Towards Automation in Hearing Aid Design

Konrad Sickel\textsuperscript{a,*}, Sajjad Baloch\textsuperscript{b}, Rupen Melkisetoglu\textsuperscript{b}, Vojtech Bubnik\textsuperscript{b}, Sergei Azernikov\textsuperscript{b}, Tong Fang\textsuperscript{b,*}

\textsuperscript{a}Pattern Recognition Lab, University of Erlangen-Nuremberg, Martensstr. 3
91058 Erlangen, Germany

\textsuperscript{b}Imaging and Visualization, Siemens Corporate Research, 755 College Road East
Princeton, NJ 08540, USA

Abstract

In the manufacturing of customized medical prostheses, such as in-the-ear hearing aids, the design process often is dictated by a source template representing the anatomy of a patient and a set of work instructions representing the description of surface modifications. Instead of carrying out the work instructions by hand with knife, file or drilling tools, the state-of-the-art relies on modern software tools, such as computer-aided-design and computer-aided-manufacturing. Work instructions are usually defined in terms of anatomical landmarks of a given template. Following the design phase, the virtual model of the customized prosthesis is produced by a rapid prototyping system, like selective laser sintering or stereolithography. An outstanding problem in prostheses design is that the work instructions are often vaguely defined, and a suitable outcome largely depends on the knowledge, experience and skill of the designer. In this paper, we present a solution to minimize the influence of human interaction. Our approach involves the abstraction of the work instructions into expert system rules that exploit a robustly identified canonical set of anatomic features. The versatility of our approach lies in \textit{a priori} defining an entire design workflow through a rule set, thereby yielding a high degree of automation that is flexible, customizable, consistent, and reproducible. The proposed solution is extensively evaluated in a real world application, and is shown to yield significant improvement in manufacturing. For instance, the consistency of the outcome was improved by about 10\% and the design time was reduced by about 8.4\%.

Keywords: Expert systems, Rule-based systems, Shape transformation, CAD,
1. Introduction

A major development in medical prostheses manufacturing in the last two decades has been the introduction of the computer-aided-design (CAD), and the computer-aided-manufacturing (CAM). The usage of these technologies precipitated an increase in the quality and the availability of various kinds of medical prostheses. So far, the focus of the CAD tools has been to improve the quality and the fitting of prostheses. The basic idea is to acquire an accurate account of the underlying anatomy via CT or 3-D laser scan to generate a 3-D model, which is utilized by an expert operator to design the desired target shape. Special attention is given to ensure the conformity of the target shape to the anatomy. Nonetheless this approach just replaces the physical cutting, grinding and filing tools with the virtual ones, controlled with a computer mouse. Neither it reduces the amount of manual labor involved, nor does it improve the consistency or reproducibility of the finished device. In this article, we specifically address these limitations of the existing CAD tools.

The general problem of automating the computer-aided-design is highly significant in various medical areas, such as hearing aid (HA) manufacturing, orthopedics, orthotics, and orthodontics. In general, medical prostheses manufacturing has to follow two fundamental requirements. First, a prosthesis has to fulfill its core functions, such as amplifying sound for HAs, resisting perpetual corrosion and pressure exerted on dental prostheses or achieving the desired shape to disguise deformations in the case of cosmetic prosthetics. Equally important, it should demonstrate a tolerable degree of conformity with the underlying anatomy. This is particularly important to ensure that it conveniently fits the anatomy of a patient. Due to the large amount of variability in size and shape, the design of such prostheses has traditionally been carried out manually. Thus, the resulting prosthesis has depended on the skills and the expertise of an operator and has failed to demonstrate desirable consistency.

Our proposed solution is to formalize the available knowledge and to augment the CAD system with an expert system that employs a feature detection unit. The resulting system is capable of designing a medical prosthesis (semi-) automatically.\textsuperscript{1}

The remainder of the paper is structured as follows. We start with an overview of the related work on the automatic manufacturing of medical prostheses and the detection of features on organic shapes. Section 3 provides a brief introduction to HA manufacturing. Section 4 formalizes the automation problem, followed by a discussion of the employed anatomical features and the

\textsuperscript{1}The framework is capable of achieving full automation, with further tweaking of the currently employed features and associated rules.
associated feature detection algorithms in Section 5. Section 6 explains the
knowledge base and the construction of associated rules. Before concluding, we
validate the proposed system in Section 7.

2. Literature Review

First attempts towards automatic manufacturing of medical prostheses were
initiated in the late 1980s by Dooley et al., who introduced the ORTHOPERT
system [1]. The ORTHOPERT consisted of a knowledge based system, helping
biomedical design engineers and clinicians in the design of implantable devices.
The ORTHOPERT was able to deduct from a given set of patient parameters
a customized design for a femur prosthesis including a finite element analysis of
the design. Similar work with focus on below-the-knee prostheses was done by
Riechmann et al., who exploited the advantages of CAD/CAM for this task [2].

In the field of dental prostheses, Hammond and Davenport introduced the
RaPiD system as a logic-based model of prostheses design [3]. RaPiD was
a declarative system that employed predicate logic for designing a removable
partial denture. A CAD/CAM solution for dental restoration was presented by
Zhu et al. [4]. Their work focused on the fabrication of customized dental crowns
based on CT data. Furthermore, Yan et al. [5] presented a semi-automatic rapid
prototyping framework for the fabrication of removable partial dentures, which
lacks the seamless and transparent integration of expert knowledge and feature
detection compared to our approach. An extensive review of the advantages
and the limitations of the existing CAD/CAM systems for dental technology
may be found in Strub et al. [6]. So far combining knowledge-based and CAD
systems resulted in applications working with non organic shapes. Lin et al. [7]
developed a framework for drawing dies, which reduces the development time
and costs. Sanders et al. [8] integrated a multi-expert system into a CAD
system, which is capable of identifying designs which are easily assembled by
machines or humans. In contrast to our work, these systems focus on combining
existing or designing non custom parts, while we focus on the automatic design
of organic shapes to a unique shape using expert knowledge and on the fly
detected features.

The detection of anatomical features on organic shapes is relatively less stud-
ied in the context of CAD systems, where major focus has been on well defined
industrial parts. For instance, Sunil et al. [9] dealt with non-organic shapes, and
Zhang et al. [10] provided the detection of geometric features based on NURBS
representation of the underlying shapes in the form of CAD models. Ye et al. [11]
went a step ahead by considering laser scanned surfaces, and employed geomet-
rical feature detection for reverse engineering. The problem is more complicated
for organic shapes, where clear boundaries between various regions usually do
not exist. Similar to [11], we work with the 3D scanned surfaces of the input
shapes. But in contrast, we consider organic shapes, and focus on the forward
design problem, which involves the detection of anatomical features. Recently,
Razdan et al. [12] addressed the problem of organic shapes, such as bones and
blood vessels, where they utilized a hybrid approach that combines vertex and
Figure 1: Examples of two ear impressions (ear negatives) after triangulation. Ear impressions display large variability in size and shape. In addition, the quality of the triangulated surface can be poor, either due to the complexity of the underlying shape, or due to improperly acquired impressions.

edge based feature recognition techniques. Despite some compositional similarities with [9, 12], we address a completely different problem, and consider more diverse set of features.

Recently, the digital manufacturing of customized hearing aids has started to gain interest of the research community. A group led by R. Larsen worked on several topics including statistical shape models of ear impressions, analysis of deformations in the human ear canal caused by mandibular movement and the generation of a one-size-fits-most hearing aid shell [13, 14, 15, 16]. Unal et al. [17], on the other hand, formulated the manufacturing process as a shape estimation problem. Their statistical shape model resulted in promising results on a limited set of ear impressions. Similar to their work, we validate the proposed approach with application to the customized HA design and manufacturing. However, our method differs significantly from theirs. In contrast to their one step solution, we decompose the design process into several well-defined and physically intuitive shape modification steps. Each step, in turn, is represented by a set of rules in a knowledge base (KB). Each rule in conjunction with some well-defined anatomical features leads to a shape modification operation. An inference machine of the resulting expert system applies the resulting operations on an input shape via the integrated CAD tools. This essentially simulates the design work instructions, i.e., the physical steps carried out by an expert operator. An added advantage is that it allows the integration of every detail of the design process, including various input constraints necessary for perfect customization and proper functioning of the resulting device [18]. Hence, our method is much more flexible as the previous approaches, which require a sufficiently large training set in order to incorporate process changes.
3. Customized Hearing Aid Manufacturing

Customized HA manufacturing is a delicate, complex and time consuming process. First, a template shape (ear impression) is acquired. Usually, an audiologist fills a malleable material into the ear of a patient, which is allowed to settle before it is taken out as the “negative” of the ear. The ear negative (ear impression) is then laser scanned to acquire a point cloud, which is triangulated using a specialized CAD software to reconstruct a mesh that represents the surface of the outer ear as shown in Fig. 1 [19, 20]. In this paper, we assume that the triangular mesh has already been reconstructed, and the design of the HA is required. Currently, this is done manually through the application of a series of interactive CAD operations representing a mesh transform.

Major CAD operations include cutting, rounding and smoothing of the input mesh. Virtual placement of electronic components (Fig. 2) is carried out to ensure that the selected electronics will fit in the HA shell during the actual physical assembly. In addition, operations to integrate other mesh structures, like the ventilation tube are performed. A vent is required, because a HA hermetically closes an ear, thereby causing pressure differences and acoustic feedback. The integration of a vent reduces these negative effects [21, 22]. Vents come in various sizes and shapes (Fig. 2) to accommodate different feedback properties, and their selection and placement is crucial for proper functioning of the device. In summary, the design of a HA involves various constraints, such as the target size of the device, the required amplification, vent size and style, feedback considerations and the extent of exact and comfortable fitting in the ear of a patient.

Once all constraints are satisfied and a customized shell is designed, rapid prototyping techniques, such as stereolithography, are employed to create a physical shell [23, 24]. Eventually, an operator manually assembles the shell and the associated components to construct the actual hearing aid (Fig. 3).
4. Problem Formulation

As described above, the design of medical prostheses involves the modification of a given anatomical template\textsuperscript{2} by an expert designer, driven by a set of bio-markers (features). The designer thereby relies on rules, which are either available as work instructions or have been acquired by personal experience. The problem may, therefore, be defined as that of a constrained shape transformation. Given an unprocessed shape (surface) $S_{\text{source}}$, represented as a triangulated mesh, the goal is to transform it into a target shape $S_{\text{target}}$, by a transformation $T_{R}$. $T_{R}$ is described as a sequence of transformations $T_{r_1}, T_{r_2}, \ldots$ defined by a set of rules $R = \{r_1, \ldots, r_n\}$:

$$S_{\text{target}} = T_{R} (S_{\text{source}}). \quad (1)$$

Typically, the transformation depends on constraints or options $O$, as well as anatomical features $F$ detected on the input shape. The features are utilized by the rules to determine the eventual transformation. Eq. (1) may, therefore, be extended to include this explicit dependence:

$$S_{\text{target}} = T_{R} (S_{\text{source}}; F, O). \quad (2)$$

A graphic representation of Eq. (2) is given in Fig. 4. Since the transform $T_{R}$ is derived from rules, Eq. (2) may be rewritten as an application of a sequence of the pertinent individual rules:

$$S_0 = S_{\text{source}},$$
$$S_{i+1} = T_{r_i} (S_i; F_i, O), \quad i = 0, \ldots, j - 1; \quad r_i \in R,$$
$$S_{\text{target}} = S_j, \quad (3)$$

where $S_i$ denotes the input shape after the application of $i$ rules. $F_i$, in turn, denotes the features detected on the current surface $S_i$. Eqs. (2) and (3) form the

\textsuperscript{2}The digitized impression of the anatomy of a patient.
The proposed framework and the associated rules are driven by a set of anatomical features ($F$) detected on an ear impression. Despite their rough similarity, each ear impression is unique. A typical ear impression (Fig. 1 and 5) consists of a long spindle shaped canal that fits deep in the outer ear and a body that sits in the external ear. The two meet at the narrow visible opening in the ear commonly known as the canal aperture. Two regions along the bottom of the ear, the tragus and the anti-tragus, hold a typical HA on either sides like clamps. On the top, the shell contains the narrow helix sandwiched between the anti-helix and the protruded crus. The so-called concha, on the other hand, offers a relatively large bowl area in the external ear for holding electronics inside a HA. All these features are distinctive among individuals in terms of their shape, and their relative size, depth and/or location.

In addition to these anatomically important features, we identify additional features (44 in total) that completely describe the HA manufacturing process. Some of them are illustrated in Figs. 6 and 7. Collectively, they capture the structure of an ear in the Canonical Ear Signature (CES) [27]. The accuracy of the resulting expert system for the design of HAs is largely determined by the accuracy of these features. We, therefore, develop automatic yet fast and robust algorithms for the detection of the CES. Our approach herein is to categorize the anatomical features of interest in terms of generic features, namely peaks,
concavities, elbows, and ridges, and a set of derived features. The idea is that although some anatomical features may not be represented by these geometric primitives, they may still be derived from the latter or a combination of other derived features.

5.1. Peak detection

A peak point is a prominent topological landmark on a surface $S$. We detect it via a height function $g : S \rightarrow \mathbb{R}$ that assigns to each point $p(x, y, z) \in S$ a value equal to its height, $g(p) := g(x, y, z) = z$. 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. 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.

5.2. 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 towards the lowest curvature point. 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.
5.3. Elbow detection

We are interested in the detection of elbows on somewhat tubular regions, such as the canal of an ear impression. 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 \( S_{ROI} \subset S \) is scanned with planes \( h_t \) oriented along its skeleton \( s(t) \): \( t \in [0; 1] \), to generate a cross-sectional profile \( \{ S_{ROI} \cap h_t, \forall t \in [0; 1] \} \) (Fig. 8). The profile contours, thus, capture information about bends on the surface. Correspondences are then established among these contours along the radial direction, as shown in Fig. 8, to form radial contours \( c_i(t) \), \( i = 1, \ldots, n \). Radial contours are parameterized, and are used to identify a set of points \( Q \) of maximal curvature along these contours. However, not all points reliably represent an elbow, due to potential presence of the bumps. The set \( Q \) is, hence, pruned via a point rejection strategy for plane fitting similar to the deterministic RANSAC method [28] to increase the robustness of the elbow.

5.4. 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, \( p_s \) and \( p_e \) respectively, of the ridge, followed by the computation of a connecting geodesic. To ensure that the geodesic passes through the ridge, we minimize the cost of going from \( p_s \) to \( p_e \), where the cost is defined as a weighted combination of the geodesic distance
and the surface curvature:

$$k(\pi) = \int_0^1 \omega(\kappa(t))\pi(t)dt,$$

where $\pi(0) = p_s$ and $\pi(1) = p_e$ and $\omega(\kappa(t))$ is selected as a decreasing functional of curvature ($\kappa(t)$). 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 [29]. 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.

5.5. Application to ear impressions

The aforementioned generic algorithms are modified slightly to adapt them to ear impressions resulting in the CES [27]. For instance, the inter-tragal notch and the crus-side 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. 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 according to the algorithm described in Section 5.4.

Among the derived features, the canal-concha (or canal-crus) intersection is detected as an intersection of two geodesics, one running along the canal (a ridge), while the other traced from the concha peak (or crus).

Helix ridge is detected as the shortest curvature weighted geodesic between the helix and the shell boundary. Crus area is computed as the area enclosed by appropriated weighted geodesics run between the following pairs of feature points: (1) center-crus-valley–helix-ridge-bottom; (2) center-crus-valley–crus-ridge-bottom; and (3) the boundary contour. Crus-concha intersection is detected by analyzing the tangential profile of the intersection of the shell with the crus valley plane, see Figures 5 to 7.

6. Knowledge Base

Before describing the construction of rules, we first introduce their syntax and how the inference machine of the expert system is integrated into the CAD software. Combining the knowledge base, the feature detector, and the CAD software results in an automation framework defined by Eqs. (2) and (3) and depicted in Fig. 9. In this framework, the parts $R$, $O$, and $F$ are application specific. For different applications, e.g., for designing dental crowns or craniofacial implants, these blocks need to be replaced with appropriate rules and features. Depending on the use case, it may require some effort in terms of feature detection and rule definition. The framework architecture, including the CAD software and the inference machine (interpreter), is general enough to
handle various kinds of automated design tasks. For evaluation purposes, we restrict ourselves to the use case of customized hearing aid design.

In the current implementation, we have employed a CAD system, which is similar to Magics by Materialise [30] or Shell Designer by 3Shape [31]. The ES uses the internal command parsing architecture to interface with the CAD tools.

6.1. Knowledge base syntax and integration

The knowledge base is constructed through a procedural representation in the form of if-then-else rules as described in [32]. In contrast to a declarative representation, such a procedural knowledge representation allows the possibility of defining rules that match the manual HA design process and may easily be interpreted by the process experts. Despite its simplicity, this representation is sufficiently powerful to encode the knowledge for the shell shaping process [32].
The rules, in turn, are transcribed by a specifically developed scripting language with a context free grammar similar to PASCAL.

The main advantage of developing a new scripting language lies in its simple integration in an existing modeling software. It allows the inclusion of new functionalities as the need arises, while keeping the framework as simple as possible.

The developed language supports the standard data types, like booleans, integers, floats, strings and arrays of all types, as well as 3-D points, planes and matrices as special data types. For each data type, the standard calculation and comparison operators are made available, which allow vector and matrix based computations. The script language supports control structures, such as if-then-else blocks as well as for, while and repeat-until loops. Furthermore, it allows the definition of functions and procedures. The script parser and interpreter are implemented with the tools bison and flex \[33\]. The available functions are grouped into two classes. The first class encompasses the internal functions, which are defined in the syntax of the scripting language. The internal functions are divided further into five groups: guide, geometric, parameter, visualization and communication. They are indicated by a capital first letter.

The second class contains the logical functions, which are defined using the internal functions and are indicated by a lowercase first letter. They are called logical functions because they store the logic of the design process in the rules. In other words, the internal functions provide the architecture and basic functionality for the rule definition and integration into the modeling software and the logical functions store the process knowledge in the KB.

The expert system is integrated in the form of a guide, in two possible usage modes. In the semi-automatic mode, it guides an operator through the process steps and offers in each step the necessary CAD tools as well as a suggested operation. For example, if a process step requires a cutting plane, then the guide defines this cutting plane and sets up the necessary tools to cut the surface. The operator will then only need to verify the correctness of the plane before applying the operation. Once the cut is applied, the inference machine will fetch the next process step. In the fully automatic mode, the verification step is skipped and every operation is automatically applied.

We now demonstrate the use of the script language through illustrative examples starting with variable definitions. For the ease of use, the detected features are made available as variable constants like the TragusPoint. However, if they are not already detected, a detection routine is invoked on the fly at the backend before initializing the variable constant.

```
int num = 3 // constant
real x = 1.1 // constant
real y = 2.2 // constant
point p1 = TragusPoint // feature variable
point n1 = TragusPoint - AntiTragusPoint // feature variables
plane h1 = Plane( p1, n1 ) // function call
```

The next example demonstrates the usage of conditional statements and loops.
The if conditions implement a simple number and a string comparison. The while loop uses a simple number comparison, which is updated in the body of the loop.

```plaintext
if x == y then
    p1 = AntiTragusPoint
endif
point n2 = Point( 0, 0, 1 )
plane h2 = Plane( p1, n2 )
while x < y do
    h2 = MovePlane( h2, p1, x )
    x = x + 1
endwhile
if Option(1) == "CIC" then
    n2 = -n2
endif
```

The `MovePlane` is a geometric function, which modifies the given plane h2, by moving it along its normal by x mm towards the reference point p1. Parameter functions, like Option, usually specify a property for a design step. Here, the normal n2 is flipped for the CIC devices.

The logical function foo in the following example takes one input parameter of type string and returns a boolean value. The procedure foo2 has two parameters x and option and uses foo to either add or subtract x to a globally defined variable y.

```plaintext
func foo(string option) : bool
    if option == "AX" or option == "AY" or
        option <> "AZ" or option == "A1" and
        option == "A2" then
        return true
    else
        return false
    endif
endfunc

proc foo2( int x, string option )
    if foo( option ) then
        y = y + x
    else
        y = y - x
    endif
endproc
```

The guide functions provide the interface to the modeling and detailing functions of the design software. Detailing and modeling describe specific phases of the design process. For instance, the vent is integrated by specifying its starting and
ending points, and then invoking the vent integration tool of the CAD system. A simplified example is given below, which triggers the vent placement rule only when a vent option is available. In the opening part, it sets up the **Modeling** CAD environment and displays the name of the guide step, i.e., **Vent placement**. In the body of this rule, just one logical function `computeVentPoints( v1, v2 )` is called, which computes the position of the vent end points. This information is delivered back to the CAD software by the closing step, which automatically invokes the integration of the vent into the shell.

```plaintext
if has_vent() == true then
    OpenGuideStep(Modeling, "Vent placement")
    point v1, v2
    computeVentPoints( v1, v2 ) // logical function
    CloseGuideStep(VentPlacement, v1, v2)
endif
```

Another example for a guide step with focus on the cutting and rounding tools is the removal of the helix part for the smaller device types:

```plaintext
if is_itc() == true or is_cic() == true then
    OpenGuideStep(Detailing, "Helix removal")
    plane h1 = CrusValleyPlane // feature
    point p1 = (TragusPoint + AntiTragusPoint) * 0.5
    plane h2 = MovePlane( h1, p1, 1.0 )
    point ref = TragusPoint // feature
    int level = 1
    CloseGuideStep(Round, h1, ref, level)
endif
```

This simplified example uses a plane as an input for the rounding tool, which is initialized using a feature plane. To make the selection unambiguous, a reference point is provided together with a rounding level.

**Visualization** functions are used to set up the working environment for a user. This includes common functionalities like toggling the transparency of an impression in cases where components are placed.

```plaintext
ShowTransparent( true )
SetTool( "Round", level )
```

The last set of functions allows the expert system to **communicate** with the user and to provide information about the progress, success or failure of a process step or the detailing or modeling procedure.

```plaintext
Info("text")
Warning("problem")
```

### 6.2. Construction of Rules

We adopted an iterative approach for constructing the KB as described in [32]. First, the written work instructions were directly translated into rules and
the resulting prototype KB was evaluated with several sample cases. With the help of the expert operators, the gaps in the written instructions were identified and filled. 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 till the first complete KB was acquired.

Nevertheless, the performance of the eventual KB still lags behind an expert operator. 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 accommodate such variation. To improve this, we again consulted our experts and fine tuned the KB further. The current KB consists of approximately 60 guiding rules and approximately 70 logical rules.

7. Evaluation

We now carry out the validation of the proposed framework. Three kinds of validations are considered. First, we focus on the evaluation of detected features. Second, we evaluate the performance of the early iterations of our KB. Third, we analyze the quality of the HAs designed through our system, via comparison with the manual approach. The evaluation was carried out on a standard PC consisting of an Intel Core 2 Duo CPU and 2 GB of RAM.

7.1. Feature detection evaluation

An extensive evaluation of the feature detector was carried out in [27]. Over a dataset of 198 impressions, statistical evaluation of the point and plane features was carried out relative to the ground truth, which was manually annotated by an expert. The curve features, on the other hand, were evaluated qualitatively by the experts, due to the absence of corresponding ground truth data. The similarity $\delta_p$ of the point features with the ground truth was computed as the Euclidean distance between them.

<table>
<thead>
<tr>
<th>Measure</th>
<th>$\mu$</th>
<th>$\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta_p$ (point distance in mm)</td>
<td>1.98</td>
<td>2.05</td>
</tr>
<tr>
<td>$\delta_{or}$ (plane orientation in degrees)</td>
<td>8.1</td>
<td>16.3</td>
</tr>
<tr>
<td>$\delta_{loc}$ (plane location in mm)</td>
<td>1.0</td>
<td>1.21</td>
</tr>
<tr>
<td>$\delta_{sen}$ (sensitivity)</td>
<td>0.83</td>
<td>-</td>
</tr>
<tr>
<td>$\delta_{spe}$ (specificity)</td>
<td>0.93</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 1: Results of the feature detection evaluation for the different measures. $\mu$ denotes the mean value and $\sigma$ the corresponding standard deviation.

For plane features, both the orientation and the location were considered. The orientation $\delta_{or}$ was compared via the angle between the plane normals. Deviation of plane locations $\delta_{loc}$ 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 $\delta_{\text{sen}}$ and specificity $\delta_{\text{spe}}$, see Table 1.

For the interpretation of the results, experts were asked to provide reasonable error tolerances for various features (3mm for location, and 15° for orientation). The error tolerances were then used to define two additional performance measures. The detection rate $\Delta_{\text{det}}$ was computed as the percentage of test cases for which an algorithm successfully detected the corresponding feature. The tolerance rate $\Delta_{\text{tol}}$ was computed 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 2.

<table>
<thead>
<tr>
<th>Performance measure</th>
<th>Feature type</th>
<th>Rate in %</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta_{\text{det}}$ (detection)</td>
<td>3-D point</td>
<td>98.6</td>
</tr>
<tr>
<td>3-D plane</td>
<td>98.8</td>
<td></td>
</tr>
<tr>
<td>$\Delta_{\text{tol}}$ (tolerance)</td>
<td>3-D point</td>
<td>87.0</td>
</tr>
<tr>
<td>3-D plane</td>
<td>82.0</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Results of detection and tolerance rate.

7.2. Early evaluations of the knowledge base

For the early evaluations of the knowledge base, a simple performance rating was used to acquire feedback from the test users by running the automation framework in the semi-automatic mode. All users were asked to rate each guide rule according to: 0 – unusable, 1 – usable with modifications, 2 – acceptable without modifications and 3 – perfect. Rating value 3 was introduced to cover the tendency of the expert designers to apply minimal modifications.

39 hearing aid orders with different constraints on the size, venting options and amplification were employed as test samples. In total, there were six expert users. Their ratings are analyzed in Table 3. The overall quality is very promising. On average, each guide rule suggested an acceptable solution for each step. However, for full automation, it is necessary to reach at least a rating of 2 in each step. In this first evaluation, automation was achieved for about 14% of the test cases (every applied rule rated $\geq 2$). The inspection of the median rating value, $\tilde{m}$, indicates that 75% of the guide rules achieved a rating $\geq 2$, which also shows that most steps were acceptable by the experts. After the first iteration, some discrepancies were identified between the digitized rules and the operator expectations during feedback sessions with the experts. This was responsible for a lower level of quality, e.g., for optional cuts. On average most of the guide rules performed quite well, but needed improvement in terms of the standard deviation, e.g., ITE anti-helix filling. A minor subset of the rules (crus scooping, CIC measured cut and ITE measured cut) suffered from the fact that
Table 3: The table shows quality values given by expert operators using an early version of the automation framework to design a customized hearing aid. For the sake of clarity and space, similar rules are grouped together. \( \mu \) is the mean quality value of all six operators for a certain rule, \( \sigma \) the standard deviation and \( \tilde{m} \) the median.

<table>
<thead>
<tr>
<th>Rule name</th>
<th>( \mu (\pm \sigma) )</th>
<th>( \tilde{m} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>All rules</td>
<td>1.96 (0.95)</td>
<td>2</td>
</tr>
<tr>
<td>Canal tip cut</td>
<td>1.74 (0.98)</td>
<td>2</td>
</tr>
<tr>
<td>Excess material cut</td>
<td>2.21 (0.86)</td>
<td>2</td>
</tr>
<tr>
<td>ITC measured cut</td>
<td>2.10 (1.01)</td>
<td>2</td>
</tr>
<tr>
<td>ITE crus scooping</td>
<td>1.07 (1.14)</td>
<td>1</td>
</tr>
<tr>
<td>ITE measured cut</td>
<td>1.58 (0.93)</td>
<td>1</td>
</tr>
<tr>
<td>Optional cuts</td>
<td>1.68 (0.87)</td>
<td>1</td>
</tr>
<tr>
<td>Receiver hole placement</td>
<td>1.97 (1.06)</td>
<td>2</td>
</tr>
<tr>
<td>Waxguard cut</td>
<td>2.57 (0.86)</td>
<td>3</td>
</tr>
<tr>
<td>Canal thickening</td>
<td>1.72 (1.13)</td>
<td>2</td>
</tr>
<tr>
<td>CIC measured cut</td>
<td>1.14 (0.81)</td>
<td>1</td>
</tr>
<tr>
<td>ITC crus cut</td>
<td>2.45 (0.78)</td>
<td>3</td>
</tr>
<tr>
<td>ITE anti-helix filling</td>
<td>2.25 (1.10)</td>
<td>3</td>
</tr>
<tr>
<td>ITE cymba rounding</td>
<td>2.43 (0.77)</td>
<td>3</td>
</tr>
<tr>
<td>Labeling</td>
<td>1.86 (1.26)</td>
<td>2</td>
</tr>
<tr>
<td>Optional vent cuts</td>
<td>2.60 (0.76)</td>
<td>3</td>
</tr>
<tr>
<td>Vent placement</td>
<td>2.04 (0.93)</td>
<td>2</td>
</tr>
</tbody>
</table>

the training set, used during rule definition, did not cover the shell variability found in the validation set.

7.3. Final evaluation

For the final evaluation, we tried to avoid subjective data in contrast to the previous experiment. Here, we relied on the fact that our industry partner employed our framework for the manufacturing of HAs in the semi-automatic mode. The evaluation was organized in four parts. First, we analyzed the general performance of the automation framework, followed by a comparison of the consistency of design during manual and automated process. Third, the quality in terms of size was evaluated. Finally, the average time needed to design a customized HA was analyzed.

7.3.1. General performance

Using data from our industry partner, we measured the performance of the framework by counting how often a guide rule was accepted with or without modifications. To accommodate the fact that every user would model a different prosthesis, we integrated several thresholds into the measurements. For example, if the expert system places a cutting plane and the user modifies the plane only slightly before applying, then we count this as an acceptable solution. If the modification exceeds a threshold, we count it as unacceptable. In
agreement with the design experts, we used the thresholds defined in Table 4. The sample data consisted of 2072 manufactured HA orders, with 35940 guide rules. The results in Table 5 show that the level of automation achieved is good and similar for all device types. 71% of the rules were found very stable in their performance. 21% of them performed from very good to mediocre, with only 8% resulting in poor performance. Full automation was not achieved because:

<table>
<thead>
<tr>
<th>HA device type</th>
<th>Step acceptance rate in %</th>
<th>Fraction in %</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>71.38</td>
<td>100.00</td>
</tr>
<tr>
<td>ITC</td>
<td>69.94</td>
<td>45.03</td>
</tr>
<tr>
<td>ITE</td>
<td>73.60</td>
<td>36.82</td>
</tr>
<tr>
<td>CIC</td>
<td>70.44</td>
<td>18.15</td>
</tr>
</tbody>
</table>

Table 5: Guide rule performance analysis.

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 destroy 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 $O$, the designer has to adjust the guide rule in order to fulfill these specific constraints.

7.3.2. Consistency

To evaluate the consistency of the designs we used the following sample sets:

1. Sample set $S_{\text{Man}}$ consists of 25 manually designed HA shells. Each shell was designed three times by the same user, resulting in 75 surfaces: $S_{\text{Man}} = \{S_1, \ldots, S_{75}\}$.
2. Sample set $S_{\text{Aut}}$ consists of 25 semi-automatic designed HA shells. Each shell was designed three times by different users, also resulting in 75 shells.
Table 6: Results of the consistency analysis.

<table>
<thead>
<tr>
<th>Sample set</th>
<th>$\delta_{mc}$</th>
<th>$\delta_{se}$</th>
<th>$\delta_{\tilde{m}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_{Man}$</td>
<td>0.0089</td>
<td>0.1310</td>
<td>0.0675</td>
</tr>
<tr>
<td>$S_{Aut}$</td>
<td>0.0016</td>
<td>0.1161</td>
<td>0.0504</td>
</tr>
</tbody>
</table>

All surfaces were pre-aligned to a common coordinate space. The below described similarity measures were then computed for each pair of surfaces:

**Mesh coverage.** $\delta_{mc}$ is an indicator how well the shapes overlap. To compute $\delta_{mc}$, the number of corresponding points whose distance is below a threshold is divided by the number of corresponding points. The measure is almost 1 for shapes which are similar in size and shape, but do not match perfectly. We selected 2mm as the threshold in our experiments.

**Normalized sum of squared error.** $\delta_{se}$ is similar to $\delta_{mc}$, but instead of counting points, the squared error is accumulated.

**Median distance.** $\delta_{\tilde{m}}$ is defined as the median of the squared errors between the corresponding points.

Table 6 shows that automated processing leads to superior consistency, even though $S_{Man}$ were generated by the same user. It improved the mesh coverage by about 80%, $\delta_{se}$ by more than 10%, and $\delta_{\tilde{m}}$ by around 25%. Large increase in $\delta_{mc}$ is due to an much more consistent placement of the faceplate (Fig. 2) in the semi-automatic case. In the manual mode, the faceplate orientation varied substantially, among various designs by the same user for the same shells.

### 7.3.3. 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 five size related measures:

- The **mesh size** is defined as the inner surface area of the designed shell.
- The **shell height** is measured as the perpendicular distance from the bottom opening to the farthest point (Fig. 10(a)).
- The **body height** is measured as the perpendicular distance from the bottom opening to the lowest point on the aperture (Fig. 10(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.
(a) Shell height measured from top to bottom.
(b) Body height measured from bottom to the aperture.
(c) Bottom opening measured as sum of the length of the major axis of the bottom opening contour.

Figure 10: Visualization of the shell measures.

- The *bottom opening* is defined as the sum of two perpendicular lines at the bottom opening (Fig. 10(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 Fig. 2.

<table>
<thead>
<tr>
<th>Sample set</th>
<th>mesh size</th>
<th>shell height</th>
<th>canal length</th>
<th>body height</th>
<th>bottom opening</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_{\text{Man}}$</td>
<td>522.9 mm$^2$</td>
<td>17.1 mm</td>
<td>9.9 mm</td>
<td>8.1 mm</td>
<td>39.7 mm</td>
</tr>
<tr>
<td>$S_{\text{Aut}}$</td>
<td>461.5 mm$^2$</td>
<td>16.8 mm</td>
<td>9.7 mm</td>
<td>8.2 mm</td>
<td>35.0 mm</td>
</tr>
</tbody>
</table>

Table 7: Shell size analysis showing the average values for the given set.

Results of the shell measurements for the samples sets are shown in Table 7. Due to the small data set available, the significance of the results is low. However, Table 7 shows the advantages of our method. For all computed measures the size decreased. Most important for the visual appearance is the body height and the bottom opening, because both influence the visibility of the device.

### 7.3.4. Processing speed

We made a time study based on 90 samples. The sample set comprises a high variation in ear impressions as well as in the selected options. The results of the study are presented in Table 8. The results clearly indicate that it is beneficial to use the framework also in respect to the processing time.

<table>
<thead>
<tr>
<th>Automation</th>
<th>Total time</th>
<th>Time difference</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>652 seconds</td>
<td>-</td>
<td>100%</td>
</tr>
<tr>
<td>Yes</td>
<td>597 seconds</td>
<td>55 seconds</td>
<td>91.6%</td>
</tr>
</tbody>
</table>

Table 8: Processing time study comparing the manual and the semi-automated design process.
average, the processing time is reduced by approximately one minute per unit. This corresponds to a reduction of almost 10%.

8. Conclusions

We have presented an innovative framework to (semi-)automatically design customized medical prostheses. The method is based on integrating an expert system combined with powerful feature detection into a computer-aided-design system.

It is adaptable to various kinds of medical prostheses by replacing the rule base and the feature detection units. The developed knowledge base syntax is fairly simple, yet highly flexible to encode the complex rules of prostheses design including interfacing the computer-aided-design software and the feature detection.

In this work, the rules and features are defined for the purpose of customized hearing aid design. For different applications, e.g., for designing dental crowns these would need to be modified accordingly. The framework architecture, including CAD software and expert system, is general enough to handle various kinds of automated design tasks.

The feature detection design has been based on a generic to specific approach. It contains general algorithms, which detect peaks, concavities, bumps, ridges and elbows of surfaces. The general algorithms are then adapted to detect anatomical features achieving a detection rate of approximately 98%.

Evaluation of the automation framework was done rigorously in collaboration with an industrial manufacturer of the hearing aids. The evaluation results demonstrate the advantages of the proposed method. It achieved complete automation in 70% of the test cases, and resulted in highly consistent results, as measured by various comparison measures.

We also evaluated the mesh size, shell height, canal length, body height and shell opening. These criteria suggest that the prostheses designed by our framework are at least as good or better as the ones designed manually. Furthermore, our framework resulted in reducing the design time by a factor of 10%. In future, we plan to further fine tune the rules and anatomical features, to achieve an automation level in access of 95%, after which the design process will be carried out offline in fully automatic mode.

References


