarreal [Vill 10]. 19
2.7 Customized jaw prosthesis. 20
2.8 Example of customized dental crown design, courtesy of Adolph et al. [Adol 01]. 21
2.9 Anatomy of the tooth. 22
2.10 Anatomy of the human ear. 23
2.11 Hearing aid manufacturing. 24
2.12 Examples of SLA printed shells showing the three major device types: ITE, ITC and CIC. 25
2.13 Brief overview of the customized HA design workflow. 26

3.1 Concavity and peak features are typically represented by the most prominent point, e. g. according to a height function. 30
3.2 Examples of plane features. The aperture plane, first bend and second bend plane features are defined by the elbow like bends in the ear canal. In contrast to that, the bottom-opening plane is derived from the rough opening contour of the mesh. 32
3.3 Examples of curve features and corresponding feature points. 33
3.4 Examples of area features and associated feature points. 34
3.5 Ear impression with inscribed coordinate system. The x- and y-axis are in the bottom-opening plane. The z-axis is the normal of the bottom-opening plane. 35
3.6 Detection of peaks based on analyzing the level sets of a height function for topological changes. 36
3.7 Detection of concavities based on analyzing the signed curvature of contours slicing the impression. In the picture, yellow markings indicate convex areas, while the light blue indicate concave areas. 36

3.8 Schematic view of the elbow detection. The algorithm computes a contour profile along the canal in a region of interest S_{ROI}. The resulting radial contours $c_i(t)$ are analyzed and the maximum curvature points of each $c_i(t)$ are pooled in a set Q. Finally, these points are used to define an elbow plane after excluding outliers in Q. 37
3.9 Detection of ridges based on previously detected feature points and a curvature constrained geodesic computation between these points. Note: The yellow slicing contours shown on the upper right of the impression are a result of the concavity detection of the crus region as shown in Figure 3.7. 38
3.10 Despite a rough similarity, each ear impression is unique. Even for one patient, left and right ear show differences in shape and size. The left column depicts the left and the middle column the right ear of one patient. In addition the scan quality differs strongly dependent on the skill of the audiologist yielding to different types of noise at top and bottom of the impression, as shown in the right column. 39
3.11 Crus-valley area detected on an ear impression. 41
3.12 Updating of features is not always meaningful. For example the helix-peak cannot be found anymore in the processed impression on the right. 41
3.13 The result of the scanning and subsequent triangulation largely depends on how the user places the impression in the scanner, if the mold was cut by the audiologist and how much material was used to acquire the impression. 42
3.14 The human ear exhibits a huge variety in size, shape and smoothness. The latter is strongly influenced by the cleanliness of the ear. 43
3.15 Visualization of the clustering-based feature detection scheme. It utilizes prior knowledge of previously labeled impressions to acquire rough features with the help of a transformation $\boldsymbol{T}$ defined by a registration of ear impressions. 44
3.16 The point-to-plane error metric minimizes the distance d_j between the tangent plane $\boldsymbol{h}_j$ and the point $\boldsymbol{p}_i$. 47
3.17 Result of the centerline computation showing the rough (jagged) and final (smooth) centerline. 49
3.18 Point matching scheme for centerline based ICP. 50
3.19 Two possible outcomes of the CRP for seven customers $C = \{c_1, c_2, \ldots, c_7\}$ using four tables $K = \{k_1, k_2, k_3, k_4\}$. The first partition (1,3)(2)(4,6,7)(5) and the second partition (1,2)(3,4,5)(6)(7) are different, but according to the Chinese restaurant process their probability is equal. 57
3.20 Sample of sample set $\mathcal{S}_{\text{cut}}$ along with the original sample. 64
3.21 Sample of sample set $\mathcal{S}_{\text{rot}}$ along with the original sample. 64
3.22 Comparison of the cumulative registration error of the three selection techniques. 65
3.23 Comparison of the cumulative error of the rejection strategies. 66
3.24 Results of point weighting strategy showing the summed up error. . . 68

3.25 The AHC with farthest neighbor criterion resulted in nine clusters. Three of the nine cluster centroids are displayed here. Each of the centroids is shown from different view points to emphasize the differences in their shapes. 70
3.26 Two aligned ear impressions. 75
3.27 Examples of plane features. 76

4.1 Scheme of the parsing and interpretation pipeline. 78
4.2 Usage of the tools flex and bison to create a KB (script) interpreter. Detailed information about the `ScriptInterpreter.l` and `ScriptInterpreter.y` files can be found in the Appendix A on page 151. 79
4.3 Overview of the functionality of the developed knowledge representation language. 83
4.4 Scheme of a knowledge base or script. The guide rules of the KB are the high level building blocks defining the design process. Depending on the given options different paths from the starting to the finishing rule have to be taken. The path of execution is controlled by if-then-else conditions. 85
4.5 Visualization functions provide functionality to switch the visualization of the CAD environment according to the needs of the process step, for example, displaying the surface either transparent or solid. . 88
4.6 The KB contains the knowledge of the work instructions, experts and own experience. In combination with the detected features the ES shell executes the rules and controls the modeling software. 90

5.1 Two ear impressions of data set $\mathcal{S}_1$. The upper row shows the original impression and the lower row after application of the expert labeled *rough cut* plane. 94
5.2 GP starts with an initial (random) population, iterates a number of times through the evolutionary loop until presenting a solution. The evolutionary loop contains the main GP steps: selection, crossover and mutation. The latter two are grouped together in *breed new population* in the figure. 97
5.3 Two ASTs with depth 2. The left one is a *full* tree having terminals (T) only at depth 2, while the *grown* tree on the right side contains terminals and functions (F) at all levels except level 0, where only functions are allowed. 99
5.4 Example of an individual defining the rough cut plane in form of an AST. The functions are denoted by circles while the terminals are indicated by rounded rectangles. Below the syntax tree the script representation is given. 100
5.5 FUSS randomly samples the fitness interval $\delta_{\mathrm{F}} \in [\delta_{\mathrm{F}_{\min}}, \delta_{\mathrm{F}_{\max}}]$ and selects individuals uniformly. $c(\delta_{\mathrm{F}})$ denotes the number of individuals with fitness δ_{F}. 102
5.6 Our GP crossover operator selects (type strong) a crossover point in each AST. The offspring is a result of the swapping of subtrees at the crossover point. 103

5.7 Scheme of the genetic programming framework. The evolutionary loop is in the center (blue). Start (initialization) and final result are indicated with red, while the possible parameter selections are colored in orange. 104

5.8 Workflow of a hearing aid design task. The different steps are marked as plane, point or area based process rules. The order is not entirely mandatory, but depends on the design goal: either the smallest possible instrument or the best possible vent. Depicted is the vent-centric workflow (two vent steps). The shown workflow is not complete, meaning that a couple of steps are grouped in one step for the sake of clarity, e. g. *Remove or Reduce Cymba*. Furthermore, a certain amount of process steps is device type (CIC, ITC, ITE) dependent, e. g. *Local Scooping at Crus Area* is only valid for ITE devices. 108

5.9 Two examples of the *Taper Canal Tip* rule. 112

5.10 Features employed by the *Taper Canal Tip* rule. On the left a side view of an ear impression is shown and on the right the intersection contour of a bend plane with the canal. Depending on the impression the length of the canal can vary significantly. In the drawing a very long canal is shown, usually it ends shortly after the second bend and the centerline ends there as well. The centerline after the second bend is usually unreliable, because the shape of the tip is often distorted by artifacts. 113

5.11 The receiver hole has to be placed next to the vent inlet. It has to be ensured that it does not intersect with the vent inlet and the tip contour. The available tip area can be small, especially if a wax guard structure is in place as shown on the right. 114

5.12 The receiver hole has to be placed together with the vent inlet on the tip of the canal. The vent inlet is placed first, therefore, the receiver hole has to be fitted in the remaining space ensuring a minimal distance to the vent of at least vent wall thickness, vent radius and a small offset for collision avoidance. A cosmetically appealing receiver hole position is the center of the remaining area. 115

5.13 The opaque black stripes mark the tip area. The darker red framed area, is the reduced target area utilized in the ESAM2 receiver hole placement rule. 116

5.14 Examples for the *Measured Shell Height Cut* rule. On the left an ITE and on the right a CIC is shown. 116

5.15 The hinge tool used in case of ITC devices for the faceplate position includes the definition of the measured cut. The two hinge points at the tragus side define the hinge axis. The third point on the concha bowl can be dragged to lower the faceplate. The rule places the three points and executes a lowering algorithm after approval by the user. The result is shown on the right. In contrast to the measured cuts for CICs and ITCs the hinge point shifts applied by the user are analyzed instead of the plane location and orientation. 118

5.16 The left impression features a rectangular cut at the canal tip, while the middle impression is cut at both sides and finally the right impression has a slanted cut at the tip and a rectangular one at the bottom side. These types of cuts are very common. Their use is to increase the wearing comfort by reducing occlusion effects. 121
5.17 For CIC devices the bottom opening is analyzed using PCA. The vent bottom point is placed on the major opening axis closest to the crus-ridge bottom point. Since the opening contour is not always elliptical, the tragus line is used as an additional constraint. 122
5.18 The vent bottom point for ITC devices is guided by the crus-ridge bottom point and constrained to be close to the tragus line and the cymba removal plane. The ITE vent bottom point is only constrained by the intersection of inter-tragal notch ridge and bottom contour. . . 122
5.19 The first *Local Fill at Cymba Area* rule defines a plane which is used to fill the cymba peak with additional material. The transparent area indicates the defined plane. The plane usually intersects with the cymba and the ear canal. In order to select the correct area, a reference point is used, which is based on the helix-peak feature point. The picture on the lower right side shows the result of the rule application. The inner wall of the impression is enlarged by the selected area. . . 123
5.20 The *Local Fill at Cymba Area* rule defines a plane in the cymba reaching, which is used to fill the cymba with material. The left picture shows the defined plane crossing the cymba starting from the bottom contour and leaving the shell close to the crus concavity area. The plane is perpendicular to the viewing direction which correlates to the tragus line depicted in Figure 5.17. The root cause for the cymba filling is to provide the acoustician with the option to manually remove material in case the patient is uncomfortable with the aid in this area. 125

6.1 Workflow scheme of ESAM. For each prosthesis design task a set of constraints $\mathcal{O}$ and a source shape S_{source} is given as input. These inputs will be provided to the knowledge base $\mathcal{R}$ respectively to the feature detector $\mathcal{F}$. The inference machine executes all applicable rules, which yields application of CAD operations and consequently the modification of the source shape. During rule execution the feature detector updates the features if necessary. The blocks marked in blue are considered general, while the red (dotted) blocks are specific for the design application in mind. 130
6.2 The manual guide rule *Check Quality of Design* displays the designed HA shell aligned with the original impression enhanced by a color mapping. The color mapping indicates if the designed shell has a positive (yellow/red) or negative (blue) offset compared to the original impression. This information is used by the designer to decide if the HA fits nicely in the ear canal without causing pain or being too loose. 132
6.3 Visualization of the ESAM-completion rate (Definition 9) together with the average number of rules contained in a design record. 134

6.4 Visualization of the developed shell measures for design quality comparisons. 139

Ownership of used images and pictures

Please note that many pictures contained in this thesis are not property of the author. The author was granted the right to use them in this thesis, but it is not allowed to further distribute or use them separately. In case of questions concerning the ownership of a certain picture, please either address the author or check the bibliography for further information on the ownership.

List of Tables

3.1 List of features composing the Canonical Ear Signature [Balo 10a]. The feature class indicates the appearance on the ear impression. The feature type defines how the detected feature is represented in our system. Concavities for example may be represented by the center point of the concavity or the whole area. Features marked with a ⋆ are considered as core features. 31

3.2 Default values for ICP experiments. 65

3.3 Results of the different point selection strategies, showing the average registration error and the average computation time. Note: The average computation time includes data loading, data modification, centerline alignment and final registration. 65

3.4 Results of point rejection strategy showing the average registration error and the average computation time for each registration in seconds 66

3.5 Results of point weighting strategy showing the average registration error (in case of noise and no noise) and the average computation time in seconds. The average number of point pairs is approximately 749. . 67

3.6 Results of AHC for different linkage criteria. 69

3.7 Rand index computed pairwise for the AHC clustering results generated by different linkage criterias. 69

3.8 Results of clustering experiments using the CRC. 71

3.9 Range of rand and adjusted rand index for the ten experiments shown in Table 3.8. 71

3.10 Rand and adjusted rand index for different clustering methods. The upper right triangle contains the values of the rand index, while the lower left triangle contains the values of the adjusted rand index. . . . 71

3.11 Results of the surface-based feature detection. The mean and standard deviations are averaged for all features of the same feature type. . . . 72

3.12 Results of detection and tolerance rate analysis. 73

3.13 Results of the surface-based and clustering-based feature detection. All values are given in mm and stdev denotes the standard deviation. 73

3.14 Performance measures for surface-based and clustering-based feature detection. Note: The large difference in the Δ_{tol} reported here and in Table 3.12 is due to the missing reference values, which were replaced with the detected values in the former evaluation. 74

3.15 Results of the surface-based and clustering-based feature detection. Error in respect of plane orientation and location is given, stdev denotes the standard deviation. 74
3.16 Tolerance values for the detected crus-valley plane considering orientation and location. 75

4.1 Data types supported by the script language. 82
4.2 Operators supported by the script language. All operators support implicit casting. 82
4.3 The table shows the quality values given by the expert operators using ESAM to design a hearing aid device. For the sake of clarity and space, similar frames are grouped together. μ denotes the mean quality value of all operators for a certain process rule, σ the standard deviation and $\tilde{x}$ the median. 91

5.1 Overview of the number of design records for each location and each version contained in data set $\mathcal{S}_2$. 95
5.2 Default parameter values for the GP-framework. The values are based on experience and literature suggestions [Koza 92, Poli 08]. 104
5.3 Fitness evaluation with and without rule update. μ indicates the mean fitness value and σ the standard deviation. Subscript 0 denotes the original rule and subscript 1 the updated rule. 106
5.4 Rule performance classification matrix. 109
5.5 Performance of the ESAM1 system. Similar guide rules (using similar design rules) are grouped together. Note: δ_a and δ_u are computed for each complete guide rule, regardless if the record is ESAM-complete. 110
5.6 Result of the RA for the *Taper Canal Tip* rule for the different ESAM versions. 111
5.7 Result of the RA for the *Place Receiver Hole* rule for the different ESAM versions. 114
5.8 Result of the RA for the different specifications of the *Measured Shell Height Cut* for ESAM1. 117
5.9 Result of the RA for the *Measured Shell Height Cut* rule for ITCs for the different ESAM versions. The average distance error describes the average shift done by the user for both hinge points. The knowledge extraction focused on the relationship between hinge points and bottom contour. It was analyzed, if the hinge points were moved up or down. 118
5.10 Result of the RA for the *Measured Shell Height Cut* rule for CICs for the different ESAM versions. In order to extract knowledge from the data, we analyzed the angles between the applied rough cut at the bottom and the applied cymba removal cut. 119
5.11 Result of the RA for the *Readjust Vent Points* rule for the different ESAM versions. 120
5.12 Result of the RA for the first *Local Fill at Cymba Area* rule for the different ESAM versions. 124

5.13 Result of the RA for the second *Local Fill at Cymba Area* rule for the different ESAM versions. 126
5.14 Performance of the ESAM3 system. Similar process (using similar rules) are grouped together. The values in bold font indicate performance improvements compared to ESAM1 (Table 5.5) 126

6.1 Development of the number of guide rules, the total number of lines of code and the ESAM-completion rate for all three major ESAM versions. In the last column only the number / percentage of completed records is given. The total number of design records were presented in Table 5.1. 131
6.2 Development of manual and autoapply guide rules for the three major ESAM versions. 133
6.3 Guide rule performance analysis for the different ESAM versions based on the acceptance rate δ_a. 134
6.4 Ranking of exit rules for ESAM1. Goal is to finish always with the *Place Label Inside of Shell* rule. The ESAM-valid column shows the exit rate for design records containing at least ten rule records (Definition 10). The last column shows the exit rate for all available design records. 136
6.5 Ranking of exit rules for ESAM2. The ESAM-valid column shows the exit rate for design records containing at least ten rule records (Definition 10). The last column shows the exit rate for all available design records. 136
6.6 Ranking of exit rules for ESAM3. The ESAM-valid column shows the exit rate for design records containing at least ten rule records (Definition 10). The last column shows the exit rate for all available design records. 136
6.7 Results of the consistency analysis. The set designed with ESAM shows superior (smaller) values than the manual design for all consistency measures. 138
6.8 Comparison of the ESAM and manually modeled shells. The shell measures are computed for all shells and for each device type. 140
6.9 Comparison of the ESAM and manual modeled shells. The shell measures are computed for all shells and for each device type. 140
6.10 Design time study comparing the manual and the semi-automated design process based on 90 design tasks. 141
6.11 Design time for customized HA modeling with and without ESAM. . 141

Bibliography

[3Dsy 10] 3Dsystems. "Viper™ SLA System". http://www.3dsystems.com/products/sla/viper/viper_sla.asp, September 2010.

[3Sha 10] 3Shape A/S. "3Shape official website". http://www.3shape.com/, June 2010.

[Adol 01] S. Adolph and S. Gürke. "Modeling of a Fitting Inlay from Various Informations". In: *VMV 2001 – Vision, Modeling, and Visualization Workshop, November, 21 - 23, 2001, Stuttgart, Germany, Proceedings*, pp. 309–316, 2001.

[Aho 85] A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers: Principles, Techniques and Tools*. Addison-Wesley, Amsterdam, 1 Ed., January 1985.

[Aker 09] R. Akerkar and P. Sajja. *Knowledge-Based Systems*. Jones & Bartlett Pub, Sudbury, MA, 1 Ed., August 2009.

[Aldo 85] D. Aldous, I. Ibragimov, J. Jacod, and D. Aldous. "Exchangeability and related topics". In: *École d'Été de Probabilités de Saint-Flour XIII - 1983*, pp. 1–198, Springer, Berlin/Heidelberg, 1985.

[Andr 94] D. Andre. "Learning and Upgrading Rules for an OCR System Using Genetic Programming". In: *WCCI 1994 – IEEE World Congress on Computational Intelligence, June 26 - July 2, 1994, Orlando, FL, USA, Proceedings*, pp. 27–29, 1994.

[Ange 10] C. Angeli. *Advanced Knowledge-Based Systems: Models, Applications and Research*. The Technomathematics Research Foundation (TMRF), Kolhapur, 2010.

[Arun 87] K. S. Arun, T. S. Huang, and S. D. Blostein. "Least-squares fitting of two 3-D point sets". *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, Vol. 9, No. 5, pp. 698–700, September 1987.

[Azer 10] S. Azernikov. "Computer aided design of ventilation tubes for customized hearing aid devices". *Computer-Aided Design*, Vol. 42, No. 2, pp. 87–94, February 2010.

[Badr 10] K. Badran and P. Rockett. "The influence of mutation on population dynamics in multiobjective genetic programming". *Genetic Programming and Evolvable Machines*, Vol. 11, pp. 5–33, March 2010.

[Bali 09] M. Bali. *Drools Jboss Rules 5.0 Developer's Guide*. Packt Publishing, Birmingham, 2009.

[Balo 10a] S. Baloch, R. Melkisetoglu, S. Flöry, S. Azernikov, G. G. Slabaugh, A. Zouhar, and T. Fang. "Automatic Detection of Anatomical Features on 3D Ear Impressions for Canonical Representation". In: *MICCAI 2010 – 13th International Conference on Medical Image Computing and Computer Assisted Intervention, September, 20 - 24, 2010, Beijing, China, Proceedings*, pp. 555–562, 2010.

[Balo 10b] S. Baloch, A. Zouhar, and T. Fang. "Deformable Registration of Organic Shapes via Surface Intrinsic Integrals: Application to Outer Ear Surfaces". In: *MCV 2010 – Medical Computer Vision. Recognition Techniques and Applications in Medical Imaging (International MICCAI Workshop), September 20, 2010, Beijing, China, Revised Selected Papers*, pp. 11–20, 2010.

[Band 07] S. Bandyopadhyay, A. Mukhopadhyay, and U. Maulik. "An improved algorithm for clustering gene expression data". *Bioinformatics*, Vol. 23, No. 21, pp. 2859–2865, August 2007.

[Besl 92] P. J. Besl and N. D. McKay. "A Method for Registration of 3-D Shapes". *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, Vol. 14, No. 2, pp. 239–256, February 1992.

[Bibb 00] R. Bibb, A. Lewis, and R. Brown. "The application of design methods in medical modelling and reconstructive surgery". In: *22nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society, July 23 - 28, 2000, Chicago, IL, USA, Proceedings*, pp. 2517–2519, 2000.

[Bish 06] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, New York, NY, 2006.

[Blei 11] D. M. Blei and P. Frazier. "Distance Dependent Chinese Restaurant Processes". *Journal of Machine Learning Research*, Vol. 12, pp. 2461–2488, August 2011.

[Bron 01] I. N. Bronstein and K. A. Semendjajew. *Taschenbuch der Mathematik*. Verlag Harri Deutsch, Thun and Frankfurt am Main, 5 Ed., 2001.

[Buch 84] B. Buchanan and E. Shortliffe. *Rule-based expert systems: the MYCIN experiments of the Stanford Heuristic Programming Project*. Addison-Wesley, Boston, MA, 1984.

[Chen 92] Y. Chen and G. Medioni. "Object modelling by registration of multiple range images". *Image and Vision Computing*, Vol. 10, No. 3, pp. 145–155, April 1992.

[Comm 47] Committee on Artificial Limbs. *National Research Council: Terminal research reports on artificial limbs (covering the period from April 1, 1945 through June 30, 1947)*. National Research Council, Washington D.C., USA, 1947.

[Cram 98] J. Cram, G. Kasman, and J. Holtz. *Introduction to Surface Electromyography*. Aspen Publisher, Gaithersburg, MD, 1 Ed., 1998.

[Craw 02] R. Crawford-Marks and L. Spector. "Size Control Via Size Fair Genetic Operators In The PushGP Genetic Programming System". In: *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, July 9 - 13, 2002, New York, NY, USA, Proceedings*, pp. 733–739, 2002.

[Cyfe 11] Cyfex AG. "Secret Ear System". http://www.secreteardesigner.com/designer.php, August 2011.

[Dark 07] S. Darkner, R. R. Paulsen, and R. Larsen. "Analysis of Deformation of the Human Ear and Canal Caused by Mandibular Movement". In: *MICCAI 2007 – 10th International Conference on Medical Image Computing and Computer Assisted Intervention, October 29 - November 2, Brisbane Australia, Proceedings*, pp. 801–808, 2007.

[De J 06] K. A. De Jong. *Evolutionary Computation, A Unified Approach.* The MIT Press, Cambridge, MA, 2006.

[Demp 77] A. P. Dempster, N. M. Laird, and D. B. Rubin. "Maximum Likelihood from Incomplete Data via the EM Algorithm". *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol. 39, No. 1, pp. 1–38, 1977.

[Dijk 59] E. W. Dijkstra. "A note on two problems in connexion with graphs". *Numerische Mathematik*, Vol. 1, No. 1, pp. 269–271, December 1959.

[Dill 01] H. Dillon. *Hearing aids.* Thieme, Sydney, 1 Ed., 2001.

[Dong 05] W. Dongmei, W. Chengtao, Z. Xiujuan, and X. Liqun. "Design and Biomechanical Evaluation of a Custom lateral mandible Titanium Prosthesis". In: *IEEE-EMBS 2005 – 27th Annual International Conference of the Engineering in Medicine and Biology Society, September 1 - 4, 2005, Shanghai, China, Proceedings*, pp. 6188–6191, 2005.

[Donn 03] C. Donnelly and R. M. Stallman. *The Bison Manual, Using the YACC Compatible Parser Generator.* Free Software Foundation, Boston, MA, 8 Ed., 2003.

[Dowl 07] C. Dowling and S. Leech. "Audit support systems and decision aids: Current practice and opportunities for future research". *International Journal of Accounting Information Systems*, Vol. 8, No. 2, pp. 92–116, June 2007.

[Duda 01] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* Wiley, New York, NY, 2 Ed., 2001.

[Dupr 09] T. L. Dupras, L. J. Williams, M. D. Meyer, C. Peeters, D. Depraetere, B. Vanthuyne, and H. Willems. "Evidence of amputation as medical treatment in ancient Egypt". *International Journal of Osteoarchaeology*, Vol. 20, No. 4, pp. 405–423, July/August 2009.

[Elva 10] Elvar Pálsson. "Image of Oscar Pistorius running in Kópavogur, Iceland". http://en.wikipedia.org/wiki/File:Oscar_Pistorius-2.jpg, June 2010. This file is licensed under the Creative Commons Attribution 2.0 Generic license.

[Erns 09] A. Ernst, R.-D. Battmer, and I. Todt, Eds. *Cochlear Implant heute.* Springer, Berlin/Heidelberg, 2009.

[Eufi 01] H. Eufinger and B. Saylor. "Computer-assisted Prefabrication of Individual Craniofacial Implants". *Journal of Association of Perioperative Registered Nurses (AORN)*, Vol. 74, No. 5, pp. 648–654, November 2001.

[Eufi 95] H. Eufinger, M. Wehmöller, E. Machtens, L. Heuser, A. Harders, and D. Kruse. "Reconstruction of craniofacial bone defects with individual alloplastic implants based on CAD/CAM-manipulated CT-data". *Journal of Cranio-Maxillofacial Surgery*, Vol. 23, No. 3, pp. 175–181, June 1995.

[Feig 96] E. A. Feigenbaum. "How the "What" Becomes the "How" - Turing Award Lecture". *Communications of the ACM*, Vol. 39, No. 5, pp. 97–104, May 1996.

[Ferg 73] T. S. Ferguson. "A Bayesian Analysis of Some Nonparametric Problems". *The Annals of Statistics*, Vol. 1, No. 2, pp. 209–230, March 1973.

[Fisc 81] M. A. Fischler and R. C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". *Communications of the ACM*, Vol. 24, No. 6, pp. 381–395, June 1981.

[Foge 05] D. B. Fogel. *Evolutionary computation.* Wiley-IEEE Press, Piscataway, NJ, 3 Ed., December 2005.

[Foge 66] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial intelligence through simulated evolution.* John Wiley & Sons, New York, NY, 1966.

[Fonl 01] C. Fonlupt. "Solving the ocean color problem using a genetic programming approach". *Applied Soft Computing*, Vol. 1, No. 1, pp. 63–72, 2001.

[Forg 79] C. L. Forgy. *On the efficient implementation of production systems.* PhD thesis, Carnegie Mellon University, 1979.

[Forg 82] C. L. Forgy. "Rete: a fast algorithm for the many pattern/many object pattern match problem". *Artificial Intelligence*, Vol. 19, No. 1, pp. 17–37, September 1982.

[Frie 03] E. Friedman-Hill. *Jess in Action: Rule-based Systems in Java.* Manning Publications, Greenwich, CT, 2003.

[Frig 10] B. A. Frigyik, A. Kapila, and M. R. Gupta. "Introduction to the Dirichlet Distribution and Related Processes". Tech. Rep. UWEETR-2010-0006, Department of Electrical Engineering, University of Washington, Seattle, WA, December 2010. http://www.ee.washington.edu/research/guptalab/publications/UWEETR-2010-0006.pdf.

[Futt 98] S. Fütterling, R. Klein, W. Straßer, and H. Weber. "Automated Finite Element Modeling of a Human Mandible With Dental Implants". In: *WSCG 1998 – 6th International Conference in Central Europe on Computer Graphics and Visualization, February 11, 1998, Plzen-Bory, Czech Republic, Proceedings*, pp. 103–110, 1998.

[Gao 09] T. Gao and S. S. Jarng. "Design and realization of hearing aids based 3D rapid shell molding CADCAM". In: *IEEE ISIE 2009 – IEEE International Symposium on Industrial Electronics, July 5 - 8, 2009, Seoul, Korea, Proceedings*, pp. 1488–1492, 2009.

[Gebh 03] A. Gebhardt. *Rapid Prototyping.* Hanser, München, 2003.

[Gebh 07] A. Gebhardt. *Generative Fertigungsverfahren, Rapid Prototyping – Rapid Tooling – Rapid Manufacturing.* Hanser, München, 3 Ed., 2007.

[Gell 74] W. Gellert, H. Küstner, M. Hellwich, and H. Kästner, Eds. *Kleine Enzyklopädie Mathematik*. VEB Bibliographisches Institut, Leipzig, 1974.

[Ghos 11] S. Ghosh, A. B. Ungureanu, E. B. Sudderth, and D. M. Blei. "Spatial distance dependent Chinese restaurant processes for image segmentation". In: *NIPS 2011 - Advances in Neural Information Processing Systems 24, 12-14 December, 2011, Granada, Spain*, pp. 1476–1484, 2011.

[Giar 05] J. C. Giarratano and G. D. Riley. *Expert Systems: Principles and Programming*. Course Technology, Boston, MA, 4 Ed., 2005.

[Gray 18] H. Gray. "Gray's anatomy of the human body (1918)". http://www.bartleby.com/107/229.html, 1918.

[Herr 03] H. Herr, G. P. Whiteley, and D. Childress. *Biologically Inspired Intelligent Robots, Cyborg Technology-Biomimetic Orthotic and Prosthetic Technology*. SPIE Press, Bellingham, WA, 2003.

[Heus 10] J. Heuser. "Artificial pacemaker". http://en.wikipedia.org/wiki/File:Pacemaker_GuidantMeridianSR.jpg, May 2010. This file is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported license.

[Hier 06] T. Hierl, G. Wollny, F. P. Schulze, E. Scholz, J.-G. Schmidt, G. Berti, J. Hendricks, and A. Hemprich. "CAD-CAM Implants in Esthetic and Reconstructive Craniofacial Surgery". *Journal of Computing and Information Technology (CIT)*, Vol. 14, No. 1, pp. 65–70, March 2006.

[Hieu 02] L. Hieu, E. Bohez, J. Vander Sloten, H. Phien, V. Esichaikul, P. Binh, P. An, N. To, and P. Oris. "Design and manufacturing of personalized implants and standardized templates for cranioplasty applications". In: *IEEE ICIT 2002 – IEEE International Conference on Industrial Technology, December 11 - 14, 2002, Bangkok, Thailand, Proceedings*, pp. 1025–1030, 2002.

[Hieu 10] L. C. Hieu, E. Bohez, J. V. Sloten, L. T. Hung, L. Khanh, N. Zlatov, and P. D. Trung. "Integrated approaches for personalised cranio-maxillofacial implant design and manufacturing". In: *BME 2010 – 3rd International Conference on the Development of Biomedical Engineering in Vietnam, January 11 - 14, 2010, Ho Chi Minh City, Vietnam, Proceedings*, pp. 119–122, 2010.

[Holl 92] J. H. Holland. *Artificial intelligence through simulated evolution*. MIT Press, Cambridge, MA, 1992.

[Hopg 01] A. A. Hopgood. *Intelligent systems for engineers and scientists*. CRC Press, Boca Raton, FL, 2 Ed., 2001.

[Hopk 05] N. Hopkinson, R. Hague, and P. Dickens, Eds. *Rapid Manufacturing: An Industrial Revolution for the Digital Age*. Wiley, Chichester, 2005.

[Horn 00] J. Hornegger and H. Niemann. "Probabilistic modeling and recognition of 3-d objects". *International Journal of Computer Vision*, Vol. 39, No. 3, pp. 229–251, September/October 2000.

[Horn 87] B. K. P. Horn. "Closed-form solution of absolute orientation using unit quaternions". *Journal of the Optical Society of America A*, Vol. 4, No. 4, pp. 629–642, April 1987.

[Horn 88] B. K. P. Horn, H. Hilden, and S. Negahdaripour. "Closed-Form Solution of Absolute Orientation using Orthonormal Matrices". *Journal of the Optical Society of America A*, Vol. 5, No. 7, pp. 1127–1135, July 1988.

[Howa 06] L. Howard and H. Lewis. "Support tool for material and process combinations during early designs". *International Journal of Production Research*, Vol. 44, No. 17, pp. 3379–3390, September 2006.

[Hutt 01] M. Hutter. "Fitness Uniform Selection to Preserve Genetic Diversity". In: *CEC 2002 – Congress on Evolutionary Computation, May 12 - 17, 2002, Honolulu, Hawaii, USA, Proceedings*, pp. 783–788, 2001.

[Iqba 05] A. Iqbal, N. He, and L. Li. "A fuzzy expert system for optimization of high-speed milling process". In: *RoMoCo 2005 – 5th International Workshop on Robot Motion and Control, June 23 - 25, 2005, Dymaczewo, Poland, Proceedings*, pp. 297–304, 2005.

[Jain 99] A. K. Jain, M. N. Murty, and P. J. Flynn. "Data clustering: a review". *ACM Computing Surveys*, Vol. 31, No. 3, pp. 264–323, September 1999.

[Jauh 04] S. Jauhar. "The Artificial Heart". *New England Journal of Medicine*, Vol. 350, No. 6, pp. 542–544, February 2004.

[Jord 05] M. I. Jordan. "Dirichlet Processes, Chinese Restaurant Processes and all that". http://www.cs.berkeley.edu/~jordan/nips-tutorial05.ps, 2005. Tutorial presentation at the NIPS conference, December 5 - 8, 2005, Vancouver and Whistler, BC, Canada.

[Jung 08] S. Jung, S. Kang, and I. Moon. "Design of biomimetic hand prosthesis with tendon-driven five fingers". In: *IEEE BioRob 2008 – 2nd IEEE/RAS EMBS International Conference on Biomedical Robotics and Biomechatronics, October 19 - 22, 2008, Scottdale, AZ, USA, Proceedings*, pp. 895–900, 2008.

[Kass 88] M. Kass, A. Witkin, and D. Terzopoulos. "Snakes: Active contour models". *International Journal of Computer Vision*, Vol. 1, No. 4, pp. 321–331, January 1988.

[Kinn 93] K. E. Kinnear, Jr. "Generality and Difficulty in Genetic Programming: Evolving a Sort". In: *ICGA93 – 5th International Conference on Genetic Algorithms, July 17 - 22, 1993, Urbana, IL, USA, Proceedings*, pp. 287–294, 1993.

[Klei 01] M. Klein, T. Lueth, A. Hein, M. Stien, O. Schermeier, S. Weber, H. Menneking, O. Schwerdtner, and J. Bier. "Robot-assisted insertion of craniofacial implants - clinical experience". *International Congress Series*, Vol. 1230, No. 0, pp. 131–137, June 2001. Computer Assisted Radiology and Surgery.

[Koni 10] Konica Minolta. "Konica Minolta 3D Laser Scanner: Applications In Medical Science". http://www.kmwebinars.com/app_notes/Medical_Brochure_03.pdf, September 2010.

[Kost 81] J. P. Kostuik. *Amputation Surgery and Rehabilitation: Toronto Experience*. Churchill Livingstone, New York, NY, 1981.

[Kouc 07] P. Kouchakpour, A. Zaknich, and T. Bräunl. "Population variation in genetic programming". *Information Sciences*, Vol. 177, No. 17, pp. 3438–3452, 2007.

[Koza 92] J. R. Koza. *Genetic programming: on the programming of computers by means of natural selection*. MIT Press, Cambridge, MA, 1992.

[Krus 10] R. Kruse, M. Steinbrecher, and M. Boettcher. "Temporal Aspects in Data Mining". In: *2010 IEEE World Congress on Computational Intelligence, July 18 - 23, 2010, Barcelona, Spain, Plenary and Invited Lectures*, pp. 1–22, IEEE, 2010.

[Lafe 10] J. Z. Laferrier and R. Gailey. "Advances in Lower-limb Prosthetic Technology". *Physical Medicine and Rehabilitation Clinics of North America*, Vol. 21, pp. 87–110, February 2010.

[Lau 07] J. W. Lau and A. Y. Lo. *Model-Based Clustering and Weighted Chinese Restaurant Processes*, pp. 405–424. Vol. 3 of *Series in Biostatistics*, World Scientific, Singapore, 2007.

[Lee 02] M. Lee, C. Chang, C. Lin, L. Lo, and Y. Chen. "Custom implant design for patients with cranial defects". *Engineering in Medicine and Biology Magazine, IEEE*, Vol. 21, No. 2, pp. 38–44, March/April 2002.

[Lee 03] M. Lee, C. Chang, C. Lin, L. Lo, and Y. Chen. "Medical rapid prototyping in custom implant design for craniofacial reconstruction". In: *SMC 2003 – IEEE International Conference on Systems, Man and Cybernetics, October 5 - 8, 2003, Washington DC, USA, Proceedings*, pp. 2903–2908, 2003.

[Lee 08] M. Lee, C. Chang, C. Kuo, and Y. Ku. "A new layer-based imaging and rapid prototyping techniques for computer-aided design and manufacture of custom dental restoration". In: *SMC 2008 – IEEE International Conference on Systems, Man and Cybernetics, October 12 - 15, 2008, Singapore, Proceedings*, pp. 2572–2577, 2008.

[Legg 04] S. Legg, M. Hutter, and A. Kumar. "Tournament versus Fitness Uniform Selection". In: *CEC 2004 – Congress on Evolutionary Computation, June 20 - 23, 2004, Portland, OR, USA, Proceedings*, pp. 2144–2151, IEEE, 2004.

[Legr 07] P. Legrand, C. Bourgeois-Republique, V. Péan, E. Harboun-Cohen, J. Levy-Vehel, B. Frachet, E. Lutton, and P. Collet. "Interactive evolution for cochlear implants fitting". *Genetic Programming and Evolvable Machines*, Vol. 8, No. 4, pp. 319–354, December 2007.

[Lehr 10] Lehrstuhl für Künstliche Intelligenz und Angewandte Informatik. "d3web". http://www.is.informatik.uni-wuerzburg.de/forschung/anwendungen/d3web/, August 2010.

[Levi 92] J. R. Levine, T. Mason, and D. Brown. *Lex & Yacc*. O'Reilly & Associates, Sebastopol, CA, 2 Ed., 1992.

[Liao 05] S.-H. Liao. "Expert system methodologies and applications–a decade review from 1995 to 2004". *Expert Systems with Applications*, Vol. 28, No. 1, pp. 93–103, 2005.

[Link 05] J. Link, P. Yager, and J. Anjos. "Application of genetic programming to high energy physics event selection". *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, Vol. 551, No. 2-3, pp. 504–527, October 2005.

[Luca 91] P. Lucas and L. van der Gaag. *Principles of Expert Systems. International Computer Science Series*, Addison-Wesley, Boston, MA, 1991.

[Luet 11] T. C. Lueth, T. Wenger, A. Rautenberg, and H. Deppe. "RoboDent and the change of needs in computer aided dental implantology during the past ten years". In: *ICRA 2011 – IEEE International Conference on Robotics and Automation, May 9 - 13, 2011, Shanghai, China, Proceedings*, pp. 2144–2151, IEEE, 2011.

[Mand 05] S. Mandl and H. Stoyan. "Evolution of teams for the asynchronous pursuit domain". *Computer Systems: Science & Engineering*, Vol. 20, No. 4, pp. 311–317, April 2005.

[Mast 05] M. Masters, T. Velde, and F. McBagonluri. *Rapid Manufacturing: An Industrial Revolution for the Digital Age*, Chap. Rapid Manufacturing in the Hearing Industry. Wiley, Chichester, 2005.

[Mate 11] Materialise NV. "Rapid Manufacturing". http://www.materialise.com/materialise/view/en/449917-Rapid+Manufacturing.html, August 2011.

[Mede 10] MedecineNet.com. "Definition of Prosthesis". http://www.medterms.com/script/main/art.asp?articlekey=5076, May 2010.

[Meht 04] R. P. Mehta and D. G. Deschler. "Mandibular reconstruction in 2004: an analysis of different techniques". *Current Opinion in Otolaryngology & Head and Neck Surgery*, Vol. 12, No. 4, pp. 288–293, August 2004.

[Mend 01] R. R. F. Mendes, F. de B. Voznika, A. A. Freitas, and J. C. Nievola. "Discovering Fuzzy Classification Rules with Genetic Programming and Co-Evolution". In: *PKDD 2001 – 5th European Conference on Principles of Data Mining and Knowledge Discovery, September 3 - 5, 2001, Freiburg, Germany, Proceedings*, pp. 314–325, 2001.

[Meye 03] U. Meyer, H. Wiesmann, C. Runte, T. Fillies, N. Meier, T. Lueth, and U. Joos. "Evaluation of accuracy of insertion of dental implants and prosthetic treatment by computer-aided navigation in minipigs". *British Journal of Oral and Maxillofacial Surgery*, Vol. 41, No. 2, pp. 102–108, April 2003.

[Mok 08] C. Mok, M. Hua, and S. Wong. "A hybrid case-based reasoning CAD system for injection mould design". *International Journal of Production Research*, Vol. 46, No. 14, pp. 3783–3800, July 2008.

[Mont 95] D. J. Montana. "Strongly typed genetic programming". *Evolutionary Computation*, Vol. 3, No. 2, pp. 199–230, June 1995.

[Morm 06] W. H. Mörmann. "The evolution of the CEREC system". *The Journal of the American Dental Association*, Vol. 137 suppl. 1, pp. 7S–13S, September 2006.

[Naka 10] J. Nakashima. "Artificial heart". http://en.wikipedia.org/wiki/File:CardioWest%E2%84%A2_temporary_Total_Artificial_Heart.jpg, May 2010. This file is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported license.

[Nerl 00] A. G. Nerlich, A. Zink, U. Szeimies, and H. G. Hagedorn. "Ancient Egyptian prosthesis of the big toe". *The Lancet*, Vol. 356, No. 9248, pp. 2176–2179, December 2000.

[Nord 98] P. Nordin, W. Banzhaf, and M. Brameier. "Evolution of a world model for a miniature robot using genetic programming". *Robotics and Autonomous Systems*, Vol. 25, No. 1-2, pp. 105–116, October 1998.

[Orab 09] F. Orabona, C. Castellini, B. Caputo, A. Fiorilla, and G. Sandini. "Model adaptation with least-squares SVM for adaptive hand prosthetics". In: *ICRA 2009 – IEEE International Conference on Robotics and Automation, May 12 - 17, 2009, Kobe, Japan, Proceedings*, pp. 2897–2903, 2009.

[Otto 07] Otto Bock HealthCare GmbH. "Armpassteile, 2007". http://www.ottobock.de/cps/rde/xbcr/ob_de_de/im_646k3_d.pdf, 2007.

[Otto 09a] Otto Bock HealthCare GmbH. "MYOBOCK-Armprothesen, 2009". http://www.ottobock.de/cps/rde/xbcr/ob_de_de/im_646k6_d.pdf, 2009.

[Otto 09b] Otto Bock HealthCare GmbH. "Prothetik, Untere Extremitäten, 2009". http://www.ottobock.de/cps/rde/xbcr/ob_de_de/646K2-D-02-0903web.pdf, 2009.

[Park 04] A. Parkins and A. Nandi. "Genetic programming techniques for hand written digit recognition". *Signal Processing*, Vol. 84, No. 12, pp. 2345–2365, December 2004.

[Paul 02] R. R. Paulsen, R. Larsen, S. Laugesen, C. Nielsen, and B. K. Ersbøll. "Building and Testing a Statistical Shape Model of the Human Ear Canal". In: *MICCAI 2002 – 5th International Conference on Medical Image Computing and Computer Assisted Intervention, September 25 - 28, 2002, Tokyo, Japan, Proceedings*, 2002.

[Paul 03] R. R. Paulsen and K. B. Hilger. "Shape Modelling Using Markov Random Field Restoration of Point Correspondences". In: *IPMI 2003 – 18th International Conference on Information Processing in Medical Imaging, July 20 - 25, 2003, Ambleside, UK, Proceedings*, pp. 1–12, 2003.

[Paul 04a] R. R. Paulsen. *Statistical Shape Analysis of the Human Ear Canal with Application to In-the-Ear Hearing Aid Design.* PhD thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2004.

[Paul 04b] R. R. Paulsen, C. Nielsen, S. Laugesen, and R. Larsen. "Using a Shape Model in the Design of Hearing Aids". In: J. M. Fitzpatrick and M. Sonka, Eds., *SPIE 2004 – Medical Imaging: Image Processing 2004, May 12, 2004, San Diego, CA, USA, Proceedings*, pp. 1304–1311, 2004.

[Pirz 06] C. Pirzanski. "Despite new digital technologies, shell modelers shoot in the dark". *The Hearing Journal*, Vol. 59, No. 10, pp. 28, 30–32, October 2006.

[Pitm 06] J. Pitman. *Combinatorial Stochastic Processes.* Vol. 1875 of *Lecture Notes in Mathematics*, Springer-Verlag, Berlin, 2006. Available at http://works.bepress.com/jim_pitman/1.

[Pitm 95] J. Pitman. "Exchangeable and partially exchangeable random partitions". *Probability Theory and Related Fields*, Vol. 102, No. 2, pp. 145–158, June 1995.

[Poli 08] R. Poli, W. B. Langdon, and N. F. McPhee. *A field guide to genetic programming*. Published by `http://lulu.com`, 2008. Available at http://www.gp-field-guide.org.uk.

[Poli 97] R. Poli and W. B. Langdon. *Genetic Programming with One-Point Crossover and Point Mutation*, pp. 180–189. Springer, Berlin/Heidelberg/New York, 1997.

[Potv 04] J.-Y. Potvin, P. Soriano, and M. Vallée. "Generating trading rules on the stock markets with genetic programming". *Computers & Operations Research*, Vol. 31, No. 7, pp. 1033–1047, June 2004.

[Psch 07] W. Pschyrembel. *Pschyrembel, Klinisches Wörterbuch*. Walter de Gruyter, Berlin, 261 Ed., 2007.

[Qin 06] Z. Qin. "Clustering microarray gene expression data using weighted Chinese restaurant process". *Bioinformatics*, Vol. 22, No. 16, pp. 1988–1997, August 2006.

[Rand 71] W. M. Rand. "Objective Criteria for the Evaluation of Clustering Methods". *Journal of the American Statistical Association*, Vol. 66, No. 336, pp. 846–850, December 1971.

[Rech 73] I. Rechenberg. *Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*. Fromman-Hozlboog, Stuttgart, 1973.

[Reut 05] P. Reuter. *Springer Großwörterbuch Medizin, Medical Dictionary*. Springer, Berlin/Heidelberg, 2 Ed., 2005.

[Rich 07] P. Richard and J. Fitzgerald. *Introduction to AutoCAD 2008, A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, 2007.

[Rile 10] G. Riley. "CLIPS: A Tool for Building Expert Systems". http://clipsrules.sourceforge.net/, August 2010.

[Rusi 01] S. Rusinkiewicz and M. Levoy. "Efficient Variants of the ICP Algorithm". *3DIm 2001 – 3rd International Conference on 3D Digital Imaging and Modeling, May 28 - June 1, 2001, Québec City, Canada*, pp. 145–152, 2001.

[Sach 99] M. Sachs, J. Bojunga, and A. Encke. "Historical Evolution of Limb Amputation". *World Journal of Surgery*, Vol. 23, No. 10, pp. 1088–1093, October 1999.

[Schu 94] C. M. Schuch and C. H. Pritham. "International Standards Organization Terminology: Application to Prosthetics and Orthotics". *Journal of Prosthetics and Orthotics (JPO)*, Vol. 6, No. 1, pp. 29–33, 1994.

[Scie 10a] Science Museum London. "Brought to life: Exploring the history of medicine, Aritificial peg leg, Europe, 1925-1935". http://www.sciencemuseum.org.uk/broughttolife/objects/display.aspx?id=93132&image=2, June 2010. Object number: 1999-468.

[Scie 10b] Science Museum London. "Brought to life: Exploring the history of medicine, Artificial right leg, Paris, France, 1946". http://www.sciencemuseum.org.uk/broughttolife/objects/display.aspx?id=5802, June 2010. Object number: 1999-464.

[Sega 09] A. Segal, D. Haehnel, and S. Thrun. "Generalized-ICP". In: *Robotics 2009 – Robotics: Science and Systems V, June 28 - July 1, 2009, Seattle, WA, USA, Proceedings*, 2009. http://www.roboticsproceedings.org/rss05/p21.pdf.

[Sick 07] K. Sickel, V. Daum, and J. Hornegger. "Shortest Path Search with Constraints on Surface Models of In-ear Hearing Aids". In: *IWK 2007 – 52nd Internationales Wissenschaftliches Kolloquium, September 10 - 13, 2007, Ilmenau, Germany, Proceedings*, pp. 221–226, 2007.

[Sick 09] K. Sickel, S. Baloch, V. Bubnik, R. Melkisetoglu, S. Azernikov, T. Fang, and J. Hornegger. "Semi-Automatic Manufacturing of Customized Hearing Aids Using a Feature Driven Rule-based Framework". In: *VMV 2009 – Vision, Modeling, and Visualization Workshop, November 16 - 18, 2009, Braunschweig, Germany, Proceedings*, pp. 305–312, 2009.

[Sick 10a] K. Sickel and V. Bubnik. "Iterative Closest Point Algorithm for Rigid Registration of Ear Impressions". In: *RBC 2010 – 6th Russian-Bavarian Conference on Bio-Medical Engineering, November 8 - 12, 2010, Moscow, Russia, Proceedings*, pp. 142–145, 2010.

[Sick 10b] K. Sickel and J. Hornegger. "Genetic Programming for Expert Systems". In: *WCCI 2010 – IEEE World Congress on Computational Intelligence, July 18 - 23, 2010, Barcelona, Spain, Proceedings*, pp. 2695–2702, 2010.

[Sick 11a] K. Sickel, S. Baloch, R. Melkisetoglu, V. Bubnik, S. Azernikov, and T. Fang. "Towards automation in hearing aid design". *Computer-Aided Design*, Vol. 43, No. 12, pp. 1793–1802, December 2011.

[Sick 11b] K. Sickel and V. Bubnik. "Clustering-based Detection of Anatomical Features on Organic Shapes". In: *BVM 2011 – Bildverarbeitung für die Medizin, March 20 - 22, 2011, Lübeck, Germany, Proceedings*, pp. 54–58, 2011.

[Siem 10a] Siemens AG. "Stärkstes Mini-Hörgerät der Welt: präzise und laut". http://www.siemens.com/press/pool/de/pressebilder/corporate_technology/in20080104-02_300dpi.jpg, September 2010.

[Siem 10b] Siemens AG. "Verstehen gibt Kraft". http://hearing.siemens.com/de/04-produkte/17-nitro/02-verstehen/understanding.jsp, September 2010.

[Sing 04] S. Singare, L. Dichen, L. Bingheng, L. Yanpu, G. Zhenyu, and L. Yaxiong. "Design and fabrication of custom mandible titanium tray based on rapid prototyping". *Medical Engineering & Physics*, Vol. 26, No. 8, pp. 671–676, October 2004.

[Sodi 05] R. Sodian, P. Fu, C. Lueders, D. Szymanski, C. Fritsche, M. Gutberlet, S. Hoerstrup, H. Hausmann, T. Lueth, and R. Hetzer. "Tissue engineering of vascular conduits: fabrication of custom-made scaffolds using rapid prototyping techniques". *The Thoracic and cardiovascular surgeon*, Vol. 53, No. 3, pp. 144–149, June 2005.

[Song 07] Y. Song, J. Li, L. Yin, T. Huang, and P. Gao. "The feature-based posterior crown design in a dental CAD/CAM system". *The International Journal of Advanced Manufacturing Technology*, Vol. 31, No. 11, pp. 1058–1065, February 2007.

[Stru 06] J. R. Strub, E. D. Rekow, and S. Witkowski. "Computer-aided design and fabrication of dental restorations: Current systems and future possibilities". *The Journal of the American Dental Association*, Vol. 137, No. 9, pp. 1289–1296, September 2006.

[Till 10] J. Tillander, K. Hagberg, L. Hagberg, and R. Branemark. "Osseointegrated Titanium Implants for Limb Prostheses Attachments: Infectious Complications". *Clinical Orthopaedics and Related Research*, Vol. 468, No. 10, pp. 2781–2788, October 2010.

[Togn 05] G. Tognola, M. Parazzini, P. Ravazzani, F. Grandori, A. Pesatori, M. Norgia, and C. Svelto. "Mesh reconstruction of hearing aid shells from unorganized 3D point-cloud". In: *IST 2005 – IEEE International Workshop on Imaging Systems and Techniques, May 13 - 14, 2005, Niagara Falls, ON, Canada, Proceedings*, pp. 42–45, 2005.

[Tsak 04] A. Tsakonas, G. Dounias, J. Jantzen, H. Axer, B. Bjerregaard, and D. G. von Keyserlingk. "Evolving rule-based systems in two medical domains using genetic programming". *Artificial Intelligence in Medicine*, Vol. 32, No. 3, pp. 195–216, November 2004.

[Turb 88] E. Turban. "Review of expert systems technology". *IEEE Transactions on Engineering Management*, Vol. 35, No. 2, pp. 71–81, May 1988.

[Turb 95] E. Turban. *Decision Support Systems and Expert Systems: Management Support Systems*. Prentice Hall, Upper Saddle River, NJ, 4 Ed., 1995.

[Unal 08] G. Unal, D. Nain, G. Slabaugh, and T. Fang. "Customized Design of Hearing Aids Using Statistical Shape Learning". In: *MICCAI 2008 – 11th International Conference on Medical Image Computing and Computer Assisted Intervention, September 6 - 10, 2008, New York, NY, USA, Proceedings, Part I*, pp. 518–526, 2008.

[Unal 11] G. Unal, D. Nain, G. G. Slabaugh, and T. Fang. "Generating shapes by analogies: An application to hearing aid design". *Computer-Aided Design*, Vol. 43, No. 1, pp. 47–56, January 2011.

[VDI 92] VDI-Gesellschaft Entwicklung Konstruktion Vertrieb (VDI-EKV); Gesellschaft für Informatik (GI), Ed. *Wissensbasierte Systeme für die Konstruktion und Arbeitsplanung*. VDI-Verlag GmbH, Düsseldorf, 1992.

[Vill 10] M. R. Villarreal. "Skull". http://en.wikipedia.org/wiki/File:Human_skull_side_simplified_(bones).svg and http://en.wikipedia.org/wiki/File:Human_skull_front_simplified_(bones).svg, June 2010. These images have been released into the public domain by its author, LadyofHats. This applies worldwide.

[Walk 91] M. W. Walker, L. Shao, and R. A. Volz. "Estimating 3-D location parameters using dual number quaternions". *CVGIP: Image Understanding*, Vol. 54, No. 3, pp. 358–367, November 1991.

[Wei 03] F. Wei, N. Celik, W. Yang, I. Chen, Y. Chang, and H. Chen. "Complications after Reconstruction by Plate and Soft-Tissue Free Flap in Composite Mandibular Defects and Secondary Salvage Reconstruction with Osteocutaneous Flap". *Plastic and Reconstructive Surgery*, Vol. 112, No. 1, pp. 37–42, July 2003.

[Will 97] M. Willis, H. Hiden, P. Marenbach, B. McKay, and G. A. Montague. "Genetic Programming: An Introduction and Survey of Applications". In: *GALESIA 1997 – 2nd International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, September 2 - 4, 1997, London, UK, Proceedings*, pp. 314–319, 1997.

[Wint 09] E. Wintermantel. *Medizintechnik, Life Science Engineering.* Springer, Berlin/Heidelberg, 5 Ed., 2009.

[Wist 09] T. Wiste, S. Dalley, T. Withrow, and M. Goldfarb. "Design of a multi-functional anthropomorphic prosthetic hand with extrinsic actuation". In: *ICORR 2009 – 11th International Conference on Rehabilitation Robotics, June 23 - 26, 2009, Kyoto, Japan, Proceedings*, pp. 675–681, 2009.

[Yama 02] T. Yamane. "The present and future state of nonpulsatile artificial heart technology". *Journal of Artificial Organs*, Vol. 5, No. 3, pp. 0149–0155, September 2002.

[Zhan 92] Z. Zhang. "Iterative Point Matching for registration of free-form curves". Tech. Rep. 1658, Rapports de Recherche, Programme 4: Robotique, Images et Vision, Institut National de Recherche en Informatique et en Automatique, Sophia Antipolis, Valbonne Cedex, France, 1992.

[Zhu 09] D. Zhu, A. Xu, Y. Qu, and T. Zang. "Customized Design and Fabrication of Permanent Dental Restoration". In: *BMEI 2009 – 2nd International Conference on BioMedical Engineering and Informatics, October 17 - 19, 2009, Tianjin, China, Proceedings*, pp. 1–4, 2009.

[Zouh 06] A. Zouhar, T. Fang, G. Unal, and F. McBagonluri. "Anatomically-Aware, Automatic, and Fast Registration of 3D Ear Impression Models". In: *3DPVT 2006 – 3rd International Symposium on 3D Data Processing, Visualization and Transmission, June 14 - 16, 2006, Chapel Hill, NC, USA, Proceedings*, pp. 240–247, 2006.

[Zouh 09] A. Zouhar, S. Baloch, S. Azernikov, C. Bahlmann, G. Unal, T. Fang, and S. Fuchs. "Freeform shape clustering for customized design automation". In: *3DIM 2009 – IEEE International Workshop on 3-D Digital Imaging and Modeling (In conjunction with ICCV 2009), October 3 - 4, 2009 Kyoto, Japan, Proceedings*, pp. 1590–1597, 2009.

[Zouh 10] A. Zouhar, S. Baloch, Y. Tsin, T. Fang, and S. Fuchs. "Layout Consistent Segmentation of 3-D Meshes via Conditional Random Fields and Spatial Ordering Constraints". In: *MICCAI 2010 – 13th International Conference on Medical Image Computing and Computer Assisted Intervention, September, 20 - 24, 2010, Beijing, China, Proceedings*, pp. 113–120, 2010.