Automated Virtual Treatment Planning in Orthodontics: Modeling and Algorithms

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Yokesh Kumar

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Adviser: Professor Ravi Janardan

July, 2012

Acknowledgements

I would like to thank all the individuals who contributed to this thesis in many different ways. My first thanks goes to my adviser, Professor Ravi Janardan for his constant support and guidance throughout the course of my graduate studies. I have learnt a great deal from him over these years, both consciously and unconsciously. His excellent research and writing abilities have always been an inspiration to me. Above all, he is very wise and receptive to the needs of his students that makes him a great adviser. I wish to emulate some of his qualities in my future endeavors.

Dr. Brent Larson at the School of Orthodontics at the University of Minnesota has been a second adviser to me. He started us on the problems that became this thesis and was always full of enthusiasm and great ideas for the project. Mike Marshall who was at GeoDigm Corporation has been very supportive providing a very sound and real-world perspective on my research over the years. I admire his clarity and vision in describing very practical research projects.

The department of Computer Science at the University of Minnesota provides a great environment for research and study. The department has been very generous in providing me with teaching and research assistantships over the years. Also, I would like to thank Dr. Brent Larson for facilitating research funds from the School of Dentistry. GeoDigm Corporation gave me an opportunity to do two summer internships which were very useful learning experiences. Eric Hockert at the Office for Technology Commercialization (OTC) at the University of Minnesota helped us with the licensing of the software developed as part of this thesis.

Many thanks for all my friends who made the time in graduate school very pleasant and lively. It would not have been the same without them. I would especially like to mention my thanks to fellow graduate students, Akash, Gyan, Leo, Hitesh, Pradeep

and Pawan for the interesting and sometimes rabbit-trail-like discussions on a variety of topics. Also, I would like express my gratitude to David and Susan who kept me well-fed during the past few years.

Words cannot express my deep gratitude towards my family, especially my mother who has always been there for me in every way. She made my education possible while facing many challenges and has always been a powerful force in my life. Also, I have been blessed with a wonderful and loving wife in Indu. She always did her best to encourage me, and support me through tough situations, especially in the final years of my studies. Looking back, I can appreciate her patience and kindness as she quietly went about helping me while I spent late nights in front of a computer screen.

Dedication

This work is dedicated to all my teachers.

Abstract

Computer-based virtual treatment planning and simulation has become increasingly important in orthodontics due to its potential to lower costs and provide better treatment outcomes. However, its clinical use is currently limited by the need for significant human intervention in segmenting the dental arch (a 3-dimensional surface mesh) into individual tooth objects and then (re)aligning these tooth objects on the arch to satisfy prescribed treatment criteria. The goals of this thesis are to develop modeling techniques, computational algorithms, and associated software to automate key components of the treatment process, including dental arch segmentation, feature identification, and tooth alignment, thereby making the advantages of virtual treatment planning and simulation available to the vast majority of orthodontic patients. Towards this end, this thesis makes the following contributions:

First, an algorithm is designed to segment a dental arch into individual tooth objects (sub-meshes) that can be manipulated downstream in the treatment planning and simulation pipeline. The algorithm is largely automated and requires human intervention in very difficult or unusual cases only. The algorithm avoids the limitations of known approaches to dental segmentation by dividing the segmentation task into two key subtasks—separation of the gums from the teeth and separation of the teeth from one another—and also incorporates a new technique to repair defects in the gumline caused by noise in the data.

Second, algorithms are developed to automatically identify suitable features on the surfaces of individual tooth objects for use in the alignment stage. These include intrinsic features such as cusps, incisal edges, grooves, marginal ridges, and the occlusal surface boundary, as well as derived features such as the archform and occlusal plane. A key technique underlying some of these algorithms is the identification and clustering of vertices of high curvature on certain planar cross-sections of the tooth objects and stitching these clusters together to extract the features of interest.

Third, an algorithm is developed to automatically align the tooth objects on two opposing arches so that the best possible intra-arch and inter-arch occlusion (i.e., contact relationship) of teeth is achieved, while respecting certain natural dental constraints.

The alignment process is modeled as a simulation-based optimization of certain configurations of constraints defined with respect to the tooth features identified previously. The simulation is based on a spring-mass model where the teeth gradually move to their final positions under the influence of forces exerted by (hypothetical) springs attached to the features.

Finally, all of the algorithms developed in this thesis have been implemented and incorporated into a comprehensive software tool. This tool has been used by dental practitioners to evaluate and validate the algorithms on clinical data. These experimental studies have demonstrated the efficacy of the algorithms for orthodontic treatment planning.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Virtual 3-dimensional treatment planning and simulation has recently become possible in orthodontics, but its application in routine patient care has been limited because of the need for significant human intervention to segment the dental arch (modeled as a 3-dimensional surface mesh) into individual tooth objects and to (re)align these tooth objects on the arch according to user-prescribed criteria. This is a domain that can benefit considerably from the interdisciplinary development of modeling techniques and algorithms to automate the different stages of treatment planning and simulation.

Historically, simulation of orthodontic treatment has been limited to set-ups done in the dental laboratory by carefully sawing individual teeth apart from a plaster tooth model and then resetting them so that the desired alignment of the teeth on each arch and occlusion (i.e., matching) of the upper and lower arches is achieved. Orthodontic treatment simulation has been rapidly changing from 2D to 3D and, at the same time, from analog to digital. This rapid evolution has enabled clinicians to treat orthodontic patients virtually, as part of the simulation process, to ensure that the selected treatment plan can reasonably be expected to result in an optimal outcome. Indeed, in acknowledgement of this paradigm shift, the American Board of Orthodontics (ABO) has recently started accepting digital models and treatment plans processed on software from approved vendors for their board exams [46].

Virtual treatment simulation has been incorporated as part of new orthodontic treatment techniques such as Invisalign® [25] and SureSmile® [52]. Both of these techniques require the use of proprietary "digital laboratories" to complete the treatment simulation,

and the cost of these services is significant since the digital laboratories utilize the time of human technicians for the manual segmentation and alignment of teeth. Treatment simulation is also possible using software provided by vendors supplying digital tooth models such as *e*model®8.5 [53].

Unfortunately, the digital tools currently available are very cumbersome and time-consuming for the orthodontist to use in routine clinical care and, therefore, treatment simulation is only used on the most complex cases. In fact, a recent survey [27] indicates that actively practicing orthodontists are using simulations, or set-ups, only 2.5% of the time and yet 18% of patients now have digital models made as part of the information gathering process prior to treatment—and the use of digital models is accelerating. This discrepancy between the use of digital models and the number of patients that can actually benefit from treatment simulation indicates that there is a large (and growing) unmet need for providing a more efficient and cost-effective method for virtual treatment planning and simulation. (To put this in perspective, there are upwards of two million new orthodontic patients each year in the U.S. alone and nearly all of them could benefit from treatment simulation prior to actual treatment.)

1.1 Goals and scope of the thesis

The goals of this thesis are to develop modeling techniques, computational algorithms, and associated software to automate key components of the virtual treatment planning and simulation process, thereby making its advantages available to the vast majority of orthodontic patients.

Figure 1.1 depicts our view of how the virtual treatment planning and simulation process for orthodontics would work. The input is a 3-dimensional (3D) triangular surface mesh of each dental arch (teeth plus gums), obtained typically by laser-scanning a plaster model of dental impressions taken from the patient. The output is a treatment plan and associated simulation for achieving the desired outcome. The four steps, shown as solid boxes, facilitate the generation of this treatment plan and simulation:

1. **Segmentation:** This is the task of separating the teeth on each arch (mesh) from the surrounding gums and from each other into individual tooth objects (sub-meshes) that can be further manipulated downstream.

2. **Feature identification:** This is the task of identifying useful features or landmarks, such as cusps, marginal ridges, and grooves, on the tooth objects produced by segmentation (see Chapter 3 for a definition of these features). These features are useful in the alignment stage and also for classifying teeth by type, detecting missing teeth, etc.

3. **Alignment:** This is the task of (re)aligning the tooth objects on each arch and then aligning the two arches relative to each other so that user-prescribed criteria involving spacing, orientations, contacts between arches, etc. are satisfied.

4. **Placement:** This is the task of placing orthodontic devices (e.g., brackets and wires) at suitably chosen locations on the tooth objects, so that appropriate forces are exerted on the teeth to achieve the desired alignment over a period of time.

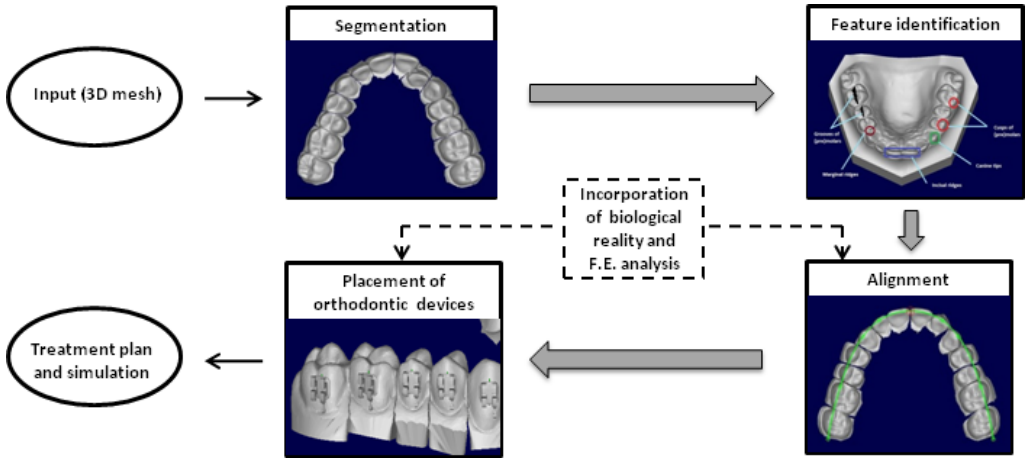


Figure 1.1: Orthodontic treatment planning and simulation process. *(All figures in the thesis are best viewed in color.)*

Ideally, one would like to automate these four steps so that, on every input dataset, the process in Figure 1.1 runs to completion without user intervention. However, this is not realistic given the complexity and diversity of patient dental data. Thus, in this thesis, we adopt the more realistic goal that the process be automatic for most input datasets (say, 90%), but may require clinician input on a small number of difficult or unusual cases. Our notion of automation should be viewed in this context. Given

the volume of orthodontic cases treated annually, this should still result in substantial savings in time and cost.

In practice, the desired final alignment is realized not all at once but through a gradual sequence of incremental changes over a period of months or years as the maxilla and mandible (i.e., upper and lower jawbones) respond to the pressure applied by the orthodontic device and reshape themselves, via a biological process called bone remodeling. This requires incorporating finite-element modeling in the alignment and placement algorithms to account for the forces exerted on individual teeth (shown as a dashed box in Figure 1.1).

The scope of this thesis is the automation of the first three stages of virtual treatment planning, i.e., segmentation, feature identification and alignment. (Placement and incorporation of biological reality, such as bone remodeling, is beyond our scope.) Our experience has shown that the difficulty of automating these stages increases in the subsequent stages, i.e., the alignment step is more difficult than feature identification step, which in turn is more difficult than the segmentation step. There are several factors that are responsible for this relative complexity. First, the proper automation of the alignment and feature identification steps, as shown in Figure 1.1, depends on the earlier steps to provide reliable, good quality input. For example, well-formed, correctly-segmented teeth are important for accurate feature identification, and correctly-identified features can drive the alignment step much more accurately. Second, consider the level of variation among the inputs for the different steps. Segmentation is only concerned with variations in the representation of the gumline (gum-teeth separation) and the interproximal contact regions (tooth-tooth separation). However, the feature identification step must work with wide variation in the detail on occlusal surfaces (affecting the identification of grooves, cusps, cusp ridges, etc.), in addition to variations in the interproximal regions (affecting identification of marginal ridges). Third, in many cases, the range of correct alignments and features that occur in nature is broader than the range of correct segmentations. This variation requires that the automated process for the alignment and the feature identification steps recognize a large set of correct solutions, which in turn increases the knowledge base that must be "built" into the algorithms. Finally, it is natural to expect the alignment step to be more complex than the other two steps, as it has to deal simultaneously with both the upper and the lower arches unlike the other

two steps which deal with the arches independently.

1.2 Contributions of the thesis

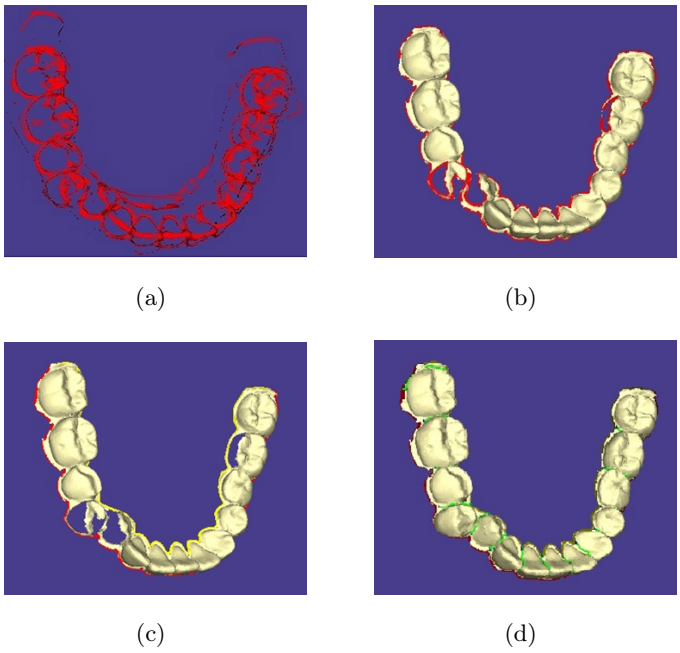
This section provides a brief description of the contributions made in this thesis. These include new algorithms for segmentation, feature identification and alignment, the implementation of these algorithms within a comprehensive software tool, and the evaluation and validation of these algorithms on real-world datasets. Brief descriptions of the key ideas underlying our approach to these problems are provided below and elaborated upon in subsequent chapters.

1.2.1 Automatic segmentation of teeth

We present an algorithm to automatically separate the dental arch (a surface mesh consisting of teeth plus gums) into individual tooth objects, with human intervention required only in the most difficult cases. Our approach consists of two tasks:

1. **Gum-Teeth separation:** This separates the mesh into two sub-meshes—one consisting of teeth and the other of the gums. We start by finding rough feature regions constituting the gumline around each tooth, based on surface curvatures (Figure 1.2(a)). In the presence of scanning errors and imperfections in the dental impressions, the gumline is often not well-represented, which leads to holes in the feature regions. Our *defective gumline repair* process identifies portions of the detected gumline that do not correspond to a natural gumline (Figure 1.2(b), in red) and repairs it to be closer to ground truth.

2. **Tooth-Tooth separation:** This separates the teeth mesh identified in Task 1 into individual tooth objects. The interproximal points on the lingual (inner) and labial (side) (red and yellow curves in Figure 1.2(c)) and interstitial points (corner points of high negative curvature between teeth) are identified and paired using a matching algorithm. Given such a matching of points from the lingual and labial sides, we find an accurate separating boundary between two teeth as the shortest path on the mesh between the matched pair of interstitial points (separators are shown as green curves in Figure 1.2(d)).



(a) (b)

(c) (d)

Figure 1.2: Steps in segmentation. (a, b) Gum-teeth separation. (c, d) Tooth-tooth separation.

The segmentation algorithm and experimental results are reported in [33]. A study evaluating the software implementation of the segmentation algorithm on real-world datasets is reported in [43, 44].

1.2.2 Feature identification

After the arch has been segmented into tooth objects and before alignment can begin, it is necessary to identify features on the surface of the tooth objects that can help with the alignment. A dental arch consists of a set of intrinsic features on tooth surfaces such as cusps, marginal ridges, occlusal surface boundary, grooves and incisal edges (Figure 1.3). Besides the intrinsic features, there are also certain other useful features that are derived from these intrinsic features, such as the archform and occlusal plane. (Chapter 3 gives detailed definitions of these features.)

We find cusps as the points representing the local minima of a suitably-chosen height

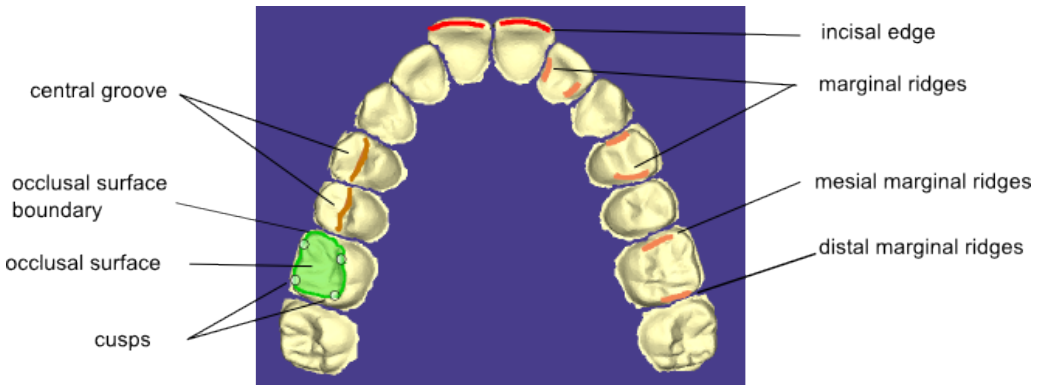
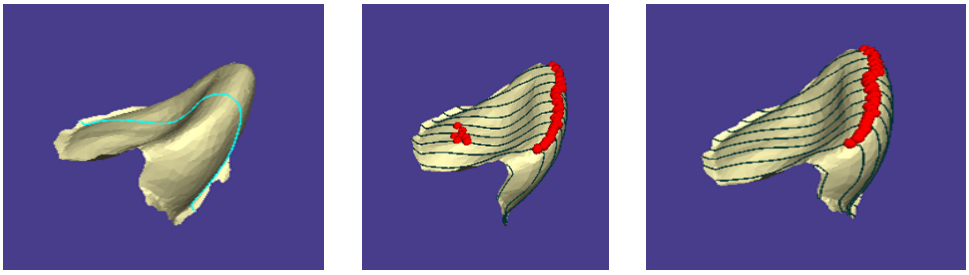


Figure 1.3: Intrinsic features on tooth surface.

function used within a watershed-based mesh segmentation technique [39]. We present a general technique to identify other intrinsic features such as the incisal edges (resp. grooves) as clusters of high positive (resp. negative) 2D curvatures regions on the set of planar cross-sections of teeth in the mesiodistal (along the archform) or the buccolingual (orthogonal to archform) direction of the tooth in question (Figure 1.4).

Our feature identification techniques and initial experimental results are reported in [32]. An expanded version of [32] that includes additional quantitative validation studies on clinical data is reported in [30].



(a) (b) (c)

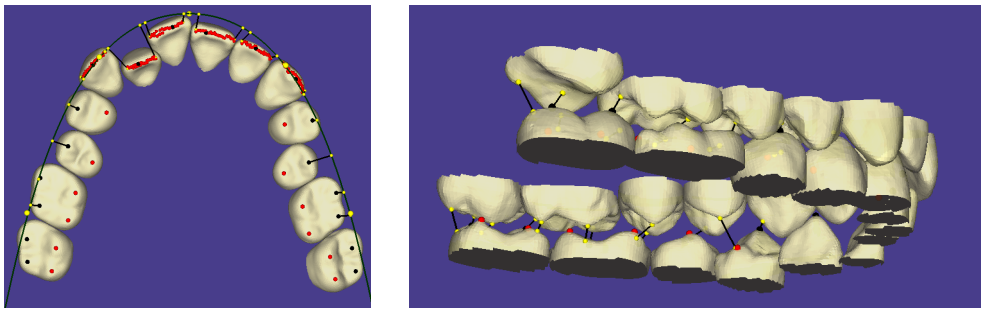
Figure 1.4: Identification of incisal edge. (a) A 2D buccolingual cross-section. (b) Clusters of high positive curvature regions on the cross-section curves. (c) Clustering to remove the noise regions.

1.2.3 Alignment

Given the individual tooth objects in a pair of upper and lower arches, our goal is to automatically compute the *optimal occlusion*, which we take to mean an arrangement of teeth that best satisfies the constraints of an ideal occlusion, as envisioned by an experienced practitioner. We define the quality of occlusion in a given dental model objectively, in terms of the relationship between certain intrinsic and derived features, e.g., the distance between the incisal edges and the archform (Figure 1.5(a)). Another example is the amount of separation between the cusps of molars and the surface of molars on opposing arches (Figure 1.5(b)). The alignment of teeth has an *intra-arch* (resp. *inter-arch*) component that aligns teeth with respect to other teeth on the same (resp. opposing) arch. However, our experience shows that the both intra-arch and inter-arch alignment of teeth must be considered simultaneously, rather than independently, to achieve a globally optimal solution.

We model the alignment process as a simulation-based optimization of certain constraints defined using tooth features. We use well-known conditions for correct dental occlusion in humans, along with the most commonly found occlusion errors in orthodontic cases, to establish a set of constraints between teeth. The simulation is based on a spring-mass model where the teeth gradually move to their final positions under the influence of forces exerted by (hypothetical) springs attached to the features. Figure 1.5 show examples of constraints acting on teeth. The simulation moves the teeth under the influence of the force exerted by the edges. At the end of simulation, the arrangement of teeth that best satisfies the constraints is reported as the best possible occlusion. Each step of the simulation checks the validity of the tooth movements across all the boundary conditions that are enforced. The virtual environment used for treatment planning poses additional boundary conditions on the repositioning of teeth, e.g., interpenetration of tooth meshes is not allowed at any stage of the realignment process.

The alignment algorithm and a detailed experimental evaluation of the associated software implementation are reported in [31].



(a) (b)

Figure 1.5: Examples of constraints edges (shown as black line segments) that drive the simulation achieve a better quality of occlusion among teeth. (a) The incisal edges (shown red) and cusps on teeth must align with the archform (shown green). (b) Cusps on an arch must contact with the occlusal surface of the opposing arch.

1.3 Organization of the thesis

The rest of the thesis is organized as follows. Chapter 2 describes our approach to the automatic segmentation of dental arches. Chapter 3 introduces the basic terminology for dental anatomy and features and describes our techniques to automatically identify important features for use in the alignment step. Chapter 4 describes in detail our techniques to solve the alignment problem. We summarize the thesis in Chapter 5 and conclude with directions for future work.

Chapter 2

Segmentation of teeth in dental models

2.1 Introduction

Virtual treatment planning in orthodontics starts with the creation of a 3D digital model of the patient's dental arches (Figure 2.1). Such a model is often extracted by laser-scanning a solid plaster model built from the patient's dental impressions. Once a 3D digital model is ready, it is imported into a computerized environment to facilitate visualization and evaluation of various treatment strategies.

One of the fundamental tasks in treatment planning is the separation of the teeth in the digital model into individual tooth objects. This task is called *segmentation*. This step is essential because much of orthodontic treatment is about manipulating and aligning individual teeth into the appropriate position in each arch and properly matching the upper and lower arches. After segmentation, the orthodontist can determine the new placement of teeth using various rules and esthetic guidelines. Once the desired final position of the teeth is defined in a virtual treatment plan, the orthodontist can then proceed to a visualization of proposed treatment outcome.

This chapter presents an algorithm to *automatically segment* a 3D triangular mesh, representing the full surface-scan of a dental arch, into individual tooth objects. Within the segmentation task, a problem of independent interest that we investigate is the automatic discrimination of the gum tissue from the teeth, which facilitates visualization

of the model. We emphasize that the starting point for our work is a 3D surface mesh, obtained as described above; this is one of the principal data modalities used in the clinic. Therefore, our segmentation problem is quite different from traditional image segmentation problems that work with 2D projections or images.

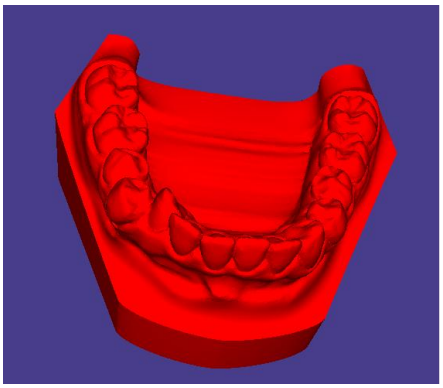


Figure 2.1: A scanned dental model. This is the input to the segmentation process.

2.1.1 Laser-scanned dental arch vs. CT-scan-based models

It is also possible to extract a 3D digital model (as a mesh) from CT-scan data of the jaw. However, laser-scanned models are more prevalent in current orthodontic practice for the following reasons:

- CT-scans are often at much lower resolution than laser-scanned models of plaster casts.
- Separating the lower and upper jaws (i.e., mandible and maxilla) in a CT-scan-based model is by itself a challenging task [34]. On the other hand, the plaster model for each jaw is created separately.
- The plaster models incorporate the gum tissue information and, hence, show the overall esthetics of the treatment.
- The creation of laser-scanned plaster models does not involve the harmful ionizing radiation associated with CT-scans.

It is worth noting that CT-scan-based models have the advantage that the absence of gums in these models exposes the roots of teeth, which allows the number and position of teeth to be estimated more accurately than in a laser-scanned model. Ideally, in the long term, it would be useful to integrate the information from both laser-scanned models (showing separate upper and lower jaws with gum-tissue) and CT-scan-based models (having root information) for improved treatment planning.

2.1.2 Organization of the chapter

Section 2.2 provides an overview of some concepts related to 3D meshes which will be useful in the rest of the thesis. Section 2.3 gives the related work and Section 2.4 describes our approach to dental segmentation. Section 2.5 describes a software tool implementing our segmentation algorithm and the experimental evaluation of our algorithm on real-world datasets.

2.2 Basic concepts of 3D mesh processing

In this section, we will introduce a few basic concepts regarding triangular meshes. We will discuss some useful properties of mesh elements and techniques to compute them.

2.2.1 Curvature estimation

For our purposes, a mesh is an unordered collection of 3D triangular faces and is an approximation to some closed smooth surface. (This is the well-known STL format.) The curvature of points on a piecewise flat triangle mesh is not well-defined because either the points are on a flat surface (no curvature) or at a corner (extremely high curvature). However, under the assumption that the mesh represents a smooth surface, we can estimate the curvature at each vertex using known methods.

Consider any point p on a smooth surface S. There is a unique normal vector to S at p denoted as N. Now consider any tangent vector t_p to the surface at p and the plane containing both N and t_p. The intersection of this plane and S is a curve whose second derivative at p gives the curvature at p with respect to tangent vector t_p. Note that generally there are infinitely many such tangent vectors at p, hence we focus on the tangent vectors (directions) among these that yield the minimum and maximum value

of curvatures. These are the *principal minimum* and *maximum* curvatures denoted as κ_{min} and κ_{max}. A classical result states that these directions are always orthogonal to each other [17]. A summary of many techniques for curvature estimation on a mesh can be found in [17].

We use the methods based on discrete estimation of principal curvatures κ_{min} and κ_{max} as described in [40]. We will briefly describe them below (detailed proofs can be found in [40]). The idea is to first estimate the *Gaussian* (κ_G) and *mean* (κ_M) curvatures at each vertex which can further be used to estimate principal curvatures. It is shown in [40] that for smooth surfaces, the 1-Neighborhood $N_1(v)$ of a vertex v, comprising of all the vertices that are direct neighbors of v on the mesh, is sufficient for estimating the curvatures. The Gaussian curvature estimate is defined as $\kappa_G = \kappa_{min} \cdot \kappa_{max}$ and the mean curvature estimate as $\kappa_M = (\kappa_{min} + \kappa_{max})/2$. The Gaussian and mean curvatures at a vertex p on the mesh are estimated as shown below. The area associated with a vertex x on the mesh surface is denoted as A_{mixed} and is computed as shown below.

Computing A_{mixed} of vertex x

$A_{mixed} = 0$
For each triangle T from $N_1(x)$

1. If T is non-obtuse
   - A_{mixed} += Voronoi region of x in T
2. Else
   (a) If the angle at x in T is obtuse
      - A_{mixed} += area(T)/2
   (b) Else
      - A_{mixed} += area(T)/4

Computing Gaussian curvature estimate at vertex x

$$\kappa_G(x) = (2\pi - \sum_{j=1}^{\#f} \theta_j)/A_{mixed}$$

where θ_j is the angle subtended by the j^{th} face at x and $\#f$ is the total number of faces incident on x.

Computing mean curvature estimate at vertex x

$$\kappa_M(x) = \frac{1}{4A_{mixed}} \| \sum_{x_j \in N_1(x)} (\cot \alpha_j + \cot \beta_j)(\mathbf{x} - \mathbf{x_j}) \|$$

where α_j and β_j are the angles opposite to the edge joining vertices x and x_j on the mesh.

In addition to the principal curvature values described above, we also need the directions in which they are observed at each vertex. Let us call this γ_{min} (resp. γ_{max}), corresponding to the principal minimum (resp. maximum) curvature. These directions are typically obtained as the eigenvectors of a tensor field at each vertex. The method we use is based on the techniques used in [5]. As mentioned at the beginning of this section, because the mesh surface is piecewise planar, there is no clear notion of curvature at a vertex of the mesh. However, with every edge, we have natural directions of curvature associated with it: the direction along the edge (*minimum* curvature) and across the edge (*maximum* curvature). This can be used to define a curvature tensor at each point on these edges. Such tensors of individual edges around a vertex can be used to define the collective tensor field at that vertex. This yields the following expression for the curvature tensor field in a surface region B around a vertex v [5]:

$$\mathcal{T}(v) = \frac{1}{|B|} \sum_{\text{edges } e} \beta(e) |e \cap B| \overline{e} \cdot \overline{e}^T,$$

where e is an edge in B, $|B|$ is the surface area of B on which $\mathcal{T}$ is evaluated, $|e \cap B|$ is the length of e in B, $\beta(e)$ is the signed angle between faces sharing e, and $\overline{e}$ is a unit vector along e. Since $\mathcal{T}$ is a symmetric matrix, it will have real eigenvalues and orthogonal eigenvectors. The two largest eigenvalues are the maximum and minimum curvatures and the corresponding eigenvectors are the directions of principal curvature directions (the eigenvectors corresponding to maximum and minimum eigenvalues are γ_{min} and γ_{max}, respectively).

2.2.2 Negative minima rule and feature lines

It has been demonstrated in various shape cognition studies [23] that humans pay more attention to the concave discontinuities between regions on 3D objects while visually decomposing a solid object into simpler ones. This suggests that, perceptually, humans tend to divide a solid shape along the lines of negative minimum curvature. This is called the *negative minima rule* in cognitive theory [22]. There has been some work, such as [36], on doing mesh segmentation directly using the negative minima rule. Typically these methods extract a set of mesh vertices of high curvature (according to some notion of curvature). These selected vertices are the starting point for defining boundaries between regions on the mesh. For example, see Figure 2.8(a) later. These high-curvature vertices form so-called *feature regions*. These feature regions are often thinned to form *feature lines*. This is currently the most successful approach to general mesh segmentation and is the one we will use as well.

2.3 Related work on dental segmentation

Although there are many studies of various properties of dental images and models, only a handful are directly relevant to the problem of automatic dental arch segmentation for orthodontic purposes. We will discuss some of these existing approaches to segmentation briefly in this section. We start with a discussion of 2D methods and their limitations and then proceed to 3D methods.

2.3.1 Range images and 2D methods

These methods work with plan-views of a 3D dental model. A recent approach based on this approach is described below. Motivated by the simplicity of the planar approach, we also designed a simple planar technique which is described in Section 2.3.1.

Range image approach

Kondo et. al. [29] proposed a solution based on plan-views of the 3D model to avoid the difficulties of working with mesh structures. Their method finds the boundary points between adjacent teeth from two different methods that work with plan-views and combines the results to compute the separating planes between teeth. As a preprocessing step, the base of the model is aligned parallel to the xy-plane and points on the mesh are mapped onto a uniform 2D grid. Each such mapped pixel (point) is assigned a range which is its height from the base. A plan-view range image from [29] is shown in Figure 2.2.

Roughly speaking, this method starts by fitting a third-order polynomial curve along the arch-line and sampling reference points on this curve. A depth profile of the heights of these reference points from the base is computed. Following this setup, sophisticated filters detect the points of local minima on this depth profile. We call the regions of the mesh that form natural boundaries between adjacent teeth or between teeth and gums as *valleys* in the mesh. The points of local minima on the depth profile belong to these valley regions and are used as *interstitial points* (these points belong to the natural boundary between two adjacent teeth). Although this technique works well for the molars and premolars, it often fails in the case of incisors as they overlap closely and their points of local minima are spaced very close to each other, making their detection with a filter difficult. This also fails in the presence of malocclusions because the curve may not always fit the arch-line.

Due to the problems mentioned above, a second complementary pass is made over a panoramic view of the model to identify the gumline and the vertical valley regions between the adjacent teeth. This looks for changes in surface normals and locates the boundary between anteriors easily because they form long valley regions (Figure 2.3). The final output with separating planes is shown in Figure 2.4. It however suffers due to

malocclusions (resulting in an imperfect reference curve) and scanning errors (resulting in an imperfect gumline and imperfect valleys) as shown in figures in [29] (also shown here in Figure 2.5). It is argued in [29] that these methods complement each other, but in practice it is seen that malocclusions usually can produce both types of errors. We will argue in Section 2.3.2 that such plan-view-based methods are not adequate for our purposes.

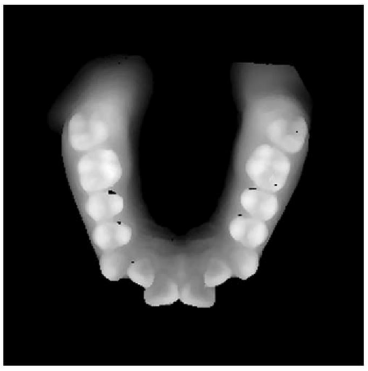


Figure 2.2: Plan-view range image of a 3D model. (From [29].)

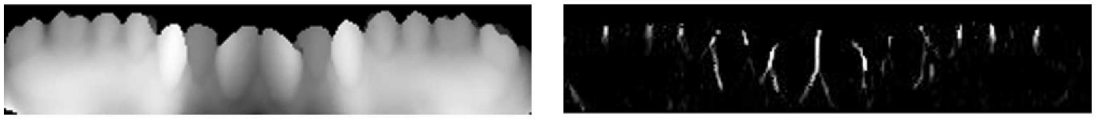


Figure 2.3: Left: Panoramic view of the model. Right: Interstitial points corresponding to valleys between teeth. (From [29].)

Our 2D solution

Motivated by the results in [29], we designed, as a first approach, a simple 2D method for segmentation that uses the contours of a cross-section of the dental model to compute the interstitial points between adjacent teeth. We will first describe the essential components of our solution, followed by some ways to implement them. These approaches try to detect the interstitial points (also called *embrasure points* in the dental field) between adjacent teeth from a contour that represents a cross-section of the model defined by a

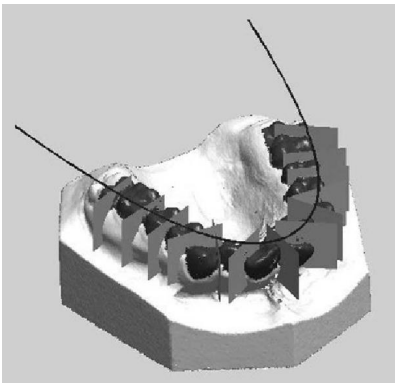


Figure 2.4: Final output of [29] with separating planes.

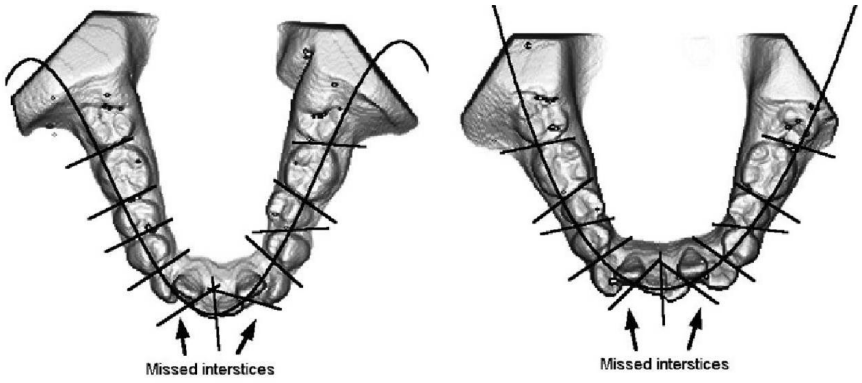


Figure 2.5: Results from [29] showing the missed boundary points due to malocclusion.

cutting plane. The main steps in this process are:

1. Compute a cutting plane at some suitable height as shown in Figure 2.6 (left).
2. Find the contour that is the intersection of this cutting plane and the model. Such a contour is shown in Figure 2.6 (right).
3. Each contour has a labial side (towards outside) and a lingual side (towards tongue). Detect sharp concave corners on both the labial and lingual sides of this contour.
4. Match the corner points on lingual and labial sides of the contour into pairs and draw separating planes guided by corner points and parallel to the z-axis as shown

in Figure 2.7.

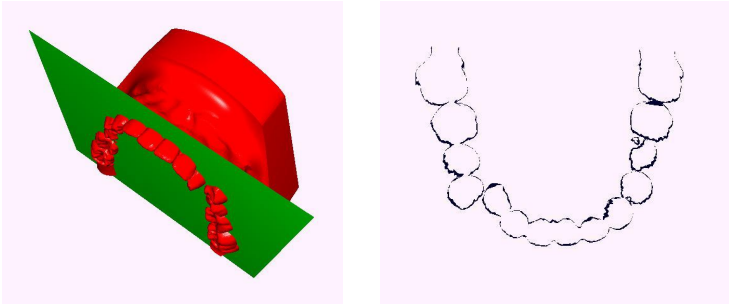


Figure 2.6: A 3D model with a cutting plane generating a cross-section view (left) and the contour of the cross section (right).

Figure 2.7: Output of segmentation with the separator planes.

Detailed description of steps

1. **Computing the cutting plane.** It is difficult to find a single optimal height for this plane that works for all dental models because of variations across dental models. Moreover, many times, a single plane may not be enough to generate complete cross-sections of all the teeth. For this purpose, we use a collection of two or three planes that cut the model at different heights.

2. **Find the contour of the cross-section.** The contour is extracted by identifying the faces of the mesh intersected by the cutting plane.

3. **Detect corners.** *Corners* are a useful concept in planar curve analysis, and are defined as vertices on a curve that provide information on boundaries between different structural regions of the curve. These are typically calculated as the vertices with dominating curvatures (sharp bends) in the sections of the curve (note that this curvature is different from the curvature of a vertex with respect to the surface of the mesh). We find embrasure points between adjacent teeth as concave corners on the contours. There are several algorithms to find such corners in planar curves, as discussed in [14]. One class of algorithms detects corners as points forming sharp angles with their local neighborhood. Another class tries to fit a triangle of some desired sharpness and area at each point and declares a point as a corner if such a triangle can be fit. There are cases that each of these algorithms miss and therefore we take the union of their results as the true output. This is a crucial step because this determines the number and location of separating planes. One can find corners on both the labial and lingual segments of the contour, but the lingual side tends to have more noise and, therefore, we only use the labial side corner points.

4. **Draw separator planes.** Each corner from Step 3 defines a family of separator planes that pass through that corner and are vertical. To pick the one with the appropriate orientation, we compute the lingual side corners and do a matching to pair up corners from both sides and use these to define the desired orientation. Another way to compute an orientation would be to use the corner's nearest point on a mid-line curve between the lingual and labial contours as the reference point. Figure 2.7 shows the separator planes constructed in this way. In an actual implementation, an interactive mode using the mouse can be used to further adjust the separator's size and orientation.

2.3.2 Limitations of 2D methods and need for 3D methods

As seen from the final output images, both the planar methods described above produce similar types of output and work well on relatively simple inputs. However, they are not suitable for our purposes due to the following reasons:

- **Severe malocclusions.** These lead to imperfect arch-lines and scanning errors

which make the identification of interstitial points difficult. Furthermore, due to malocclusions, the cross-sectional contours are sometimes not well-formed.

- **Noisy gums.** The gums near the cross-sections are often noisy. This leads to incomplete or incorrect contours and often the height of the cutting plane has to be adjusted manually to get a sufficiently large part of each tooth in the cross-section. This is especially true for the regions near molars.

- **Imperfect separation.** Since we are working with a planar view of the contour, we cannot automatically orient the separating planes properly in 3D. Instead, this boundary extraction needs to be done manually to improve the quality of separation (comparable to the separation obtained from 3D methods). This means that it is difficult to extract the exact boundary curve between two teeth using just planar separators as shown in Figures 2.5 and 2.7.

2.3.3 3D methods

Segmentation of 3D dental models has been studied in the past and revisited recently in the context of orthodontic treatment planning. We will first describe some traditional mesh segmentation algorithms followed by algorithms developed specifically for dental models.

Overview of mesh segmentation algorithms

Our problem is essentially a mesh segmentation problem, albeit with the nuances of the dental model to be taken into account. Mesh segmentation algorithms are often designed with some principal domain of application in mind (e.g., architectural, engineering, etc.) and may not perform well on other domains, as we briefly discuss below (more discussion can be found in [56, 57]). Good recent surveys of mesh segmentation approaches can be found in [4, 9] and references therein.

One segmentation algorithm partitions meshes into predefined primitives that approximate shapes like cylinders, boxes, spheres, etc. [8]. Although these are very useful in engineering domains, they are not suited for recognizing objects like teeth which are more complex and cannot be easily represented as a hierarchical collections of well-formed shapes. The use of library tooth objects as templates for recognition in real

models also fails because of missing and improperly developed/oriented teeth [29]. This is especially true for the dental models that we target as they come from patients with imperfect alignments.

Approaches based on "watershed" models [38, 39] have been used to identify features of interest (e.g., cuspids tips, cutting edges, etc.) in a dental model, which are useful for the subsequent alignment of teeth. A common problem with watershed models is that they often lead to over-segmentation, as also noted in [4]. This leads to detection of too many spurious teeth which must then be corrected.

The most useful segmentation techniques for our purposes are the ones that try to identify arbitrary shapes based on computing feature lines on the mesh [35, 50]. These methods, however, are too general to be accurate on dental models. Nevertheless, these approaches have some core components that can form the basis of a good dental segmentation algorithm. We will explain these and some existing work based on these principles below.

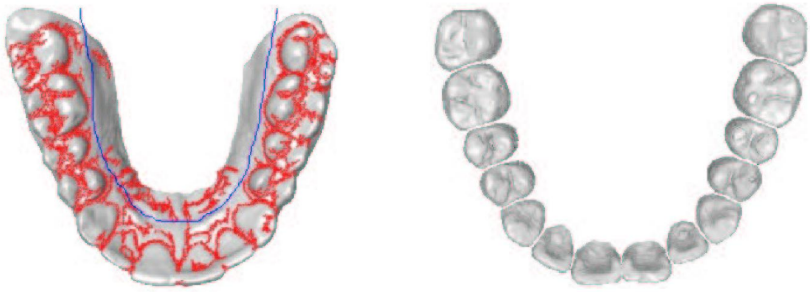
Segmentation of dental models

Feature regions are an important concept in mesh segmentation and related applications like shape recognition. We define feature regions as connected regions of high curvature points on the mesh; for example, in Figure 2.8(a), the red vertices form a connected set of feature regions. The mesh segmentation literature often uses the term *feature line* to represent a minimal representation (skeleton) of feature regions and can be computed using techniques given in [50]. The vertices in feature regions or lines are called *feature vertices*. A general algorithm that attempts to automatically detect arbitrary shapes (e.g., different types of teeth) in a mesh model works along the following lines:

1. Create feature lines (or regions) on the mesh that may separate dissimilar/distinct regions [50]. See Section 2.2.1 for details on curvature estimation.

2. Refine the feature lines to remove noise and spurious regions [17, 35]. For example, in Figure 2.8(a), we need to remove the noisy feature regions above the blue curve and on the molar ridges as both of these do not help in tooth segmentation.

3. Partition and cluster the individual regions or groups of regions of the mesh into meaningful shapes to identify target objects (e.g., tooth objects).



(a) (b)

Figure 2.8: (a) The feature regions (shown in red) as computed by [57]. (b) The final output of teeth segmentation [57].

The algorithms given in [56, 57] follow the above approach. First, they find feature lines using techniques from [50] using high curvature vertices of the mesh as shown in Figure 2.8(a). These form connected chains of feature regions and are a natural way to identify the different parts of a 3D object (see Section 2.2.2). After refining the feature regions to form a set of closed contours around the teeth regions (which involves some manual interaction), they separate the mesh into individual tooth objects as shown in Figure 2.8(a).

Limitations of existing methods

The methods described above perform well except that they may require a lot of user interaction on most models in orthodontics. This is because the gum regions in dental models contain a lot of noise in the form of isolated high curvature regions (e.g., sinus regions) which are not part of any interesting boundary on the mesh. Hence, these methods have to eliminate such noise. This requires that a user mark off regions that do not have any true feature lines (e.g., the blue line in Figure 2.8(a)). Although this can be automated on a few models, from our experience with dental models and from the discussion in [57], it appears that some manual interaction will still be required frequently.

Also, it is necessary to "close" the broken feature lines around each tooth. This is an expensive step and requires a user to manually create the missing portions of feature lines with brush strokes [56], or define two ends of a feature line (by clicking with a

mouse) that must be joined by (say) a shortest path [57]. All these are undesirable for the automated approach that we seek.

We have already discussed the disadvantages of range image methods. In a clinical environment, all of these methods translate into a lot of human time per patient and, hence, only the most complex cases are treated currently using digital methods. In the next section we present our approach to overcoming the limitations mentioned.

2.4 Our solution to dental segmentation

We now present our approach to identification of teeth in a 3D dental model (our solution can be classified as a 3D method similar to [56, 57]). This section will identify the specific problems that must be solved to achieve automatic segmentation and will discuss our approaches to the subproblems that arise.

2.4.1 Challenges in segmentation of dental models

Let us review some specific challenges associated with the segmentation of dental models that must be considered while designing a robust algorithm.

As mentioned in Section 2.3.3, feature regions can be used to identify the "interesting boundaries" in dental model, i.e., between the gums and teeth (gumline), and between adjacent teeth (interstitial points). The feature regions approach, however, requires reliable estimation of curvatures at vertices which is a difficult problem as discovered in our experiments. In particular, the curvature thresholds used to identify meaningful feature regions vary across different models even after normalization and, hence, cannot be hard-coded. Thus, we must refine and complete these feature regions [56, 57] using morphological operators [50] (*dilate, shrink, skeletonize,* etc.) or by extending feature regions along directions of principal curvatures.

There are several difficulties in relying completely on the above-mentioned techniques to refine and complete the interesting boundaries in dental models. First, morphological operators work well only when the gaps to be repaired are small (a few vertices wide) and computing the accurate directions of principal curvatures is difficult on noisy meshes [19]. Moreover, these operators are applied on the entire collection of feature regions in an automatic setting and, hence, can affect some portions of the already well-formed feature

regions when attempting to repair others. For example, in Figure 2.8(a), if we increase the curvature thresholds, or grow feature regions on the gumline at the molars, the feature regions near the incisors may become incorrect due to noise from gums in the proximity of interesting boundaries. In general, the gums (especially in the upper jaw) have many small patches of high curvature that must be eliminated.

Also, in practice, dental models obtained from laser scans sometimes have a few scanning errors due to malocclusions. Malocclusions of teeth lead to improper input to model generation and prevent the scanner from getting an accurate image of the boundary between teeth. In a few cases, the boundary between teeth and gums is also not prominent enough to be noticed by a curvature thresholding filter. Figure 2.10 shows some examples.

In addition to these issues, the previous methods also require considerable effort in removing noise due to molar ridges. In our approach these are avoided naturally by using a two-phase approach (Gum-Teeth separation followed by Tooth-Tooth separation; see Section 2.4.3). Also, it is observed in practice that most of the manual interaction is used to separate out the noise on gums rather than identifying interstitial points (there are at most 28 of the latter). Thus, it is important to automate as much as possible the process of differentiating gums and teeth in the dental model.

2.4.2 Our approach

Our approach to automatic segmentation is based on solving the following two problems:

1. **Gum-Teeth separation.** Classify the gums and teeth faces of the mesh correctly.
2. **Tooth-Tooth separation.** Identify the separating boundary curves between adjacent teeth.

We investigate these two problems separately, first solving the Gum-Teeth separation problem and then the Tooth-Tooth separation problem. In general, both problems have a similar flavor because the gum/teeth boundary (gumline) and teeth/teeth boundary (embrasure lines) are similar in structure in the sense that they are composed of valley regions on the mesh surface. However, the Tooth-Tooth separation problem demands more precision and involves working with different kinds of noise on the surface compared to the Gum-Teeth separation (the distinction will become clear in our discussions below).

It is also worth mentioning that the Gum-Teeth separation problem has other desirable applications in orthodontics. For example, it leads to better visualization of dental models with gums and teeth in their natural color. Also, when the orthodontist experiments with treatment plans, the dynamics of simulated tooth movement over time under the influence of forces are studied and this involves tracking collisions between adjacent teeth. Of course, we would like only the teeth to participate in the collision detection, not the gums. This is only possible if we know what parts of the mesh correspond to gums so that collisions involving gums can be ignored.

The major steps in the process of segmentation are given in the algorithm below. Steps **S1** to **S4** address the problem of separating gums regions from teeth and steps **S5** to **S7** address the problem of finding separating curves between adjacent teeth. We will explain the Gum-Teeth and Tooth-Tooth separation problems in Section 2.4.3 and Section 2.4.5 respectively.

Preprocessing. Remove faces near the base of model that do not provide any information useful for segmentation.

Gum-Teeth Separation

S1. Compute the approximate curvature estimates (κ_1 and κ_2) at the vertices of the mesh (See Section 2.2).

S2. Use an automatic or interactive curvature thresholding filter to obtain vertices of feature regions.

S3. Refine the feature regions and use the flood-fill approach to separate gums from teeth and form connected components of teeth (See Section 2.4.3).

S4. Extract and repair the gumline, and re-initialize the teeth and gum components of the mesh (See Section 2.4.4).

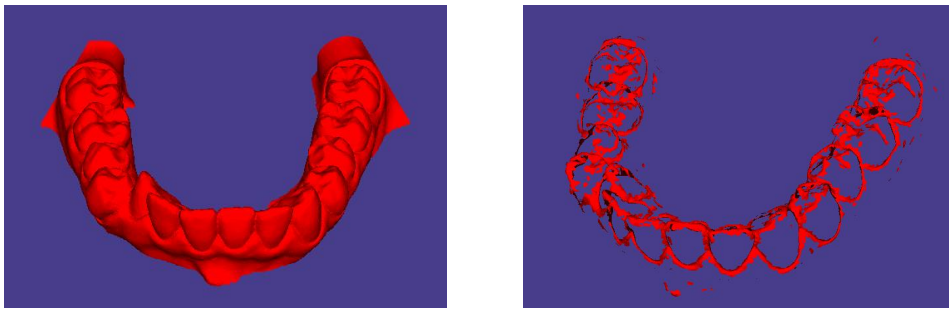
Tooth-Tooth Separation

S5. Extract the gumline (boundary curve) corresponding to each teeth component and split into lingual (tongue-side) and labial (outside) chains.

S6. Detect valid corners (interstitial sites on gumline) and match corresponding pairs from lingual and labial chains.

S7. Construct separating curves between adjacent teeth as the shortest path (along the mesh surface) between each pair of matched interstitial sites (Steps **S5** and **S6**).

S8. Delete the vertices on these curves from the mesh to separate the mesh into individual meshes for each tooth object. Also, patch the holes created in these meshes from the deletion of vertices by adding new faces to these meshes.



(a) (b)

Figure 2.9: (a) Model after preprocessing to remove faces that do not contribute to segmentation. (b) Feature regions obtained after thresholding on curvature.

2.4.3 Gum-teeth separation

After preprocessing to remove the bases (Figure 2.9(a)), we begin with the computation of curvature estimates at each vertex of the mesh according to the description given in Section 2.2. After this we filter out the vertices with curvature above a certain threshold to form feature regions. This threshold was found to vary in different models and, thus, a single value may not work satisfactorily for all the models. We believe that the sources of this variation in of curvature are the resolution of scanning, the size of models, and the quality of plaster casts. Figure 2.9(b) shows the feature regions after thresholding.

The issue of determining the correct curvature can be easily handled by providing the user with an intuitive slider-tool that enables setting the curvature threshold to a satisfactory value while observing the results visually. This is the only interactive part in our segmentation algorithm and it is far less time-consuming than the user interaction component of previous systems. From our studies on a range of models (26 laser-scanned models of plaster casts), we conclude that this broad range seen for the correct curvature threshold is an inherent property of the dental model and scanning process and, therefore, we must allow ourselves some flexibility in choosing this value.

Feature region computation and flood-fill

After applying the curvature thresholding filter, we obtain a set of vertices that represent interesting feature regions. Ideally, we would like them to form a closed contour around

connected component of teeth, defining the natural boundary between the teeth and gums, i.e. the gumline. However, due to scanning errors, sometimes the gumline is not represented prominently in the model (see Figure 2.10). When the gaps are small (a few vertices wide), the morphological operators from [50] can be used to "close" the contour. Another way to achieve this is to extend the existing feature regions by using directions of the principal curvature at selected vertices. We look at a small neighborhood of the high curvature vertices and include vertices that lie in the direction of the principal curvatures of current feature vertices. Figure 2.11 (a) shows an example of the directions of curvatures and a gap in the gumline, which is repaired as shown in Figure 2.11 (b). However, as discussed earlier, these are essentially global operators and, hence, in the process of closing a feature line in one region, we might introduce spurious features at other locations.

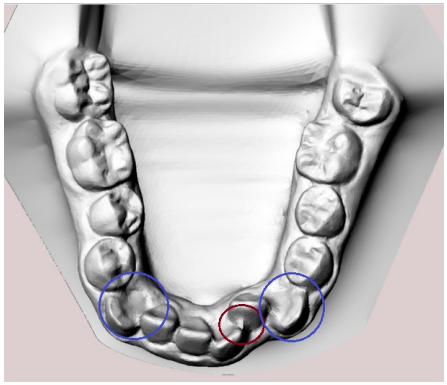
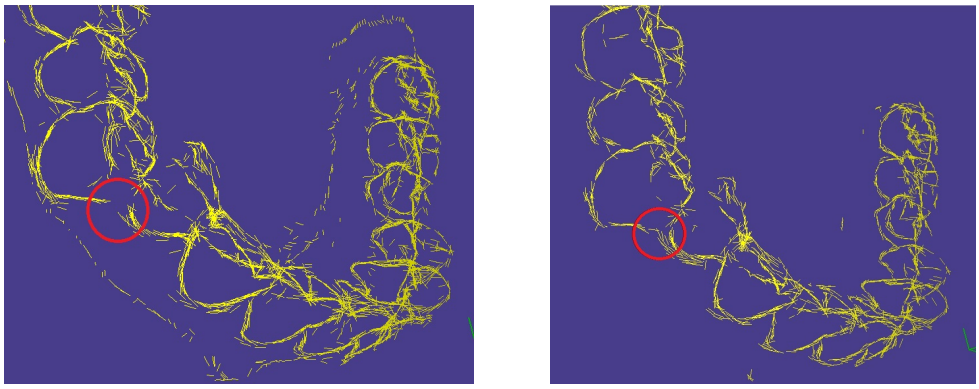


Figure 2.10: Example of scanning errors in models. The blue circles indicate the improper representation of portions of the gum/teeth boundary in model. Vertices in this region will not be found as high curvature feature regions and, hence, the connected gumline will be disconnected. Similarly, the red circle shows scanning errors in the presence of malocclusions.

The next step is to classify the vertices of the mesh as either belonging to the gums or the teeth. We call the feature regions *closed* if no vertex in the presumed teeth regions is connected with any vertex in the presumed gum region by a path on the mesh that avoids all feature vertices. In such a case, we perform a restricted breadth-first search (BFS) from a vertex known to be in the gums to classify all vertices visited as being in

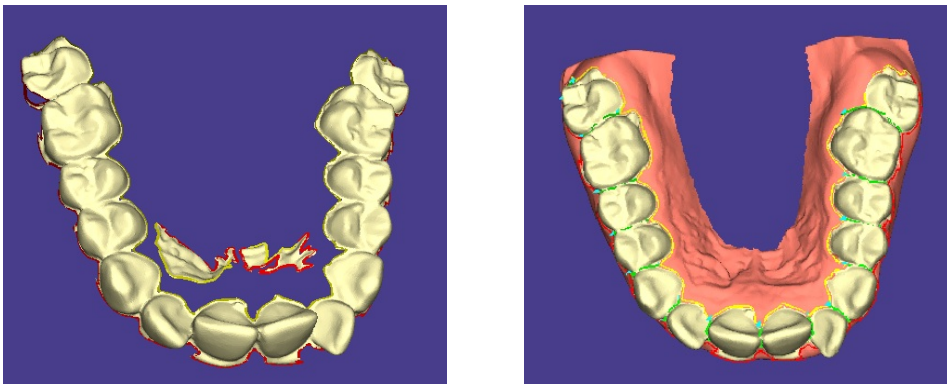


(a) (b)

Figure 2.11: (a) Directions of curvatures at feature vertices. The red circle shows a gap in the gumline. (b) Gap closed using morphological operators and growing feature regions in the direction of the principal curvatures.

the gum regions. This BFS is constrained to not cross any feature vertices (i.e., feature vertices are marked as already visited by BFS). As a result of this "flood-fill" BFS, we get connected regions representing gums and teeth. The noisy feature regions in the gum will lead to small connected components that are not classified as gums. To merge these noisy components with the gums automatically, we merge any noisy component with the gums if it has fewer vertices than a prescribed lower bound on the number of vertices in a tooth component (see Figure 2.12).

However, if the feature regions are not closed (i.e., the gumline has gaps), then the BFS starting from a gum vertex will infiltrate the teeth regions, thereby classifying some portions of teeth as gums. An example of such a flood-fill process is shown in Figure 2.13 (a) which shows missing portions of teeth classified as gums. If the gaps are small, this could be ameliorated by using a *bounded-width* flood-fill which advances the BFS at a vertex only if at least a small number of unvisited vertices exist in the neighborhood of that vertex (we have used a bound of at least 4 unvisited vertices in the 2-neighborhood of a vertex). Figure 2.14 shows the output of the flood-fill process applied to the model in Figure 2.9(a). Additionally, even in the case when BFS has infiltrated the teeth regions of the mesh, we still want to extract as much of true teeth regions as possible to facilitate the gumline repair process (see Section 2.4.4). To this end, we also use the vertices on



(a) (b)

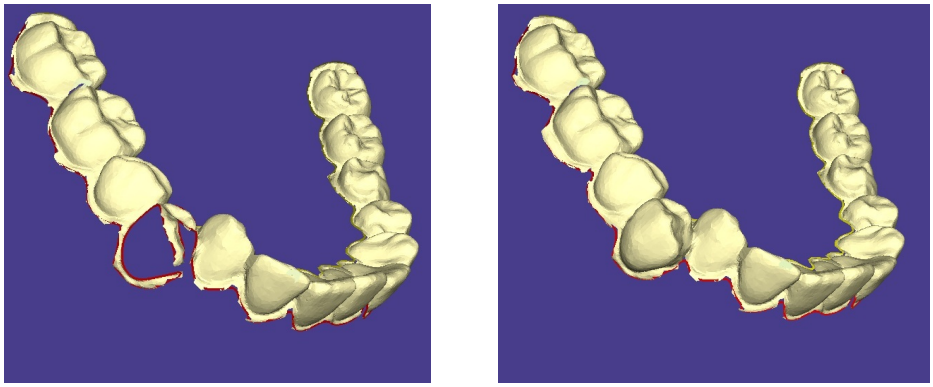
Figure 2.12: (a) Noisy high-curvature patches in upper jaw. (b) Final output after merging the noise components with the gums. The regions of the gums corresponding to the sinus leads naturally to high-curvature patches as shown on the left-hand side.

the ridges of anteriors and cusps of molars as feature vertices. This prevents the BFS from searching past these vertices after it has infiltrated the gaps in the gumline, thus preventing large portions of the teeth regions from being labeled erroneously as gums. These additional feature regions are obtained as the vertices with high positive curvature.

It is easy to see that flood-fill naturally avoids problems associated with the noise on molar ridges because if the gumline is properly closed, then the BFS will never reach these ridges.

2.4.4 Gumline repair

After the flood-fill stage separates teeth from gums, we extract a first approximation to the gumline as the curve through the boundary edges which are defined as edges that only occur in a single face of teeth components. Such a curve is shown in Figure 2.13 (a). Our goal is to repair such an initial curve to obtain the curve shown in Figure 2.13 (b) which is a much better approximation to the true gumline of the model. Our experiments show that this is a common problem in real-world dental models. Considering both these curves at the places where they differ, notice that the "good" gumline (right) seems much smoother in contrast to the "bad" one (left) and can be easily distinguished from the former (the terms "good" and "bad" will be made precise below). Thus, our working



(a) (b)

Figure 2.13: (a) An incorrect gumline (shown in red) after the flood-fill process. The missing portion of teeth are classified as gums (not shown). (b) After gumline repair and re-initializing teeth components.

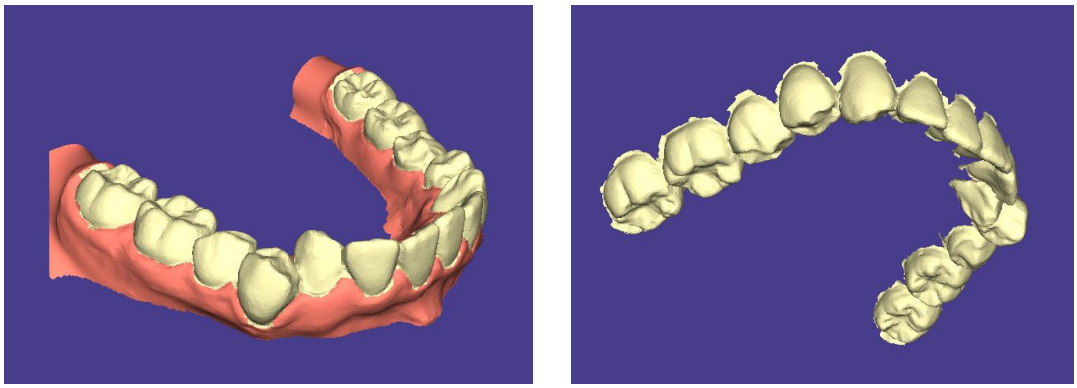


Figure 2.14: Result of flood-fill (given the closed feature regions) showing the vertices of mesh classified as teeth or gums.

hypothesis is that if a portion of a curve is "bad", then it can be easily distinguished by a relatively inexperienced user. We intend to translate this intuitive ability to discriminate between a natural and a defective gumline into an algorithm that can detect and repair such portions automatically.

Definitions

We begin the discussion of our approach to the gumline repair problem with a few definitions. As mentioned above, a curve is a closed path of n vertices on the mesh and is represented as $C = v_0, v_1, \ldots, v_{n-1}$ (v_0 is the first and last vertex on curve). We define an *interval* between two vertices v_i and v_j on the curve as the sub-path in C between these vertices and denote it as $[v_i, v_j] = v_i, v_{i+1}, \ldots, v_j$. Figure 2.15 shows such a curve C (in red) and vertices p, a, q, c and b (in order of their occurrence on C). Some intervals in this example are $[p, q], [a, b]$ and $[a, c]$.

For any interval $[v_i, v_j]$, we define the *quality* q_{ij} of v_i with respect to v_j as the ratio of the distance between v_i and v_j along the curve to the Euclidean distance between them. So, $q_{ij} = d_{ij}^C / d_{ij}$, where d_{ij}^C is the sum of the lengths of the edges between v_i and v_j on the curve and d_{ij} denotes the Euclidean distance between v_i and v_j.

With each vertex v_i of the curve, a score s_i is defined as follows:

$$s_i = \max_{(i+L) \leq k \leq (i+M)} \{q_{i(k \bmod n)}\},$$

where the terms L (lower bound) and M (upper bound) are constants that restrict the neighboring vertices that are searched to compute the score for any vertex. (The mod function allows wraparound on the curve.) This means that for any vertex, no vertex from within its L-neighborhood or beyond its M-neighborhood can be selected to compute its score. (Experiments showed that for our implementations values of $L = 10$ and $M = 100$ perform well over all models, and the mean value of n was about 350 across all models.) For each vertex v_i we also record the vertex $B(i)$ that generated its best score s_i (ties in scores are broken by selecting the vertex closest to v_i).

Consider a sub-path $v_i, v_{i+1}, \ldots, v_j$ on C denoted as P_{ij}. Let the shortest path from v_j to v_i be $S_{ji} = v_j, v_{k1}, v_{k2}, \ldots, v_i$. Concatenate S_{ji} with P_{ij} to obtain the closed circuit $Z_{ij} = v_{i+1}, \ldots, v_j, v_{k1}, v_{k2}, \ldots, v_i$. Clearly, Z_{ij} partitions the mesh vertices into three disjoint sets: vertices on Z_{ij}, vertices enclosed by circuit Z_{ij} (denoted as e_{ij}) and vertices

lying outside Z_{ij}. After flood-fill, each vertex of the mesh has been labeled as either a gum or a tooth vertex. If the majority of the vertices in the set e_{ij} belongs to gums (resp. teeth) we call the interval $[v_i, v_j]$ *Type-G* (resp. *Type-T*). Section 2.4.4 gives an algorithm to efficiently compute the set e_{ij} and thus decide if an interval is Type-T or Type-G. In Figure 2.15, $[p, q]$ is Type-T because the closed circuit would enclose a majority of teeth vertices. On the other hand, intervals $[a, c]$ and $[a, b]$ are Type-G.

Next we define an *invalid interval* as an interval that does not correspond to the natural gumline and must be corrected. In our example, the intervals $[p, q], [a, c], [a, b]$, etc., clearly correspond to such invalid intervals. Using the scores for all the vertices, we define a set of *candidate* invalid intervals

$$\mathcal{C} = \{[v_i, B(i)] \text{ , where } s_i > S_m \text{ and } B(i) \text{ generated } s_i\},$$

where S_m is a constant that serves as a minimum threshold on the score of a vertex for it to qualify as an invalid interval (we use $S_m = 2.5$, based on experimental validation).

Note that each vertex in the curve defines a unique interval. Given $\mathcal{C}$, we need to select a subset of its intervals to process because repairing all of them simultaneously may not be feasible or even correct. The repair will typically involve replacing the current sub-path between v_i and $B(i)$ in C with another path that better represents the natural gumline (we will discuss this problem below).

Some examples of candidate intervals in Figure 2.15 are $[p, q], [a, c], [c, b]$ and $[a, b]$. It is clear that we must prefer to repair $[a, b]$ among these intervals. Note that if we choose $[p, q]$ prior to $[a, b]$, we may not even get the correct gumline after repairing. Also, repairing $[a, c]$ and $[c, b]$ separately instead of $[a, b]$ will not result in a gumline as good as the one obtained from repairing $[a, b]$. This motivates the following problem:

Invalid Interval Maximization (IIM). Given a candidate set, $\mathcal{C}$, of invalid intervals to be repaired, compute a subset $\mathcal{I} \subseteq \mathcal{C}$ of invalid intervals that are pairwise non-overlapping and whose total length is maximum. (The length of an interval $[v_i, v_j]$ is defined as the number of vertices on the curve that lie between v_i and v_j.)

In what follows, we describe in detail our solution to the gumline repair problem. This includes the key subproblem of classifying intervals as Type-T or Type-G, solving Problem **IIM**, and repairing the set $\mathcal{I}$ thus found.

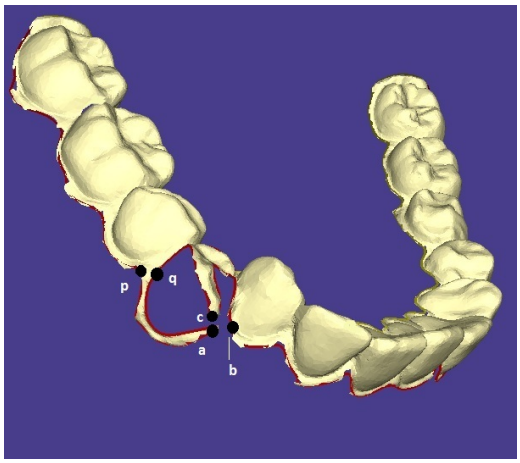


Figure 2.15: The curve from Figure 2.13 shown with a few vertices.

Deciding the type of interval

We orient the edges of the paths as going away from v_i (resp. v_j) in P_{ij} (resp. S_{ji}) to get a natural predecessor/successor relation among the vertices on paths (Figure 2.16 shows the orientation of the edges).

Given paths P_{ij} and S_{ji} on the mesh, our goal is to find the set e_{ij}, i.e., the vertices of the mesh that are completely enclosed by Z_{ij} and compute the majority of the type of vertices in e_{ij}. We solve this problem in two stages, as follows:

Decompose into sub-circuits. We first decompose the given Z_{ij} into closed edge-disjoint sub-circuits $Z_{ij}^{(1)}, Z_{ij}^{(2)}, \ldots$ such that are not self-crossing. An example of such a decomposition is shown in Figure 2.16 (b). The path P_{ij} (resp. S_{ji}) starts at v_i (resp. v_j) and ends at v_j (resp. v_i).

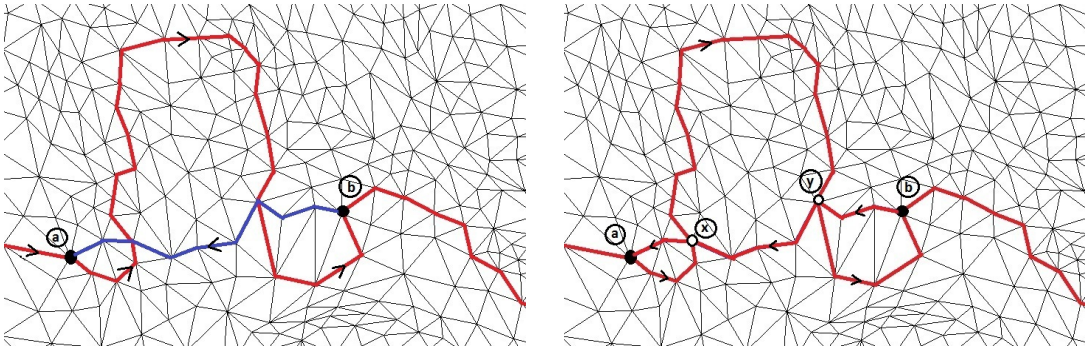
An algorithm to achieve the decomposition of Z_{ij} is given below. We terminate when all the vertices of either P_{ij} or S_{ji} are exhausted. The output is a sequence of closed circuits $Z_{ij}^{(1)}, Z_{ij}^{(2)}, \ldots$

Decomposing Z_{ij} into disjoint sub-circuits

S1. Set $v_b \leftarrow v_j$. Set v_s to the vertex immediately after v_j in S_{ji}. Set $k \leftarrow 1$.

S2. Start walking along S_{ji} from v_s. Let u be the first vertex that is also present in P_{ij}. Create the circuit $Z_{ij}^{(k)}$ as the sequence $u \rightsquigarrow v_b \rightarrow v_s \rightsquigarrow u$. Set $k \leftarrow (k+1)$.

S3. If the vertex next to u on S_{ji} matches with the vertex prior to u on P_{ij}, pick the vertex next to u (this is to avoid degenerate sections where S_{ji} and P_{ij} overlap completely). Repeat until no such overlap is seen. Let u be the last vertex that is shared by both curves after the previous operation. Let u_s^- (resp. u_p^+) be the predecessor (resp. successor) of u in S_{ji} (resp. P_{ij}). Delete the sequence $v_b \rightsquigarrow u_s^-$ from S_{ji} and $u_p^+ \rightsquigarrow v_b$ from P_{ij}. Set $v_b \leftarrow u$. Set v_s to successor u in S_{ji}. Go to Step **S2**.



(a) (b)

Figure 2.16: (a) An interval P_{ab} on the gumline (shown in red) and the shortest path S_{ba} (shown in blue). (b) A self-crossing Z_{ab} obtained by concatenating P_{ab} with S_{ba}. We can see that Z_{ab} is decomposed into edge disjoint circuits: (1) $a \rightsquigarrow x \rightsquigarrow a$, (2) $y \rightsquigarrow x \rightsquigarrow y$, and (3) $b \rightsquigarrow y \rightsquigarrow b$. Note that the order of vertices in the original curves (left) is preserved in the new curves as well.

Compute enclosed vertices. As mentioned previously, a closed circuit on a 3D mesh partitions the vertices of mesh into three classes, i.e., either on the circuit, inside the circuit or outside the circuit. We assume that the number of vertices inside the circuit is much smaller than the number of vertices outside the circuit. This is a reasonable assumption because the regions of the mesh enclosed by the circuits corresponding to the intervals on the gumline are much smaller than the remaining mesh. Given edge-disjoint $Z_{ij}^{(1)}, Z_{ij}^{(2)}, \ldots, Z_{ij}^{(k)}$, we individually compute $e_{ij}^{(1)}, e_{ij}^{(2)}, \ldots, e_{ij}^{(k)}$, and take their union. Each vertex v of the mesh has access to a list of its neighboring vertices denoted as N_v. The vertices in N_v are ordered as they appear around v on the faces incident on v, starting from an arbitrary neighbor.

An algorithm to compute the enclosed vertices e_{ij} of a non-self-crossing closed circuit Z_{ij} with oriented edges is given below:

Computing the set e_{ij} of vertices enclosed by the circuit Z_{ij}

S1. Find an edge $v_1v_2 \in Z_{ij}$ such that the third vertex on each of the two triangles sharing the edge v_1v_2 is not in Z_{ij}, and set $E_1 \leftarrow \{o_1\}$ and $E_2 \leftarrow \{o_2\}$, where o_1 and o_2 are the opposite vertices.

If no such edge is found, pick an arbitrary edge $v_1v_2 \in Z_{ij}$. Let its opposite vertices be o_1 and o_2. Assume $o_1 \in Z_{ij}$. Then add the vertex x_1 (centroid of $\Delta v_1v_2o_1$) to the mesh and re-triangulate $\Delta v_1v_2o_1$ to get three new triangles: $\Delta v_1v_2x_1$, $\Delta v_1o_1x_1$ and $\Delta v_2o_1x_1$. Set $E_1 \leftarrow \{x_1\}$. If $o_2 \in Z_{ij}$, process it similarly.

S2. Start simultaneous BFS to grow the two connected components represented by E_1 and E_2 as described below. Initialize the BFS queue Q_1 (resp. Q_2) corresponding to E_1 (resp. E_2) with its single existing member. In each round, add a reachable unvisited vertex v to both E_1 and E_2. If $v \notin Z_{ij}$, add the vertices in N_v to the queue from which v was extracted.

However, if $v \in Z_{ij}$, explore N_v as follows: Walk around N_v and partition it into two sets D and D' as the vertices occurring between v^- and v^+, where N_v is ordered around v, and, v^- and v^+ are the predecessor and successor of v in Z_{ij}). Choose D as the set that contains the parent of v in BFS order (say u). Also, v^- and v^+ are not included in D or D'.

Now, if $u \in E_1$, then set $Q_1 \leftarrow Q_1 \cup D$ and $Q_2 \leftarrow Q_2 \cup D'$. Else, set $Q_1 \leftarrow Q_1 \cup D'$ and $Q_2 \leftarrow Q_2 \cup D$.

Repeat **S2** and terminate as soon as either E_1 or E_2 stops growing.

S3. Among E_1 and E_2, choose the set with fewer vertices as e_{ij}. It follows from the assumption stated earlier that the partition inside Z_{ij} has fewer vertices than the partition outside and, thus, the inside set would be the first one to get exhausted by a BFS.

Computing invalid intervals to be repaired

Using the definition of intervals and a score threshold S_m, we now have a candidate set $\mathcal{C}$ of invalid intervals to be repaired. The goal in this section is to solve Problem **IIM**, i.e., to extract a set of intervals $\mathcal{I} \subseteq \mathcal{C}$ that are pairwise non-overlapping such that the sum of the lengths of the intervals in $\mathcal{I}$ is maximized. A set of non-overlapping intervals is called a *feasible* set. Below we describe a technique to find $\mathcal{I}$. (In practice, we may need more than one round of invalid intervals detection and repair.)

As a first step, we compute a threshold such that only those intervals having a score greater than this threshold are retained in $\mathcal{C}$. Higher scoring intervals correspond to portions of the gumline that are poorly-formed and so these must be included in $\mathcal{I}$ for repair. We obtain such a threshold by clustering the scores in the gumline into two groups (k-means clustering) and use the mean score of the cluster with the highest scores as the threshold.

Next, we eliminate all the intervals from $\mathcal{C}$ that are completely covered by the union of other intervals. For example, in Figure 2.15, interval $[a, c]$ is completely covered by $[a, b]$ and thus, will be eliminated. The intuition behind this is that if there is an invalid interval I that is covered by the union of multiple intervals, then I can be safely ignored as its range is subsumed by the overlapping interval(s).

So, we obtain the modified set $\{I_1, I_2, \ldots, I_N\}$ which has intervals that are not completely covered by other intervals; we continue to call this set $\mathcal{C}$. We solve Problem **IIM** on this set $\mathcal{C}$ using dynamic programming. For simplicity, we represent each interval as $[l_i, r_i]$, where l_i (resp. r_i) corresponds to the *index* of the first (resp. last) vertex of the interval, as measured from a fixed vertex (e.g., v_0) on the gumline curve. The length of each interval I_i is denoted as d_i. The algorithm to compute $\mathcal{I}$ is given below:

Solving problem IIM

S1. Sort all the intervals in $\mathcal{C}$ based on the left endpoints of the intervals. Thus, after sorting, $l_1 \leq l_2 \leq \cdots \leq l_N$.

S2. Define $M(i)$ as the maximum sum of lengths possible in a feasible solution that includes intervals $I_1, I_2, \ldots, I_i$. For each vertex, compute $M(i)$ as follows: $M(0) = 0$ and $M(1) = d_1$. For $2 \leq i \leq N$,

$$M(i) = \begin{cases} M(i-1) + d_i, \text{ if } I_i \text{ does not overlap with } I_{i-1} \\ \max\{M(i-1)\ ,\ M(i-2) + d_i\}, \text{ otherwise} \end{cases} \tag{2.1}$$

S3. The maximum possible sum of lengths of a feasible solution is $M(N)$ and the intervals realizing this length are the ones chosen in the process.

The above algorithm runs in $\Theta(N \log N)$ time as it involves a single sort and scan of the input, where $N = |\mathcal{C}|$. The correctness of the algorithm can be proved by analyzing the cases of equation 2.1 (the base cases are trivial). If I_i does not overlap with I_{i-1}, then it can be clearly included in the solution. If I_i does overlap with I_{i-1}, then we can prove that it does not overlap with I_{i-2}.

Proposition. Let $I_1, I_2, \ldots, I_N$ be a sequence of intervals $I_i = [l_i, r_i]$ sorted in non-decreasing order of the left endpoints. If no interval is contained in the union of other intervals, then I_i does not overlap with I_{i-2}, where $i \geq 2$.

Proof. It is enough to show that $r_{i-2} < l_i$, i.e., the interval I_i starts after interval I_{i-2} ends. Assume this is not true for the sake of contradiction. So, we have $r_{i-2} \geq l_i$ for some $2 \leq i \leq N$. We know from the sorted sequence and our assumption that $l_{i-2} \leq l_{i-1} \leq l_i \leq r_{i-2}$. Given this, we have the following possible configurations for inserting r_{i-1} and r_i into this sequence:

1. $l_{i-2} \leq l_{i-1} \leq l_i \leq r_{i-2} \leq r_{i-1} \leq r_i$. This leads to the conclusion that interval I_{i-1}

is covered by the union of I_{i-2} and I_i.

2. $l_{i-2} \leq l_{i-1} \leq l_i \leq r_{i-2} \leq r_i \leq r_{i-1}$. This trivially shows that I_i is covered by I_{i-1}.

3. $l_{i-2} \leq l_{i-1} \leq r_{i-1} \leq l_i \leq r_i \leq r_{i-2}$. This and the following cases are trivial as in all these cases, I_{i-2} covers both I_{i-1} and I_i.

4. $l_{i-2} \leq l_{i-1} \leq l_i \leq r_{i-1} \leq r_i \leq r_{i-2}$.

5. $l_{i-2} \leq l_{i-1} \leq l_i \leq r_i \leq r_{i-1} \leq r_{i-2}$.

Thus, each possible configuration lead to a contradiction of the fact that no interval is completely covered by the union of other intervals, which proves the proposition. □

Repairing invalid intervals

Repairing an invalid interval depends on whether it is Type-T or Type-G. Type-G (resp. Type-T) intervals enclose actual teeth (resp. gum) vertices that have been wrongly classified as gums (resp. teeth). Thus, for repairing a Type-G interval $[v_i, v_j]$, we relabel vertices in e_{ij} as teeth and add them to the teeth component. Also, the invalid interval portion of the current gumline (P_{ij}) is replaced with the shortest path between v_i and v_j (S_{ij}). Similarly, if $[v_i, v_j]$ is a Type-T interval, vertices in e_{ij} are added back to the gum components and the gumline is modified in a similar way. Figures 2.17 and 2.18 show some examples of invalid intervals and the result their repair.

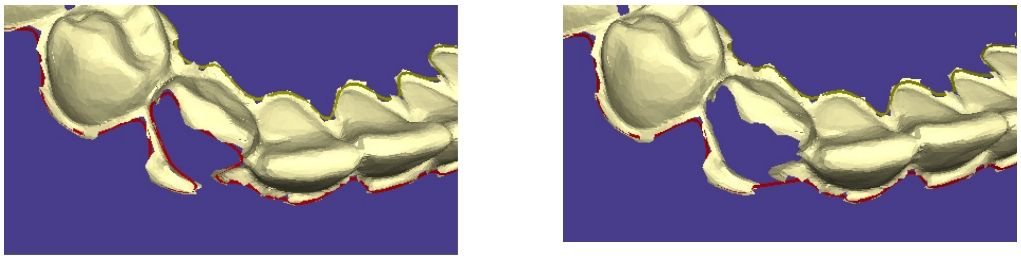


Figure 2.17: Example showing how an invalid interval (Type-G) in a curve is repaired by a shortest path between its end vertices (For clarity, the relabeled vertices in corresponding e_{ij} are not shown to illustrate the corrections made to gumline.)

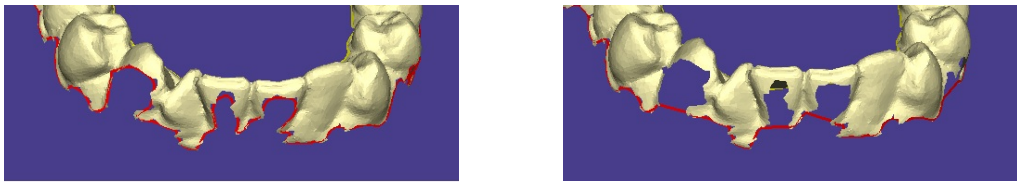


Figure 2.18: Multiple instances of adjacent invalid intervals (Type-G) can also be repaired.

2.4.5 Tooth-tooth separation

At this stage, we have separated gums from teeth and extracted a curve that closely approximates the natural gumline of model. In this section, we will discuss some problems related to identifying different teeth in a teeth component. We first discuss below the problem of identifying interstitial sites between adjacent teeth. We then discuss the problem of matching these sites on the lingual and labial sides of the gumline and constructing actual separators between adjacent teeth on the mesh.

Detecting interstitial sites

As mentioned previously, corners on curves are defined as vertices on a curve that provide information on boundaries between different structural regions of the curve. These are calculated as the vertices with dominating curvatures (sharp bends) in the sections of the curve. However, there is no generally agreed definition of corners for planar curves that fits all logical meanings of corners (i.e., it is quite application-dependent). Moreover, sometimes vertices themselves may not be high-curvature vertices locally, but when considered globally, they may bring out structural information about the curve. On the other hand, some high-curvature vertices may not be "true" corners when viewed globally (due to noisy, irregular sections in the curve). More examples and discussions can be found in [14, 51].

Observe that the interstitial sites on a model are included among the "corners" of the gumline. It is difficult to compute corners reliably on a gumline because of local irregularities due to noise and the fact that it is essentially a 3D curve. We use techniques for 2D curves from [51] and extend them to 3D curves. A typical result of this corner detection is shown in Figure 2.19 (left). The method works in two phases: first we find a

set of candidate corners, and then follow this by the computation of the actual corners. The algorithm is given below:

Computing the corners containing interstitial points

S1. Fix a constant t. For each vertex v_i on the gumline curve, let v_k the vertex with index $k = (i+t)$ mod n, i.e., v_i and v_k define a window with a constant number of vertices. For each vertex in this window, compute its distance from the 3D line through points v_i and v_k. Let v_j be the vertex that achieves a maximum score of c_j. If this is the current best score for v_j and this score is greater than a constant threshold D, then record the triplet $[v_j, c_j, v_i]$ as a candidate corner. This triplet indicates that v_j is a candidate corner, was generated by interval at v_i and has a score c_j.

S2. Scan all the candidate vertices and select a vertex as a corner if there is no other candidate with a better score in its r-neighborhood, where r is a constant.

Our experiments used $D = 0.2$ mm, $t = 30$ and $r = t$ across all models. The parameters vary from one application to other and must be chosen experimentally to be most effective. However, note these values are quite robust and can handle a wide range of models as in this case. This algorithm, however, does not solve our problem entirely because along with correct interstitial sites, we also get the vertices from midpoints of gumline around each tooth (Figure 2.19) as they are also high curvature vertices according to our definitions. These spurious corners can be easily filtered out by checking the type of interval generating them (see Section 2.4.4). Observe that in most cases, a true interstitial site (resp. spurious corner) would be generated by a Type-G (resp. Type-T) interval on the gumline. Thus, using such a test we obtain the true set of interstitial sites as shown in Figure 2.19 (right). Note that even with this test, a few rare spurious corners may persist. The next step of matching these corners tries to address this problem.

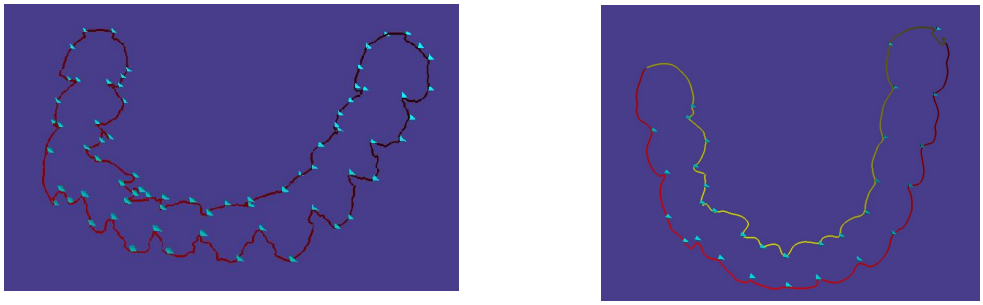


Figure 2.19: Left: Detection of candidate corner vertices on the gumline. Right: Identification of interstitial points among the candidate corners. Note that the gumline is now split into a lingual curve (shown in yellow) and a labial curve (shown in red).

Matching interstitial sites

We first split the gumline into labial and lingual sides and compute the set of interstitial sites belonging to these sections. Now we proceed to pair them up from lingual and labial sides to form the embrasure points between adjacent teeth. Denote the set of lingual (resp. labial) vertices as L_N (resp. L_B). A practical requirement of our matching algorithm is that it produce as few false positive pairings as possible. This is because in a real-world system, false pairings lead to spurious separators that a user must delete manually, which is time-consuming (instead of maybe just adding a new separator curve). We use the following matching algorithm:

Matching corners from the lingual and labial sides

S1. Without loss of generality, assume that there are fewer corners in L_B than in L_N. Find the nearest neighbor in L_N of each $u_b \in L_B$. If each corner from L_N was selected as a nearest neighbor for a unique corner in L_B, then it is conflict-free (i.e., no two corners in L_B claim a common corner in L_N). Else, the conflicts must be resolved to avoid any false positives. Out of all the candidate vertices from L_B that are mapped to a vertex $v \in L_N$, we pair up v with its closest vertex (with respect to Euclidean distance) among all the candidates and delete the other mappings to v.

S2. Draw separating curves between every pair of matched interstitial points as a shortest path on the surface of the mesh. If the teeth are well-represented in the model and interstitial sites detected are not too far away from ground truth, then these separating curves pass through the valley regions between adjacent teeth. This gives a clean separation of teeth as shown in Figure 2.20.

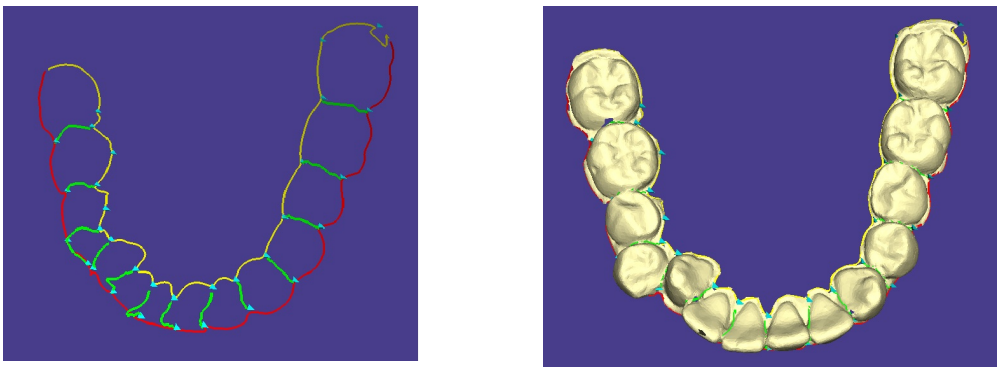


Figure 2.20: Left: Result of drawing separating curves (shown as green curves) between matched pair of interstitial points. Right: Separating curves shown with the teeth.

2.5 Software tool for segmentation

We begin with a brief description of the software tool developed using the techniques described in this report and then discuss the results obtained using this system. The average number of vertices and triangles in a full dental model are about 250K and 80K, respectively. After preprocessing to remove the faces that do not contribute to the segmentation task, the average number of vertices and triangles retained are about 70K and 30K, respectively.

2.5.1 Software tool

This software tool is written in C++ in the environment provided by Qt Open-Source Edition 4.5 [42]. It consists of a simple, intuitive user interface that can be used to load dental models and segment them. Figure 2.21 shows the screenshot of the tool with all the commands that pertain to segmentation[1]. The main window on the right displays the graphics which provides the common viewing transformations. The top left window has a vertical slider that is used to adjust the curvature threshold to be applied to the model. The user can view the result of curvature thresholding on the feature regions in real time and can set it to a reasonable level using the slider in an intuitive way. Following this, a single click on the "Segment Teeth" button proceeds to carry out all the steps of the algorithm from Section 2.4.2. The result of this is the final output showing teeth and gums in their natural color with separator curves between adjacent teeth.

The button called "Show Detailed Steps" brings up (as well as hides) the window on the bottom left side which shows all the operations that are carried out to achieve segmentation. Note that, once the curvature value is set, the system is fully automatic and requires no further input from the user. In practice, however, the orthodontist may need to make fine adjustments to some separators. For such real-world situations, the system provides manual controls as well. However, note that the interface is absent of any parameters to be manipulated by the user except the setting of the curvature threshold,

[1]The tool incorporates algorithms for segmentation, feature identification and alignment. Discussions and screenshots of the tools as they pertain to feature identification and alignment appear in Chapter 3 and Chapter 4, respectively. The entire tool comprises about 30K lines of code.

which is quite intuitive and common in such systems (we discussed briefly in Section 2.4.3 why this may be necessary for any automatic dental segmentation approach). Thus, this system provides a tool which can be used in a real-world clinical environment with minimal user interaction.

In addition to the operations mentioned, the window also has controls to turn off the display of specific information computed on the model, e.g., gumline, teeth, gums, separator curves, etc.

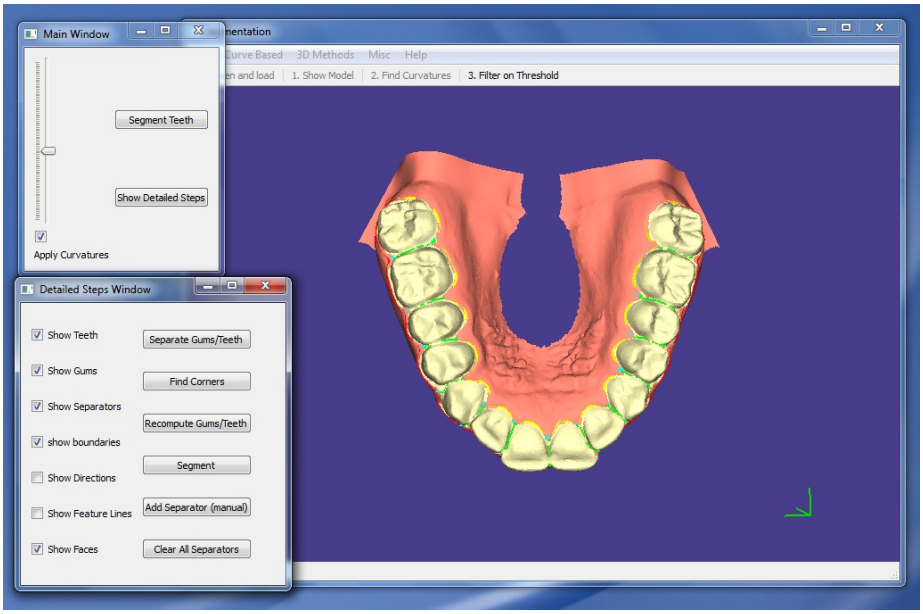


Figure 2.21: Screenshot of the segmentation tool. A model is shown after removing the triangles near the base. The window on the right displays the graphics. The upper-left window has the controls to setup a curvature threshold and initiate teeth segmentation. The lower-left window, with detailed steps and view controls, can be opened by using the "Show Detailed Steps" button on the window above.

2.5.2 Discussion of results

We will begin our discussion of performance of our system with two typical types of models that one comes across in orthodontics and show the behavior of this system in the presence of malocclusions and poorly-represented gumline in the base models. Next,

we will describe a common problem concerning noisy regions in the upper jaws and how this problem is addressed in the current system. Finally, we will look at the results of this system on a model which is difficult to automatically segment because of both malocclusions and a flat gumline.

Table 2.1 shows the process of automatically segmenting two models. The information computed at various stages is shown. This process is triggered by the "Segment Teeth" button after the user has set an appropriate curvature threshold while viewing the corresponding feature regions shown in row 1. Following this, the system executes each step without any manual intervention and produces the results shown in the images in row 4. The gumline repair method is found to be quite robust to different kinds of abnormalities in the gumline at the molars and the anteriors.

A typical problem with the laser-scanned models from plaster casts is the noise in the upper jaw (Figure 2.12 (b)). Upon observing the gum regions below the incisors, we find that there are many small patches of high-curvature vertices in the gum region. These are classified erroneously as teeth instead of gums. This is an inherent problem observed by previous approaches as well [57] (e.g., Figure 2.8(a)). In contrast to previous approaches [56, 57] where isolated patches must be marked off using a curve drawn manually, our approach generates a small number of connected components of noisy regions (Figure 2.12 (a)). We have designed heuristics that handle these spurious components. One such technique, which we described earlier, uses a lower bound on the number of vertices on a valid tooth object and merges with the gums any component that has fewer vertices. However, sometimes these noise components may become large. For these cases, we use the following observation: Consider the mean height, from the base, of all vertices classified as teeth. A majority (we use 90%) of vertices of the spurious components lie below this height. In our experiments, these heuristics covered most of the cases involving noisy components. In our software, we have also added a feature wherein a user can delete these noisy regions using a mouse click. Note that this is much simpler in terms of human effort than marking off regions on a 3D surface, as in previous approaches.

The result of our algorithms on a case with poor gumline representation is shown in Table 2.2 (this has also been illustrated in Figure 2.10). Observe that it is difficult to estimate even the archform of such a model correctly. In addition to this, the model

also presents malocclusions. However, our methods are able to identify the gumline and repair it to a reasonable quality. Some of the interstitial points on the gumline are off by a few vertices on the gumline curve. In such cases, the orthodontist may adjust the separator curves using manual controls. Finally, we separate the model into individual tooth objects by partitioning along the separator curves as shown in Figure 2.22.

Figure 2.22: Model separated into individual tooth objects by cutting the teeth components along the separating curves shown in Table 2.1 (row 4 (left)).

We note that a validation study of an early implementation of the software tool was conducted by one of our collaborators from the School of Orthodontics at the University of Minnesota. A set of 10 upper and 10 lower real-world dental arches of varying difficulty were chosen as the test cases. The study compared the automatic segmentation of our tool to the segmentation output of a commercially available tool (3Shape OrthoAnalyzer^{TM} [2]) which requires some user-input to initiate the segmentation process. The comparison showed that 95.1% (194 out of 204) of the separating curves between pairs of adjacent teeth were accurately placed by our segmentation tool (see Figure 2.20 for an example of separating curves). Of the 10 errors, the number of omitted and misplaced separating curves were 6 and 4, respectively. There were a few additional errors in the gum-teeth separation, but, these did not affect the tooth-tooth separation process. There were a total of 14 (resp. 5) instances where a tooth (resp. gum) region of the mesh was incorrectly classified as a gum (resp. tooth) region. Detailed results of this study can be found in [43, 44].

2.6 Conclusion

An approach to the basic problem of automatic segmentation of teeth in a 3D dental model is described. The approach to automatic tooth segmentation presented here is observed to work well on real-world dental models, obtained from actual patients, and reduces considerably the amount of human intervention needed on a majority of the models. The previous approaches to this problem, which mainly used an approach based on morphological operators along with feature regions, have some inherent limitations when applied to laser-scanned dental models in orthodontics. This is due to flat gumlines and scanning errors, which in turn are due to malocclusions in the patient's jaws. Our solution overcomes these limitations by first dividing the problem into two subproblems (Gum-Teeth Separation and Tooth-Tooth Separation) and then solving these sequentially.

Chapter 3 considers the identification of dental features such as cusps, marginal ridges, incisal edges and grooves on the individual tooth surfaces which will be subsequently used in the final alignment of teeth.

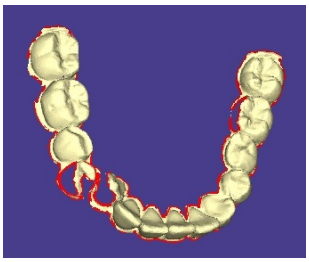
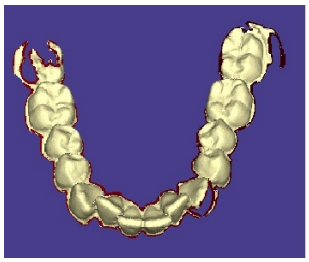
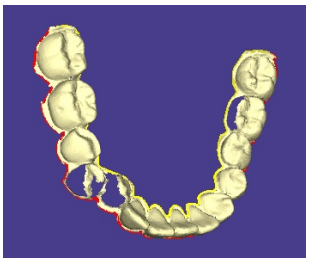
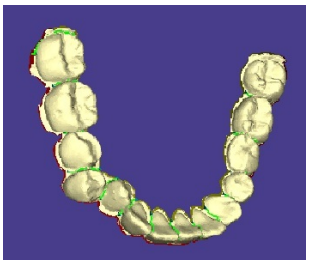
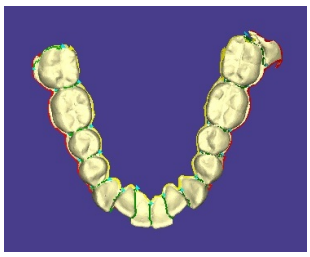
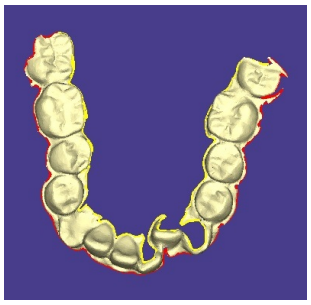
Model 1		Model 2
	(1)	
	(2)	
	(3)	
	(4)	

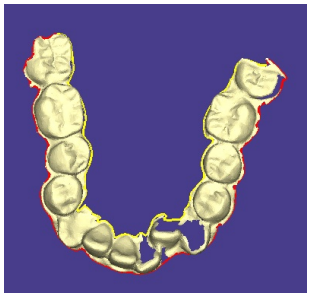
Table 2.1: Different steps in automatic segmentation on two different models: (1) Extraction of feature regions based on curvatures. (2) First approximation to gumline using flood-fill method. (3) Gumline repair. (4) Re-computation of teeth and gum components and detection and matching of interstitial points to create separating curves between adjacent teeth.

(1)

(2)



(3)



(4)

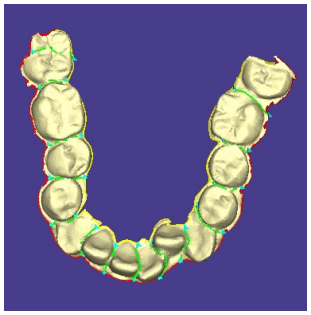


Table 2.2: Results of our approach on a difficult case with poor gumline representation and malocclusions. (The steps are labeled similar to Table 2.1.)

Chapter 3

Feature identification in dental meshes

3.1 Introduction

Chapter 2 presented an algorithm to segment a 3D dental arch into individual tooth objects (i.e., submeshes). We now turn to the identification of relevant features on the surface of these tooth objects. The identification of such a feature set is crucial for orthodontic treatment planning (see Section 3.1.1). However, identifying features manually, by having a clinician "eyeball" each tooth on a virtual interface is labor-intensive, time-consuming, and prone to error. Our goal is to identify certain intrinsic tooth surface features (cusps, ridges, grooves, etc.) automatically, with intervention by the user to verify or correct identified features only in difficult or unusual cases.

This chapter discusses the computational issues associated with automatic feature identification and presents a collection of algorithms to do this effectively. Our algorithms are based on curvature analysis, clustering on 2D cross-sections of tooth surfaces, and an adaptation of the watershed method for segmentation. As mentioned earlier, we have incorporated these algorithms in a software tool. At the end of this chapter, we present experimental results on clinical data that show that our algorithms are effective at automatic feature identification on noisy and incomplete real-world dental models.

3.1.1 Dental features and their importance

Proper orthodontic tooth alignment and functionality (e.g., chewing) depend on a number of intrinsic features on the tooth surface, including cusps, grooves, incisal edges, ridges, occlusal contact region, etc. (These terms are defined in Section 3.2.) These features are important for many reasons: They provide a set of landmarks that can be used to define alignment requirements quantitatively (e.g., height difference between adjacent marginal ridges) [47]. Also, the features themselves do not change over time (as they are intrinsic to the tooth surface), so they can be used to monitor the progress of the alignment. Moreover, since certain (derived) features need to remain invariant throughout treatment (e.g., the distance between canine tips), the intrinsic features dictate the best possible alignment that can be achieved and provide a means for evaluating a computed alignment. Finally, features such as number of cusps are useful in classifying teeth automatically (e.g., molars, premolars, etc.)

Dental features have applications in other related areas. In the field of computer-aided planning and simulation in craniomaxillofacial surgery, the correct dental occlusion or the *maximum intercuspation* of digital 3D representations of the arches of a patient must be reestablished. This is in general difficult in a virtual environment and is computed approximately by aligning the grooves and marginal ridges of the upper arch with the corresponding cusps and incisal edges of the lower arch [13]. Also, finer dental occlusion depends on getting a close fit in the areas of contact between the surface of the two arches [13, 21]. It is known that only the tooth surface regions bounded by the occlusal and marginal ridges participate in occlusal contacts. Thus, these features can speed up significantly the surface matching and alignment algorithms in virtual surgical planning.

3.1.2 Challenges in feature identification

The input to the feature identification task is a collection of meshes representing individual tooth objects. The feature identification task is complicated by the fact that the meshes are almost always noisy and incomplete. This is due to the limitations (resolution) of the laser scanning process itself in the presence of malocclusions and crowding of teeth. A practical requirement of the feature identification algorithms that we seek is that they should be robust to noise and missing information.

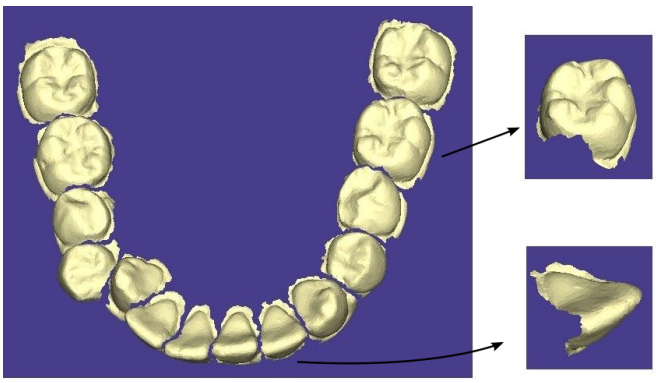


Figure 3.1: Input to the feature identification task. A segmented lower arch is shown on the left and the individual meshes, representing tooth objects, are shown on the right. Note the highly irregular occlusal surfaces of molars and premolars that tend to create problems for the detection of ridges, grooves and cusps.

3.1.3 Organization of the chapter

Section 3.2 defines relevant anatomical terms and the features of interest. Section 3.3 describes an algorithm for cusp identification based on the familiar watershed method. Section 3.4 introduces a general approach based on curvature analysis and clustering on 2D cross-sections of tooth surfaces for identifying other features such as incisal edges, grooves, marginal ridges, and occlusal surface boundary. Section 3.5 discusses how these features are identified and presents the results on some test cases. Section 3.6 describes the identification of some derived features and Section 3.7 discusses the quantitative evaluation of the techniques developed on real-world clinical data.

3.2 Dental anatomy and dental features

3.2.1 Dental anatomy

Teeth are classified as *incisors*, *canines*, *premolars* and *molars*. Each *dental arch*, i.e., row of teeth, can be divided into a *left* and a *right* side. Each side has two incisors (*central* and *lateral*), one canine, two premolars (*first* and *second*), and three molars (*first*, *second* and *third*), where the premolars and molars are enumerated from front

to back as first, second, etc. The incisors and canines are collectively called *anteriors* and are used in cutting action. The premolars and molars are called *posteriors* and are involved in chewing action.

The inner (resp. outer) part of the tooth on the tongue (resp. face) side is called the *lingual* (resp. *facial*) side. The face side on posteriors (resp. anteriors) is called the *buccal* (resp. *labial*) side. A tooth's surface towards the front (resp. back) of the arch, i.e., towards (resp. away from) the central incisors, is called the *mesial* (resp. *distal*) side.

A suitably chosen line through the mesial and distal side of each tooth defines the *mesiodistal* line of the tooth. Similarly, the lingual and buccal (or labial) sides define the *buccolingual* line. These lines are important in feature identification and in understanding tooth functionality. All definitions above are illustrated in Figure 3.2.

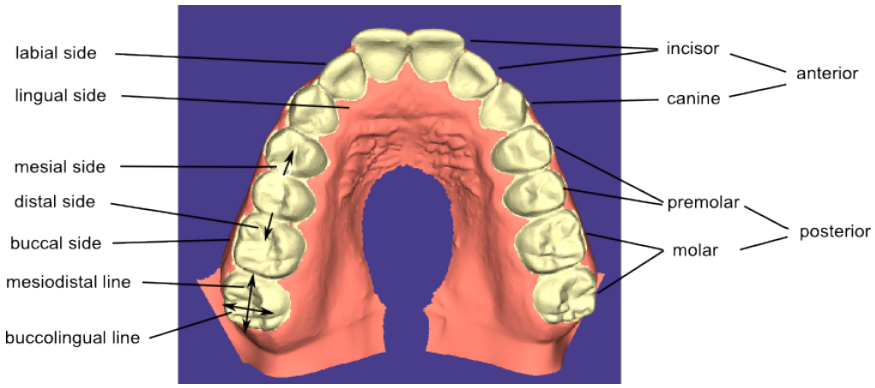


Figure 3.2: Dental anatomy.

3.2.2 Dental features

Now, we define some of the features on tooth surfaces that are important in orthodontic treatment planning. The surface of the teeth resembles a terrain with "mountain peaks", "ridges" and "valleys". For the ease of understanding, it is convenient to define dental features in terms of these terrain-like features. More details on dental anatomical features can be found in standard textbooks on dental anatomy such as [55].

Incisal edges: These are the sharp ridges on the anteriors (incisors and canines) running along the mesiodistal line. Figure 3.3 shows an example.

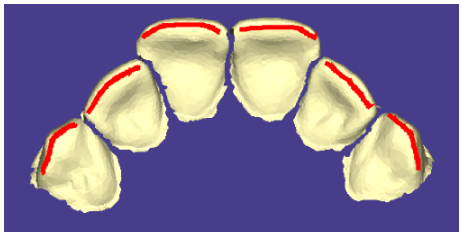


Figure 3.3: Incisal edges on the anterior teeth (shown as red curves).

Cusps: On premolars and molars these are the mountain peak-like structures on the surface, at the corners of the tooth. Each cusp has *cusp ridges* radiating from its tip, similar to ridges that connect mountain peaks to valleys on a terrain; these can be used to define other features such as the occlusal surface boundary (explained below).

Canines have a single cusp which plays an important role in determining the overall quality of the alignment. The premolars (resp. molars) have 2 or 3 (resp. 4 or 5) cusps depending on the arch (upper or lower) and the individual. These cusps are named according to their position on the tooth surface. For example, a molar would have a mesiolingual cusp situated on the mesial and lingual side of the tooth. Similarly, for mesiobuccal, distolingual, distobuccal, lingual and buccal cusps. Some of these cusps are shown in Figures 3.4 and 3.5.

Occlusal surface and marginal ridges: The *occlusal surface* of a posterior tooth is the area of the tooth surface where chewing occurs. This is also the contact area between corresponding posterior teeth from opposing arches (Figure 3.4). Thus, the occlusal surface complements the functionality of incisal edges.

The *marginal ridges* are located at the mesial and distal ends of the occlusal surface. These are the regions where the mesial or distal walls of a tooth make contact with the occlusal surface (Figure 3.4). Thus, each tooth has a mesial and a distal marginal ridge. For incisors, the marginal ridges are on the vertical sides of the teeth since the occlusal surface of incisors is the lingual (resp. labial) surface on the upper (resp. lower) arch.

The occlusal surface of a posterior tooth is bounded on the buccal and lingual sides by the cusp ridges (inclusive of the cusps). On the mesial and distal sides, the occlusal surface is bounded by the two marginal ridges. This provides a boundary around the occlusal surface area called *occlusal surface boundary* (Figure 3.4). Thus, the occlusal surface boundary is a curve on the tooth surface that connects the marginal ridges, cusp ridges, and the cusp peaks.

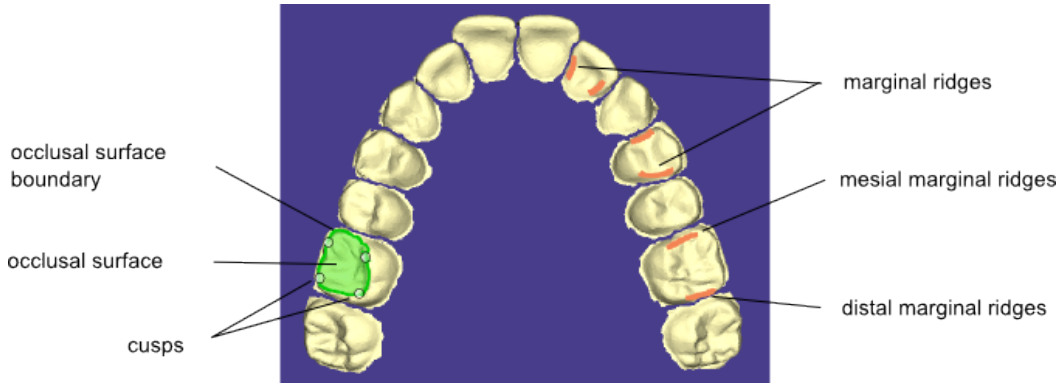


Figure 3.4: Cusps, marginal ridges, and occlusal surface boundary in an upper arch, as found by our feature identification algorithms. Note the difference in the marginal ridges on incisors and posteriors.

Grooves: These are the depressions and fissures on the occlusal surface of a posterior tooth that resemble riverbeds and valleys on a terrain. There are various types of grooves and corresponding classification and naming conventions. For our study, we are interested in the long grooves running along the mesiodistal line of the tooth, called *central developmental grooves*, or just *central grooves*. Figure 3.5 shows some examples of (approximations of) grooves that are useful for orthodontic purposes.

Figure 3.5 also highlights the difference between the grooves on the premolars on the upper and lower arch (these are more curved). On the other hand, the grooves on lower arch molars are much more straight than their upper arch counterparts. In addition to these, we also use the *buccal grooves* on molars, which are depressions on the buccal side of the crown running from the occlusal surface towards the gums. Information on several other types of grooves and their uses can be found in [55].

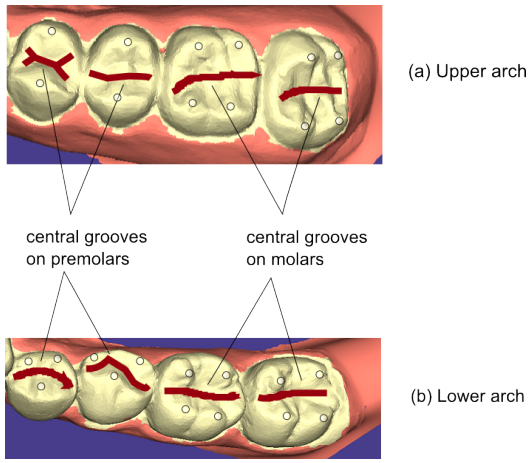


Figure 3.5: Grooves on the molars and premolars of the lower and upper arch. Many different types of grooves can be defined on the tooth surface based on their position and dimensions. The images above show the approximation of the central grooves that are believed to be the most relevant for orthodontic treatment planning. The circles indicate the cusps on teeth.

Other derived features: In addition to the above-mentioned features, which are intrinsic to the tooth surface, there are other features that are derived from these intrinsic features and are important in alignment. These include the archform, the occlusal plane, and the interproximal contact points [55].

The *archform* is defined as an appropriate smooth curve through the incisal edges of anteriors, canine cusps and the buccal cusps of molars and premolars (Figure 3.6). The archform determines the overall quality achievable by the alignment process.

The *occlusal plane* is defined as a smooth surface passing through the occlusal or biting surfaces of the teeth. It is an imaginary surface at which the upper and lower arches meet and is important in establishing the vertical positioning and the buccolingual orientation of teeth in the final alignment.

The *interproximal contact points* are the contact points between two adjacent teeth on the same arch; they are viewed as derived features as they are not intrinsic to a tooth but can be determined from the surfaces of two adjacent teeth. They are important in establishing the correct vertical alignment of posterior teeth from opposing arches.

Figure 3.6: Archform of a dental model.

3.3 Cusp identification

This section describes the identification of cusps on tooth surfaces. Our approach is based on the *watershed* class of algorithms [39] for mesh segmentation, reviewed briefly in Section 3.3.1.

3.3.1 Watershed-based mesh segmentation

This is a class of mesh segmentation algorithms that partition the mesh using the analogy of flooding of the mesh [39]. As an illustration, consider the one-dimensional curve shown in Figure 3.7. Each point on the curve is assigned a real-valued height using a height function $\mathcal{H} : v(x, y) \rightarrow \mathcal{R}$. In Figure 3.7, the height is simply the point's vertical elevation from the base (i.e., its y-coordinate). Imagine that each point on this curve has a drop of water stored as initialization. During the course of the algorithm, the water drop flows towards its most steep neighbor (with respect to the height function) until it reaches a minima point. The initial positions of the water drops are shown as the filled circles in Figure 3.7 along with the direction of their flow. The water is thus collected in catchment basins which are defined by a single local minimum point shown as an empty circle at the bottom of the curve. All the points that are visited by a water drop during its flow to a minimum point are given the same label, which is denoted by the minimum point. This process partitions the curve into regions defined by the labels as shown in Figure 3.7. In the vicinity of a flat (i.e., zero-slope) portion of the curve, we

can choose to arbitrarily advance the flow in either direction so that a valid partitioning is achieved.

This technique was generalized to segmentation of 3D surfaces by Mangan and Whitaker [39] who used the surface curvature value for the height function. The algorithm in the form described is cumbersome to implement because we have to keep track of each flow and also use placeholders for assigning labels to vertices on its path because the final local minima of the flow are not known beforehand. Many improved implementation schemes have been suggested in the literature to overcome these difficulties [48].

These algorithms do not compare well with negative minima-based approaches for mesh segmentation as they are not very good at finding partition boundaries in arbitrary meshes (see [48, 49] for details). However, they are good at finding the different local minima of a given height function, but often lead to over-segmentation (i.e., a large number of partitions). Thus, to obtain large meaningful partitions, it is often necessary to merge small-sized spurious partitions with other partitions in their neighborhood. For example, Figure 3.7 shows a set of three initial partitions resulting from the procedure described above. However, if the goal is to find less than three partitions, we may merge partitions 1 and 2 instead of partitions 2 and 3 (depending on the *depth*, i.e., the smallest height difference between the minimum of a partition and any boundary point of that partition) to reduce the number of final partitions.

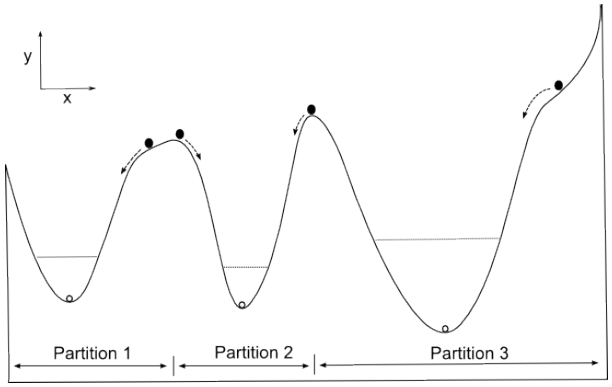


Figure 3.7: Example illustrating the watershed-based approach to segmentation.

3.3.2 Watershed-based cusp detection

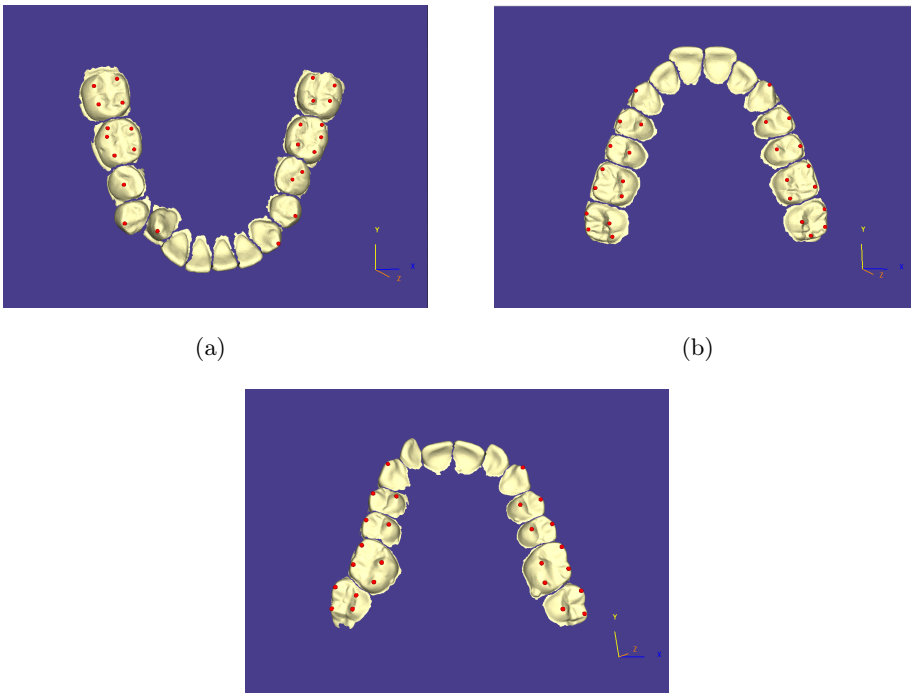
As mentioned in Section 3.3.1, watershed-based techniques are well-suited to find the cusps on the tooth surface which are defined as vertices corresponding to local minima for some intuitively defined height function. (For teeth on the upper arch, cusps are local minima while for those on the lower arch they are local maxima; we treat both cases uniformly, as local minima, by negating vertex coordinates in the latter case.)

The key to watershed-based cusp extraction is designing a good height function $\mathcal{H}$. Most generic mesh segmentation schemes define $\mathcal{H}$ solely on the basis of surface curvature, so as to be able to segment meshes from a variety of sources. Here we take advantage of the fact that we are dealing with tooth surfaces and we define $\mathcal{H}$ on the basis of both surface curvature and the elevation of the vertices (i.e., their z-coordinates, since the scanned meshes are provided with a base that is parallel to the xy-plane). More precisely, for a vertex v on the surface of a tooth in the upper arch, we define its height as $\mathcal{H}(v) = (1-\alpha)\cdot(-K_v) + \alpha \cdot v_z$, where v_z is v's z-coordinate, K_v is the surface curvature at v [40], and α is a parameter that controls the relative influence of K_v and v_z on $\mathcal{H}(v)$. In our implementation, we found that choosing α from the interval $[0.4, 0.6]$ worked well for all the models tested.

3.3.3 Results and discussion

Figure 3.8 shows the result of applying the cusp identification algorithm with the above height function on three different models. Figure 3.8(a) shows a lower arch in which teeth are not well-aligned. Figures 3.8(b) and 3.8(c) show two upper arches where the molars are rotated away from the vertical z-axis by different amounts. Note that the number of cusps for teeth on the lower arch can be four or five, and we may sometimes have to deal with partially erupted molars.

After the cusps have been identified automatically, our software tool allows the user to fine-tune the results manually (for very difficult cases) by clicking on the tooth surface to add/delete (hence move) cusps. For instance, a click to add a cusp triggers a search for the local minimum in a small neighborhood of the surface around the location of the click and places a cusp at this minimum. Thus the user does not have to "eyeball" the mesh to find the exact location of the desired cusp.



(a) (b)

(c)

Figure 3.8: Cusps (shown red) identified using the watershed method.

3.4 A general approach to feature identification

In this section we discuss how to identify the remaining features of interest (beyond cusps), i.e., incisal edges, central grooves, marginal ridges, and occlusal surface boundary. It turns out that these features can all be identified using a general approach that is based on computing certain 2D cross-sections of the tooth surface, analyzing the curvatures of the (piecewise-linear) curves that define these cross-sections to determine points of sufficiently high (positive or negative) curvature, and identifying clusters of such points that form the features of interest.

Our approach starts by creating a sequence of parallel planar cross-sections of the tooth surface, one set oriented along the mesiodistal line and one along the buccolingual line. For each cross-section in each set, we perform 2D curvature analysis at the vertices of the piecewise-linear curve that defines the cross-section and identify points

of high curvature and cluster these to identify regions of interest on each curve. Then appropriate regions are selected from each curve and stitched together with those from curves of neighboring parallel cross-sections to obtain a connected and coherent feature (e.g., a ridge or groove). Thus, this approach provides a means to extract the desired 3D features in a "guided" fashion from a set of 2D regions of interest on individual cross-sections. This has several advantages over an approach that attempts to extract features directly, via curvature analysis in 3D, as we will discuss in Section 3.4.5.

We will now explain the main steps of the approach in detail.

3.4.1 Computing a medial curve

Consider the left side of the arch in Figure 3.2. We can define a curve that passes suitably through the mesial and distal sides of each tooth on the left side starting from the last molar and ending at the central incisor. A similar curve can be defined for the right side of the arch. These two curves meet at the mesial sides of the two central incisors. We define the *medial curve* of the arch as the curve obtained by joining the curves from the left and right sides of the arch. (The term "medial curve" signifies that the curve involves both the mesial and distal sides of each tooth on the arch.) The medial curve will guide the feature identification process outlined above. The medial curve represents an approximate layout of all the teeth on the arch and also provides a mesiodistal line for each tooth that will be needed to compute the mesiodistal cross-sections, as we will see in Section 3.4.2.

The medial curve is not unique since many different curves can satisfy the given definition. For our purposes, it suffices to have an approximate medial curve that reliably identifies the mesial, distal, buccal (labial) and lingual sides of a tooth. Accordingly, we compute the medial curve as shown below. Figure 3.9 shows the results of this algorithm.

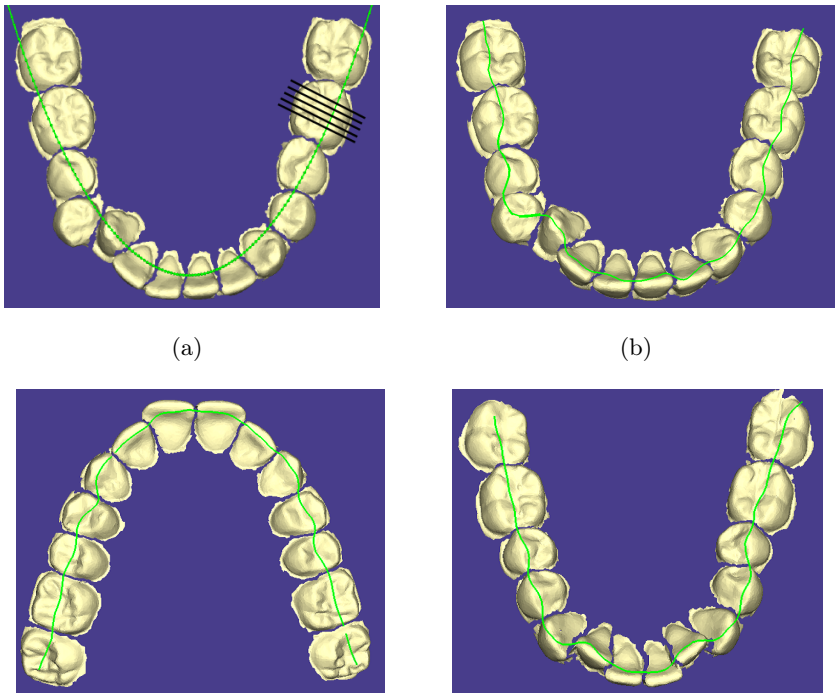
Computing medial curve of an arch

1. Project the triangular faces on each tooth surface to a set of triangles, F_{xy}, on the xy-plane.
2. Find a cubic curve C_{fit} that best fits the 2D projections of the vertices in F_{xy} (Figure 3.9(a)).
3. Sample the length of C_{fit} at suitably small intervals to get an ordered sequence, $P = p_1, p_2, \ldots, p_n$, of points on C_{fit}.
4. At each point $p_i \in P$, define a line l_i orthogonal to C_{fit} (Figure 3.9(a)). Find the median point, m_i, of the intersection points of l_i with F_{xy}. Output the medial curve as the piecewise-linear curve that connects the ordered sequence, $C_{med} = m_1, m_2, \ldots, m_n$, of medians (Figure 3.9(b)).

3.4.2 Computing planar cross-sections

Using the medial curve described above, we compute an approximation, L_{md}, of the mesiodistal line, of a given tooth by finding the best-fitting line segment for the section of the medial curve segment that goes through the tooth. We can also compute an approximation, L_{bl}, of the buccolingual line of the tooth by taking the line orthogonal to the mesiodistal line and passing through the centroid of the xy-projection of the tooth. We will refer to the approximation of the mesiodistal (resp. buccolingual) line as simply the *mesiodistal* (resp. *buccolingual*) *line*.

We define the *cross-section* of (the mesh of) a tooth object with respect to a plane as the piecewise-linear curve obtained by intersecting the plane with the triangular faces of the mesh. A plane that is parallel to both L_{md} and the z-axis and intersects the tooth object generates a *mesiodistal cross-section*. Let P_{md} be a set of such uniformly-spaced planes and let the set of corresponding cross-sections generated be $\mathcal{M}$. Similarly, a plane that is parallel to both L_{bl} and the z-axis and intersects the tooth object generates a *buccolingual cross-section*. Let P_{bl} be a set of such uniformly-spaced planes and let the set of corresponding cross-sections generated be $\mathcal{B}$ (see Figure 3.10).



(a) (b)

(c) (d)

Figure 3.9: Computation of a medial curve (shown green): (a) Initial fit of the arch with a cubic curve. (b) Computed medial curve for arch in (a). Additional examples of medial curves on an upper arch (c) and a lower arch (d).

3.4.3 Computing 2D curvatures of cross-sections

For any point p on a planar curve C, the curvature at p can be defined as the reciprocal of the radius of the osculating circle at the point, which is the largest circle tangent to the curve on the concave side of p [28]. The 2D curvature of C at p can also be defined as the rate of change, at p, of the angle between the tangent vector to the curve at p and the positive x-axis [28]. It measures how sharply the tangent to C rotates at the given point. Thus, the curvature at point p of the curve shown in Figure 3.11 can be defined as,

$$\kappa(p) = \lim_{\Delta s \to 0} \frac{\Delta\theta}{2(\Delta s)}, \tag{3.1}$$

(a) (b) (c)

Figure 3.10: Examples of planar cross-sections on the tooth surface (shown in cyan). (a) Buccolingual cross-sections. (b) Mesiodistal cross-sections. (c) Buccolingual cross-section of a molar.

where Δs is a small length along the curve and $\Delta\theta$ is the angle between the tangents at points $p + \Delta s$ and $p - \Delta s$ along the curve.

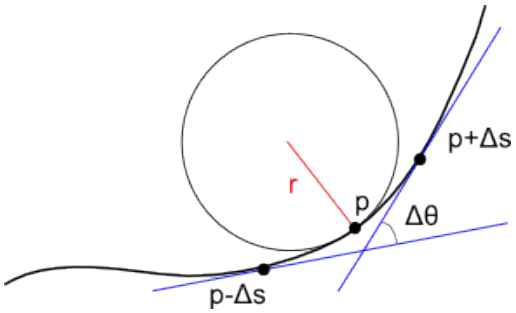


Figure 3.11: Curvature of point p on a planar curve is defined rate of change, at p, of the angle between the tangent vector to the curve at p and the positive x-axis. It is also defined as $1/r$, where r is the radius of the osculating circle at p (shown in red).

The curves in real-world applications may not have a reasonable analytical form and are often represented as a sequence of points sampled along the curve (Figure 3.12). Some examples of these are the contours in images, boundaries of shapes and, as in our case (Section 3.4.2), the intersection of a plane and a tooth surface. For these piecewise-linear curves, we have to approximate the curvature at its sampled points. A natural solution to this would be to first get a smooth spline approximation to the given curve

followed by computing the curvature analytically using the first and second derivatives at each sampled point. However, this becomes an expensive operation for large numbers of points and as we need to compute the curvatures for a large number of curves, we need an alternative method to compute approximate curvatures efficiently.

In analogy with Equation 3.1, the approximate curvature of the point p_i (see Figure 3.12) on a planar curve can be evaluated as

$$\kappa(p_i) = \frac{\Delta\theta}{|p_{i-1}p_i| + |p_ip_{i+1}|}, \tag{3.2}$$

where p_{i-1} (resp. p_{i+1}) is the previous (resp. next) point with respect to p_i on the curve C and $|p_ip_j|$ is the length of the edge from p_i to p_j. If the curve is not closed (i.e., $p_{start} \neq p_{end}$), we define the curvatures $\kappa(p_{start}) = \kappa(p_{end}) = 0$. Also, we use the signed angle ($\Delta\theta$) to define positive and negative curvatures at points. The points corresponding to the positive (resp. negative) curvatures form the convex (resp. concave) regions on the curve with respect to the plane containing the curve (Figure 3.10(c)).

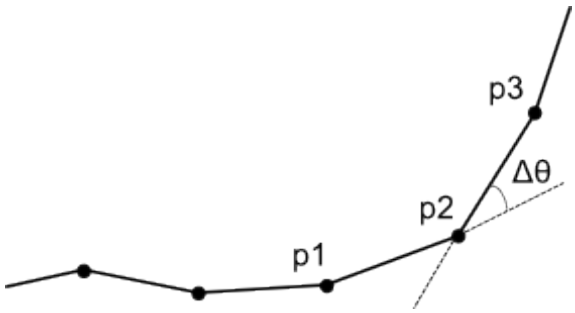


Figure 3.12: Planar curve represented as a sequence of points.

3.4.4 Clustering of high-curvature regions

Given the buccolingual and mesiodistal cross-sections of a tooth, we find the vertices on the curves for which the magnitude of the 2D curvature is greater than a specified threshold, T_{curv}. The sign of the curvature threshold may be positive or negative depending on the targeted feature (e.g., ridges (resp. grooves) have high positive (resp. negative) curvatures). The set, S, of high-curvature vertices resulting from the thresholding serves as the input to the clustering algorithm.

Our goal behind clustering is to aggregate the vertices in S into connected components of high-curvature vertices that correspond to features of interest. The extracted clusters reveal these features and also cause noise regions to be grouped into small isolated components. The latter can be identified and deleted easily using a threshold, T_{noise}, on the number of noise vertices.

We compute the connected components as follows. Initially all vertices in S are taken to be singleton components. Starting with this, we repeatedly merge a pair of components if the closest distance between them is less than a specified threshold, T_{ccd}. The closest distance between two components is defined as the minimum of the pairwise distances between their corresponding vertices.

The clustering method requires three user-specified threshold parameters: T_{curv}, T_{noise}, and T_{ccd}. We have observed experimentally that, for each feature type of interest, values of these parameters can be found that work across a broad range of patient datasets. Thus, the parameter values can be predetermined and our algorithm does not, in general, require user input during routine operation. Section 3.5 below, which describes how the methods of this section can be used to identify various features, provides the specific parameter values that our implementation uses.

3.4.5 Discussion

The features of interest to us could be defined naturally in terms of the curvatures at the mesh vertices (relative to the underlying 3D surface) [17, 23, 35, 40, 54]. For example, cusps, incisal edges, and ridges consist of clusters of vertices of high positive curvature (convex regions), whereas grooves consist of clusters of vertices of high negative curvature (concave regions). Hence, it is natural to wonder if 3D curvature analysis can be used for feature recognition.

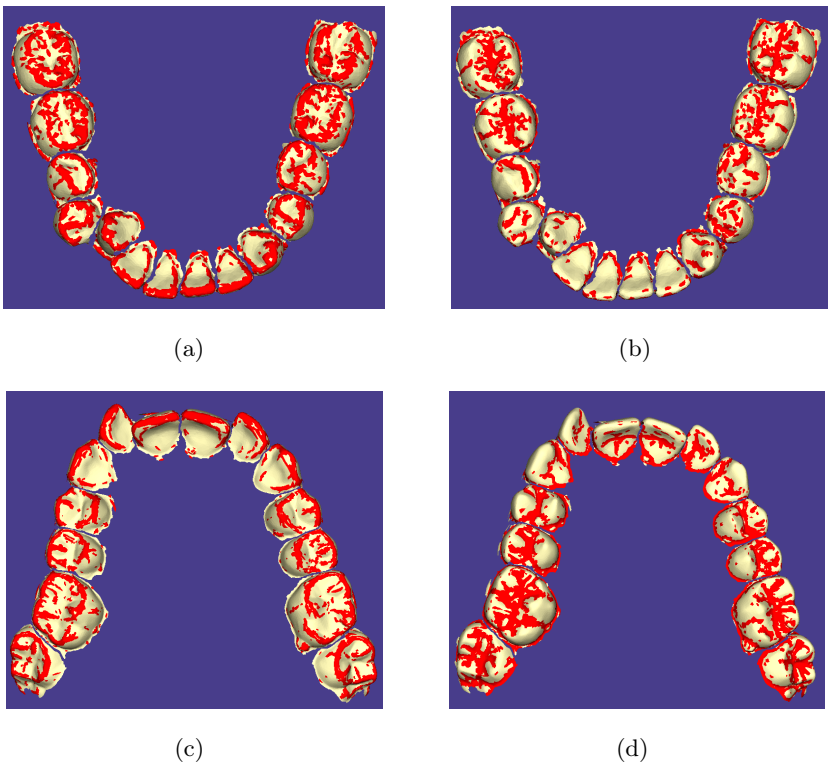
We implemented this approach and found that, unfortunately, it did not perform well on most datasets. There are several reasons for this:

- Tooth surfaces tend to exhibit sharp changes in curvature over small neighborhoods. This makes it difficult to compute 3D curvatures reliably and use them to identify partitioning ridges and valleys.

- For each feature type, the 3D curvature threshold settings vary quite considerably

from one model to the next (in contrast to the situation in the 2D approach); this requires users to set the thresholds on a case-by-case basis and imposes an undue burden.

- The presence of spurious ridge- and groove-like structures on the occlusal surface and very close to the target features makes the identification and cleanup of actual features difficult. We tried using skeletonization methods for 3D meshes that are based on morphological operators [50], but these did not perform well given the nature of the problem. Moreover, as Figure 3.13 shows, the sizes of these noise regions are comparable to those of actual features, which makes identification of true features extremely difficult.



(a) (b)

(c) (d)

Figure 3.13: Examples of poor feature recognition using 3D surface curvatures on a lower and an upper arch. The sizes of noise regions are comparable to those of actual features.

It is for these reasons that we chose to develop the 2D method, based on cross-sections, outlined earlier. It is easy to see that using a large number of cross-sections effectively reproduces the entire tooth surface to high accuracy. Since the cross-sections are generated by a set of parallel planes, this process can be thought of as a "guided" reconstruction of the 3D surface. An advantage of this is that we now have the flexibility to select only those parts of the cross-sections that yield interesting information. For example, on a buccolingual cross-section of a molar, the outermost convex regions corresponds to the mesial and distal cusp ridges and the innermost concave regions correspond to the grooves (Figure 3.10(c)). The notions of "outermost" and "innermost" are easy to define relative to the natural directionality provided by a 2D curve, but it is not clear how to do this on a surface. Furthermore, the parallel orientation of the cross-sections makes it easy to stitch together the portions of features found on each cross-section into a 3D feature.

3.5 Feature identification

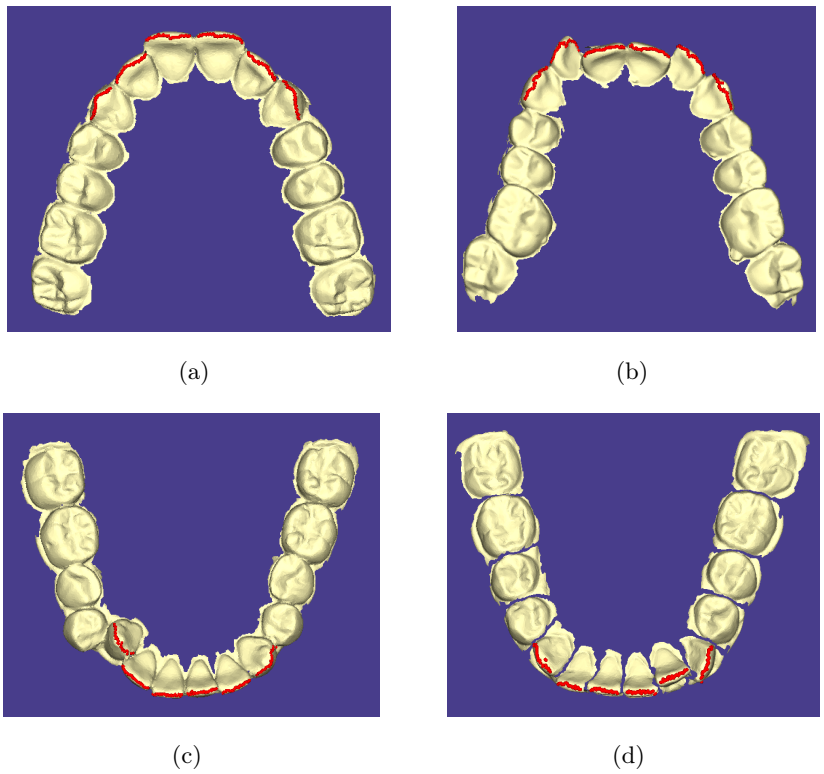
This section describes how the techniques of Section 3.4 can be used to automatically identify features other than cusps, such as incisal edges, grooves, marginal ridges, and occlusal surface boundary. (Cusp identification, via a different approach, was discussed in Section 3.3.)

3.5.1 Incisal edges

These can be extracted as a cluster of vertices of high positive curvature (i.e., convex regions) on the buccolingual cross-sections of the anterior teeth. The results of our incisal edge detection algorithm are shown in Figure 3.14 for two pairs of upper and lower arches. Note that the algorithm performs correctly even when the tooth is in a rotated position from the global archform (Figures 3.14(b) and 3.14(d)).

As mentioned in Section 3.4.4, there are three parameters that control the clustering process. Their specific values for incisal edge identification are as follows: $T_{curv} = 0.6$ (the curvatures are normalized to the range $[-1, 1]$ here and later), $T_{noise} = 1/10$th of the number of buccolingual cross-sections and $T_{ccd} = 0.5$ mm. (We reiterate that these parameters work well on all models tested here and later, and do not require adjustment

by the user.)



(a) (b)

(c) (d)

Figure 3.14: Incisal edges (shown red) on anteriors identified using buccolingual cross-sections. Upper arches shown in (a) and (b); lower arches in (c) and (d).

3.5.2 Grooves

Grooves are found on the posterior teeth and are extracted as a cluster of vertices of high negative curvature (i.e., concave regions) in the middle-section of the buccolingual cross-sections (Figure 3.10(c)). As mentioned in Section 3.2, there are many types of grooves on the posteriors, especially the molars. We are interested in the central grooves that run along the mesiodistal line of the tooth. Figure 3.15 shows the results of our groove identification algorithm on two pairs of upper and lower arches. As expected, the grooves on the upper arches have more variation than the ones on the lower arches and consist of multiple "branches". For the purpose of treatment planning, one may choose

to approximate these grooves by a straight line segment or a curve.

The specific parameter values used for groove identification are as follows: $T_{curv} = -0.4$, $T_{noise} = 1/10$th of the number of buccolingual cross-sections and $T_{ccd} = 0.5$ mm.

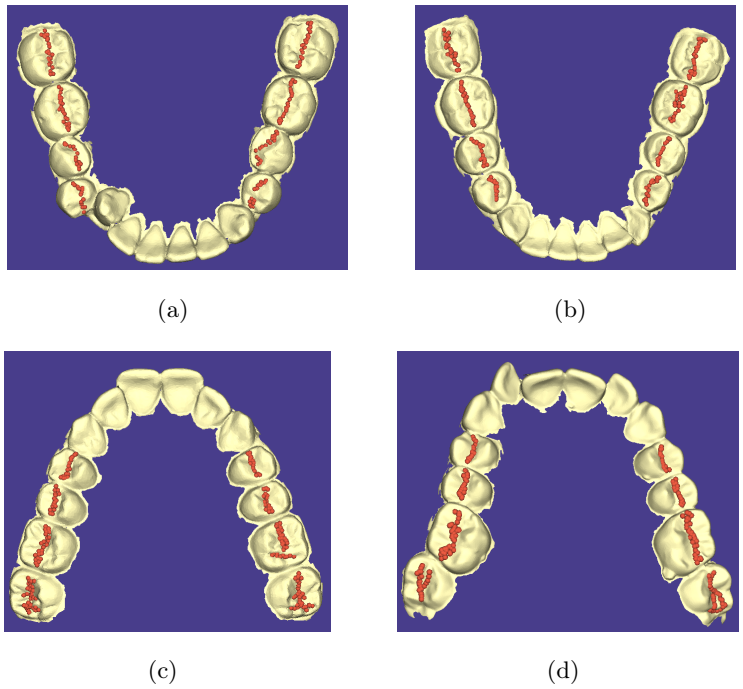
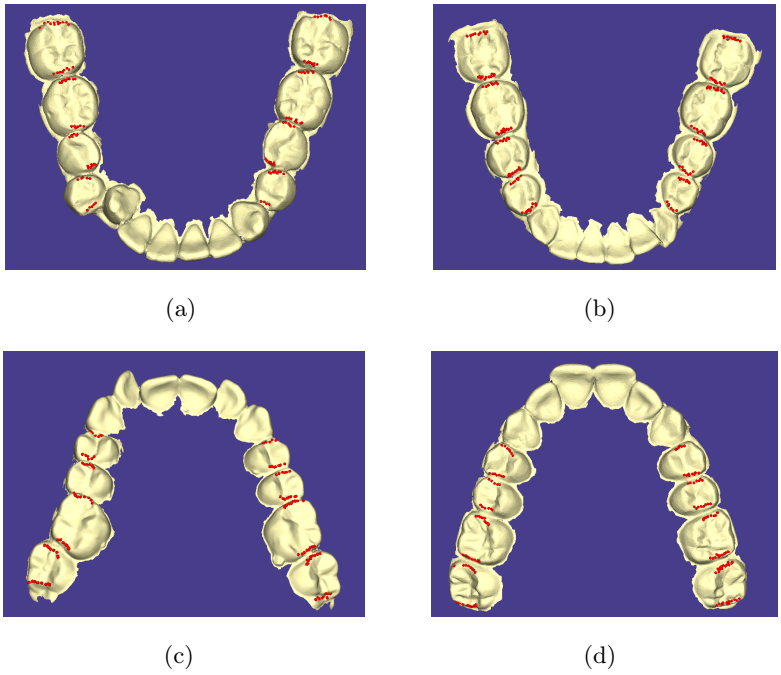


Figure 3.15: Central grooves (shown brown) on posteriors identified using buccolingual cross-sections. Note the difference in the groove patterns between the lower arches ((a), (b)) and upper arches ((c), (d)).

3.5.3 Marginal ridges

These are extracted as a cluster of vertices of high positive curvature on the mesiodistal cross-sections of the tooth surface. Figure 3.16 shows the results of our algorithm on two pairs of upper and lower arches. The marginal ridges on the lower arch premolars are difficult to identify because they are not easily separable from the cusp ridges and also may not be properly aligned with the mesiodistal line (see Figure 3.16(a) and (b)). Also, the distal marginal ridge on the last available molars often cannot be accurately identified because of the inherent ambiguities in the structure of these teeth.

The specific parameter values used for marginal ridge identification on an upper arch is $T_{curv} = 0.3$, $T_{noise} = 1/10$th of the number of mesiodistal cross-sections and $T_{ccd} = 0.5$ mm. The mesial and distal walls on the lower arch posteriors are higher than those on the upper arch and, thus, we set $T_{curv} = 0.6$. The threshold for T_{noise} and T_{ccd} are the same as for the upper arch.



(a) (b)

(c) (d)

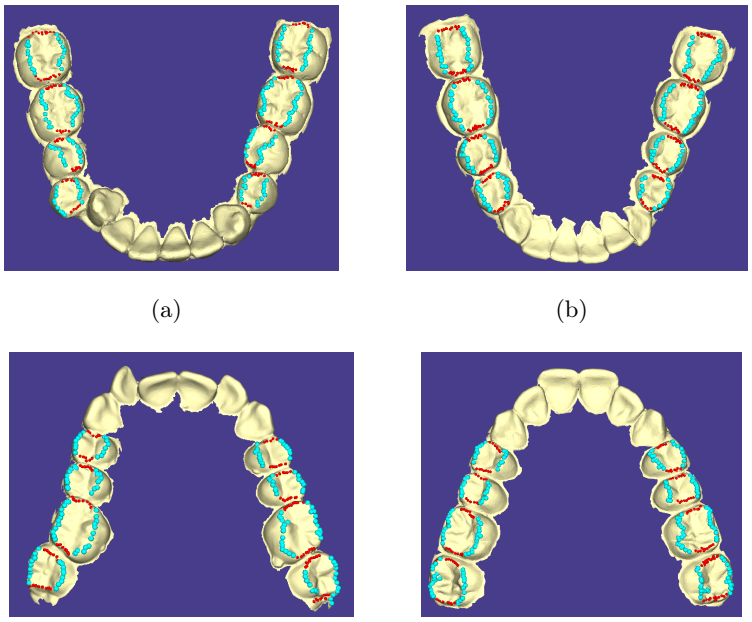
Figure 3.16: Marginal ridges (shown red) identified on posterior teeth using mesiodistal cross-sections.

3.5.4 Occlusal surface boundary

The occlusal surface boundary of a posterior tooth is defined by the marginal ridges on the mesial and distal sides, and the cusp ridges on the buccal and lingual sides (Section 3.2). Section 3.5.3 describes how to identify the marginal ridges. Cusp ridges on the buccal and lingual sides are also identified via buccolingual cross-sections, employing an approach similar to that for the incisal edges, but using both the buccal and the lingual sides of the cross-sections. Figure 3.17 shows the result of our cusp ridge identification

algorithm.

The specific parameter values used for the cusp ridge identification are $T_{curv} = 0.5$, $T_{noise} = 1/10$th of the number of buccolingual cross-sections, and $T_{ccd} = 0.5$ mm.



(a) (b)

(c) (d)

Figure 3.17: Occlusal surface boundary identification on posterior teeth using buccolingual cross-sections. The occlusal surface is bounded by the marginal ridges found earlier (shown red) and the cusp ridges on the buccal and lingual sides (shown cyan).

3.6 Identification of derived features

This section describes the estimation of derived features (archform and occlusal plane) from the already identified intrinsic features (cusps and incisal edges).

3.6.1 Archform

The archform is defined as an appropriate smooth curve through the incisal edges of anteriors, canine cusps, and the buccal cusps of molars and premolars. There have been

several attempts to mathematically describe the archform, the most relevant ones being based on *catenary curves* [18], *cubic splines* [11, 15] and *beta functions* [12]. However, due to the large variation in the shape of the archform among humans with normal occlusion (square, tapered, ovoid, etc.[45]), it is difficult to formulate a single mathematical expression to describe all archforms.

Moreover, in most situations, an orthodontist is very likely to make a few minor changes to the archform during virtual treatment planning. Thus, our approach is to compute an initial approximation to the archform using the intrinsic features and allow the user to modify it using five control points along the archform curve (Figure 3.18).

We use the experimentally-derived equation in [12] to compute the initial approximation to the archform as:

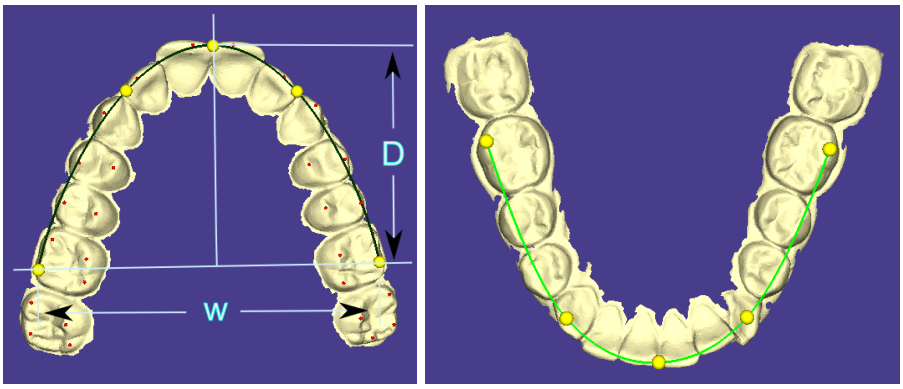
$$Y = 3.0314 \cdot D \cdot \left[\frac{1}{2} + \frac{X}{W}\right]^{0.8} \left[\frac{1}{2} - \frac{X}{W}\right]^{0.8}, \tag{3.3}$$

where W is the distance between the distobuccal cusps of the first molars and D is the perpendicular distance between the line joining these cusps and the most anterior point between the two central incisors (this is computed as the midpoint between the incisal edges of central incisors). See Figure 3.18(a). We note that our definition of W follows common orthodontic practice and is slightly different than the one in [12], where W is measured as the distance between the distobuccal cusps of the second molars. This change is motivated by the fact that the second molars may not have completely erupted in many patients.

As mentioned above, our software interface provides five control points (Figure 3.18) to modify the archform. Once one of these control points is moved, the archform latches on to the five control points and is then computed as a cubic spline through these control points, which is another popular way to describe the archform in dental literature. (The control points can be moved symmetrically, in pairs or independent of one another.)

3.6.2 Occlusal plane

The occlusal surface is the surface of contact between teeth from opposing arches. Generally, even though the occlusal surface is not planar in normal-occlusion arches, a flat occlusal plane is set as the treatment goal in orthodontics [6]. This is based on a natural



(a) (b)

Figure 3.18: (a) An approximation to the dental archform (shown green) and the parameters of the beta function that describe it (shown as W and D). The archform can be modified using the five control points (shown yellow). (b) Archform of a lower arch.

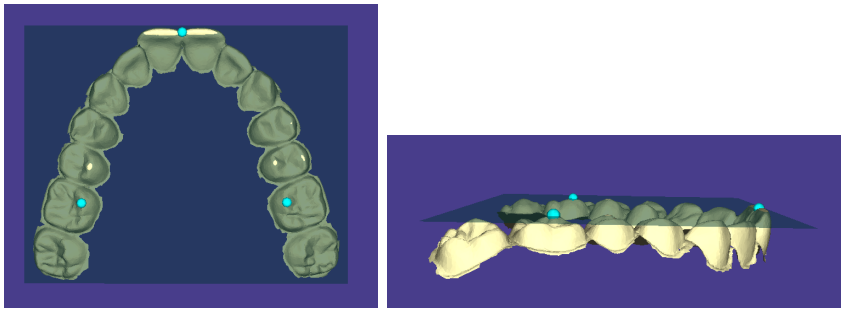
tendency of the teeth to rotate away from this plane (thus deepening the *curve of Spee*, which is a smooth curve connecting the posterior buccal cusps, canine cusps and the incisal edges on a single side of the arch [55]).

We approximate the occlusal surface as a plane through the mesiolingual cusps of the first molars and the midpoint of the incisal edges on the two central incisors (Figure 3.19). As in the case of the archform, a user may want to adjust the orientation and the height of the occlusal plane and we provide the required interface in our system to accomplish this.

3.7 Evaluation

This section describes implementation of the feature identification algorithms within our software tool and the results of a quantitative evaluation of these algorithms on clinical data.

The software tool consists of a simple, intuitive user interface that allows the user to load dental models and compute their features using the techniques described above. Figure 3.20 shows a screenshot of the feature identification module of the tool and



(a) (b)

Figure 3.19: Two views of the occlusal surface of an arch approximated as a plane through the mesiolingual cusps of the first molars and the midpoint of the central incisors (shown cyan).

describes its functionality. Note that the initial computation of features does not require any input from the user; however, a user can subsequently modify the computed features using buttons shown on the upper portion of the rightmost window in Figure 3.20.

3.7.1 Results

The techniques for feature identification described in Sections 3.3 and 3.5 were evaluated on a set of 8 lower and 8 upper arches. These presegmented arches were selected from a database of actual cases treated at the University of Minnesota's dental clinic. The models were obtained through the SureSmile® software [52] which provides a large number of models of real-world cases that are useful for the subsequent study of orthodontic alignment of teeth using the features computed here (discussed in Chapter 4). They cover a range of difficulty including malocclusions, improperly formed teeth, and varying anatomy on posteriors.

All the features identified by algorithms described earlier were evaluated by an experienced orthodontist. The criteria for evaluation was the accuracy of the detected feature with respect to the anatomy of the corresponding tooth and its purpose in the orthodontic alignment planning. For example, often the lower molars may have 4 or 5 prominent cusps all which must be detected. On the other hand, often due to wear, some cusps may not be prominent enough on a tooth. Thus, these will not be included

Feature Type	Partial	Incorrect	Missing	Correct	Total	% Correct
Cusps	3	3	23	383	412	92.9
Marginal ridges	8	6	1	241	256	94.1
Cusp ridges	8	5	0	243	256	94.9
Incisal edges	2	0	0	94	96	97.9
Grooves	1	1	0	126	128	98.4

Table 3.1: Table showing the aggregated results of the evaluation of features on 16 different dental arches.

in the target set of cusps. Similar issues exist with other features as well.

The errors in features were classified into three categories: partially detected (e.g., only a portion of a ridge was identified), incorrectly detected (the feature was found in an incorrect part of a tooth) and missing (feature was not identified at all, e.g., an unidentified cusp). The aggregated results of the evaluation on the 16 input arches are shown in Table 3.1.

Incisal edges and grooves: The identification of incisal edges and grooves is extremely accurate as seen in the last two rows of Table 3.1. Each arch had 6 incisal edges (4 incisors and 2 canines) and 8 grooves (4 posterior teeth on both left and right sides). The difficulty in incisal edge identification is primarily due to malocclusions and tooth wear on the incisors. Similarly, groove identification is sometimes difficult due to irregular anatomy and tooth wear. Also, we are interested in approximating the central grooves on the posteriors (along the mesio-distal axis), which are quite curved on the lower premolars and the upper molars.

Cusps: Table 3.1 shows the results of cusp identification. The number of true cusps varies across different arches, and the majority of errors is due to unidentified cusps on second molars (Figure 3.21(a)). This is primarily due to incorrect orientation and insufficient anatomical details. It was observed that more than half of these errors came from only three different arches.

Occlusal surface boundary: Table 3.1 shows the results for identification of marginal and cusp ridges which constitute the occlusal surface boundary. Each arch has 8 posterior teeth with 2 marginal and 2 cusp ridges. As with the other features, it is observed that most of the errors are from the second molars due to similar reasons (Figure 3.21(b) and 3.21(c)).

3.8 Conclusion

This chapter presented techniques to automatically identify tooth surface features in noisy and incomplete dental models. Our methods use buccolingual and mesiodistal cross-sections of the tooth surface to facilitate a "guided" extraction of incisal edges, central grooves, marginal ridges, cusp ridges, and occlusal surface boundary. We also described a watershed-based cusp identification algorithm. Next, Chapter 4 describes the application of these features in the alignment step of the virtual treatment planning process.

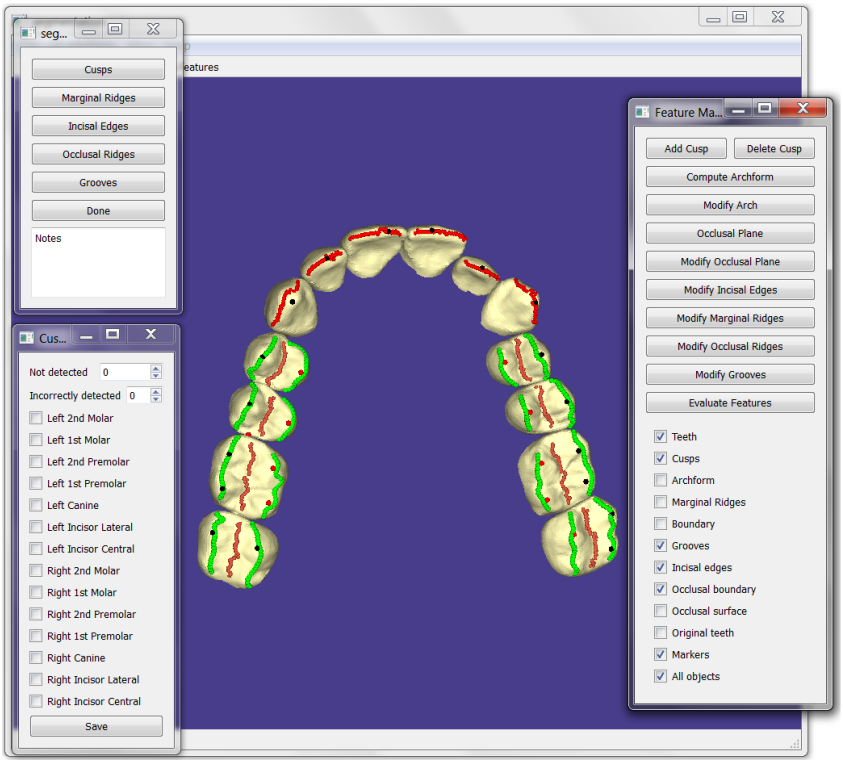
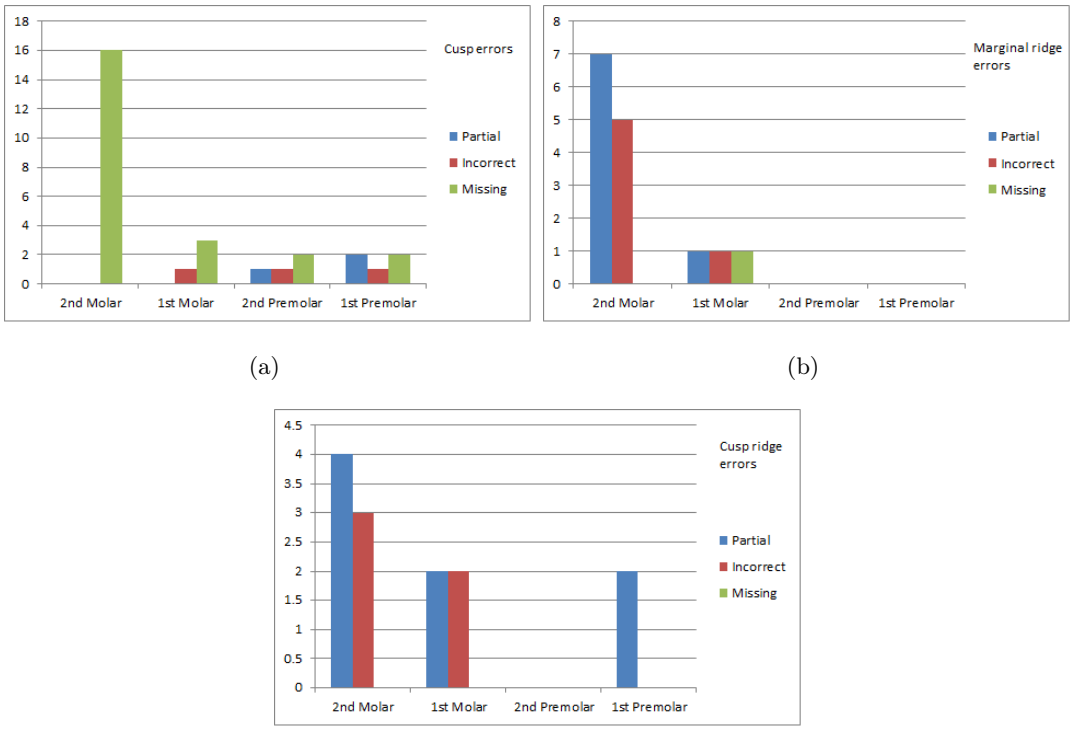


Figure 3.20: Screenshot of the feature evaluation tool. A model is shown along with its computed features in the central window. The checkboxes on the rightmost window are used to show or hide the different feature types. Each button on the upper-left window, when clicked, selects a type of feature to be evaluated using the window shown on the lower-left side. The expert can record the number of errors corresponding to that feature type and specify the teeth contributing to the errors using the checkboxes. The expert's input are recorded in a file for later analysis.



(a) (b)

(c)

Figure 3.21: These charts show the errors in feature identification with respect to the different tooth types (shown on the x-axis): (a) Errors in cusps, (b) Errors in marginal ridges, and (c) Errors in cusp ridges.

Chapter 4

Virtual alignment of dental arches in orthodontics

4.1 Introduction

Given a 3D dental model of the upper and lower arches, Chapter 2 and Chapter 3 described techniques to identify the individual tooth objects in these arches and to compute certain dental features on the surface of these teeth. This chapter addresses the problem of establishing proper occlusion (i.e., *alignment*) of teeth in a virtual digital setting *before* the actual treatment starts. This is accomplished by a physically-based simulation of the alignment process that makes crucial use of the computed features.

Our goal is to *automatically* compute the *optimal occlusion*, which we take to mean an arrangement of teeth in a given pair of opposing arches that best satisfies the constraints of an ideal occlusion, as envisioned by an experienced practitioner. There can be significant variability in dental anatomy across different patients, including differences in the tooth size and shape, crowding, type of malocclusion, tooth condition (missing or restored teeth), and the extent of improper contacts. Therefore, across patients, this results in differences in what constitutes an optimal alignment, i.e., differences in the quality of the dental occlusions that can be realized from one patient to another. The alignment of teeth has an *intra-arch* (resp. *inter-arch*) component that aligns teeth with respect to other teeth on the same (resp. opposing) arch.

Virtual planning makes it possible to study the optimal occlusion outcomes of different scenarios in planning, e.g., the optimal occlusion outcome neglecting an existing tooth (extraction). Another example is to optimize occlusion for one of the occlusion classes, Class I, II, or III (described in Section 4.3.2). Virtual planning also yields considerable benefits when training of dental students as it allows the trainees to observe and emulate different treatment outcomes for a given case, as well as learn about the effectiveness of the different treatment plans on cases already solved by experts.

4.1.1 Challenges in virtual alignment

The computation of the optimal occlusion of a given set of teeth is difficult for several reasons, as discussed below.

Evaluation of dental occlusion: The orthodontic literature contains very few robust, quantitative metrics to evaluate the quality of the dental occlusion in a given arch [16, 20]. Such a metric is important in determining if an intended correction to the arrangement of teeth brings about a change in the overall functionality. A widely used evaluation metric is the American Board of Orthodontics (ABO) model grading system [47]. (The ABO grading system is described in detail in Section 4.3.2.) This system is used to evaluate the test cases in orthodontic board exams. It presents several types of common errors that can be evaluated to get an objective score based on well-agreed-upon constraints. For example, the absolute difference in height of the so-called marginal ridges of the posterior teeth (defined in Section 3.2.2) is mapped to an integer score in the interval [0,2]. Such a scheme is useful, but is very coarse, leaves a large room for approximations, and does not capture many of the finer aspects of orthodontic occlusion. There are detailed studies on the ideal arrangement of teeth in terms of the relationship to the archform, adjacent teeth, occlusal plane and the teeth from the opposing arches [6]. However, there are no established metrics to measure the relative "goodness" of a given arrangement with respect to a normal one.

Algorithms for computing occlusion: There are no known practical algorithmic techniques to search for the optimal occlusion for a given input case. This means that most of the occlusions established currently are primarily a result of "try-and-refine"

procedures guided by the experience of the user. This is very tedious and has a high risk of leading to sub-optimal occlusions that may require costly re-treatment.

Properties of ideal occlusion: As mentioned above, there are many well-agreed-upon constraints that must be satisfied for a good dental occlusion. However, the relative importance of these constraints is not established and, as a result, the opinion of different practitioners regarding this may differ. This is important because many of these constraints are essentially of a conflicting nature and cannot be simultaneously satisfied in real-world cases. For example, the marginal ridges of posteriors must be at the same vertical height. However, ideal occlusal contacts require a posterior tooth to be in tight contact with the teeth from the opposing arch. Thus, there is an intra-arch and an inter-arch constraint that are in conflict with respect to the height of the teeth. Also, there is not much statistically-observed information available on which of the constraints are the most important in defining the course of alignment in real-world cases.

Special cases in occlusion: Often the types of occlusion in orthodontics are classified as Class I, II, or III, based on the relationship of the posterior teeth of the opposing arches (see Section 4.3.2). These classes are also determined by whether a premolar is extracted from the upper or the lower arch. It is important to know which of these classes apply to a given case, as the alignment plan is different for each of them. However, it is, in general, difficult to find the occlusion class of a given model automatically.

Computational issues: There are additional computational issues in the virtual setting as compared to a physical setup environment. For example, we need real-time collision detection and avoidance between the teeth in a virtual world, as they move from their initial positions to their final positions during the treatment simulation, whereas this is naturally "enforced" in a physical setup [13]. Also, the search-space of the possible alignments is much larger in a virtual environment. This is due to the large number of potentially sub-optimal arrangements generated by an unguided, automatic movement of teeth compared to the physical setup performed by an expert.

4.1.2 Scope of the alignment problem

There is considerable variation and sophistication in the manual orthodontic treatment planning techniques used in the clinic. In this thesis, we consider the basic problem of aligning teeth from two opposing arches so that an optimal occlusion can be achieved on a broad range of cases. (Our notion of optimality is as stated at the beginning of Section 4.1.) We note that the use of advanced techniques such as tooth extraction (commonly premolar), trimming of crowns (commonly anteriors), etc. (see Class I, II, III description in Section 4.3.2) are beyond the scope of the thesis.

4.1.3 Organization of the chapter

Section 4.2 describes the related work in this field. Section 4.3 describes the some known approaches to characterize the problem of alignment in orthodontics. These ideas are used to model the alignment problem in terms of satisfying certain dental constraints in Section 4.4. An implementation of this model along with a simulation algorithm is described in Section 4.5. Finally, Section 4.6 presents the software implementation and the experimental results obtained on real-world datasets.

4.2 Related work

As mentioned above, the traditional approach of alignment is to re-position the teeth on a stone or plaster model that is mounted on an articulator. However, tools such as SureSmile® [52] and *e*model® [53] provide the ability to manipulate 3D tooth models in a virtual environment. These computer-aided tools are becoming increasingly popular because of the facilities they provide to interact with, visualize, and animate treatment plans. Also, these tools allow long-term, robust storage of models and accurate measurements of tooth features and dimensions. However, the alignment process itself has not changed much and still involves users manipulating the position and orientation of individual teeth, mostly using a mouse-based interface.

We mention briefly two areas where problems related to dental occlusion arise. In prosthodontics, the goal is to reconstruct portions of an arch through the use of dentures and implants. The key task here is to align ideal tooth objects optimally along a predetermined archform which can be chosen based on the alveolar bone in the mandible [1].

The problem in orthodontics is different as one has to work with patient-specific teeth (which may not be ideally-shaped) and the current archform determined by their arrangement. Thus, the current archform cannot be altered arbitrarily and must respect certain biological limits.

In craniomaxillofacial surgery, the goal is to establish the optimal dental occlusion of a given pair of *unsegmented* upper and lower arches [13, 21]. This involves computing the translations and rotations through space that would "fit" the rigid upper and lower arch meshes together to achieve maximum intercuspation (maximum occlusal contact area). Most of these techniques try to align arches based on some intrinsic features on the teeth, e.g., the grooves and marginal ridges on the upper arch are aligned with the buccal cusp ridges and incisal edges on the lower arch in [13]. This is different from the problem we consider here, where an individual tooth can move within the arch. Also, in orthodontic cases, the optimal maximum intercuspation is not known beforehand and is related to finding the optimal alignment of teeth. For example, given a pair of opposing arches that have good intra-arch alignment (no malocclusions), one can maximize the contact area between opposing arches. However, in the presence of orthodontic errors these criteria may not apply.

Another approach is to compute a close surface match of the opposing arches by moving the arches through space. This usually involves minimizing some function of the distance between the two occlusal surfaces of opposing arches [13]. As noted in [13], this is effective if the two surface to be matched are close to each other, in which case such a minimization converges to a very close fit quickly. However, this technique requires establishing correspondences between the vertices of the two meshes and the errors due to this may be very large when the meshes are far apart. Also, wrong correspondences can lead to a locally optimal arrangement wherein a tooth is "trapped" among other teeth in a sub-optimal configuration. Note that this can be a frequently-occurring situation in orthodontic cases where there is essentially some amount of crowding and malocclusions to be expected.

All the techniques mentioned above are essentially "global" (single optimization function) and work on the entire arch as an individual unit, ignoring the "local" effects in the neighborhood of each tooth, thus, being sub-optimal by nature for the class of our alignment problems. Also, they completely ignore the intra-arch alignment which is very

significant (e.g., the upper anterior teeth must be aligned to be esthetically pleasing). Thus, they fail to exploit the critical properties observed and distilled over the years regarding the correct dental occlusion in humans.

4.3 Formulation of the alignment problem

In this section, we first describe the concepts and ideas involved in defining the correct occlusion in normal and abnormal cases (Section 4.3.1). Next, we present a set of constraints based on the commonly observed occlusion errors in orthodontics, which can be used to quantify approximately the quality of occlusion in a given pair of arches (Section 4.3.2). Finally, we discuss the inter-dependencies and conflicts among the different constraints (Sections 4.3.3 and 4.3.4).

4.3.1 Orthodontics background for occlusion

This section describes the key ideas in understanding and characterizing the concepts for achieving correct alignment and occlusion. A more detailed discussion of other influences such as the underlying alveolar bone, roots of teeth, functioning of jaws, etc. can be found in texts such as [7, 55]. These properties determine the correct placement of a tooth vertically, horizontally, and along the archform; the placement depends on not only the tooth in question but also on many nearby teeth from the same and opposing arch. A study of all these factors is beyond the scope of the thesis. Here, we consider the problem of occlusion with respect to the position and surface features of teeth only.

Characterization of normal occlusion

There have been many studies to get an accurate characterization of correct dental alignment in humans. The main purpose of these studies was to find a set of properties that are applicable to a wide variety of normal cases. Early work [7] established the correct occlusion in terms of the position of the upper first molar. All the normal cases were observed to have the mesiobuccal cusp of the upper first molar align vertically with the buccal groove of the lower first molar. The first molars are large teeth that are among the first permanent teeth to erupt and, thus, influence the placement of other teeth on the arch. They also determine the extent of separation of the jaws (while open)

and are the most consistent in assuming their correct positions on the arch. Thus, it was hypothesized that if the first molars could be locked in correctly, then this would lead to the correct relationship between other anterior and posterior teeth. However, it was noted that in spite of the importance of the first molars, their correct placement still left enough room for some abnormal occlusion to occur. These were the Class I, II and III relationships defined by Angle [7] (see Section 4.3.2).

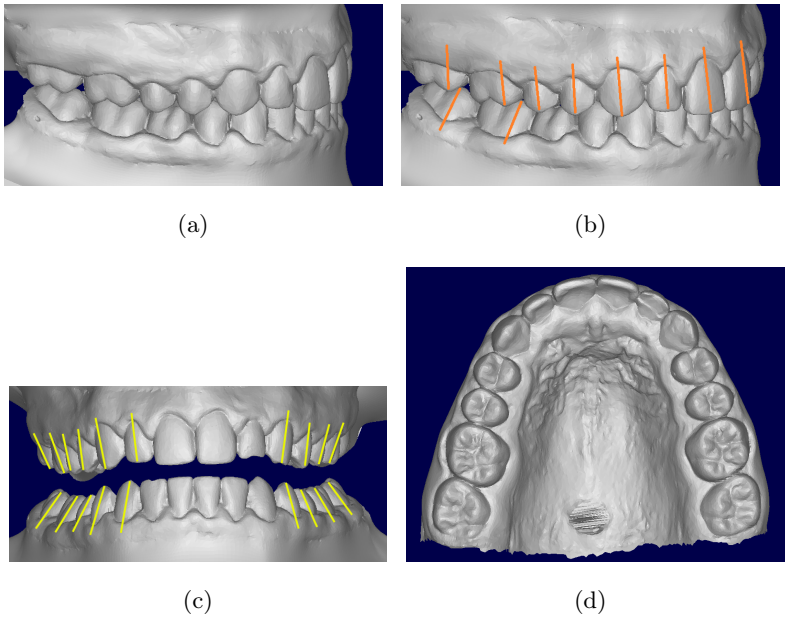


Figure 4.1: Important keys to normal occlusion: (a) Molar relationship. (b) Crown angulation (shown orange). (c) Crown inclination (shown yellow). (d) An arch with no tooth rotations.

A more extensive characterization of correct occlusion was given by Andrews [6] as a set of six "keys" of normal occlusion observed in more than one thousand cases. These six keys were shown to be necessary and sufficient to describe the normal occlusion in a majority of the cases:

1. *Molar relationship:* As mentioned above, the upper first molar's mesiobuccal cusp occludes with the buccal groove of the lower first molar. The mesiolingual cusp of the upper first molar should make contact with the central groove of the lower

first molar. Also, the upper first molar should be angulated so that the distal marginal ridge aligns with the mesial marginal ridge of the lower second molar. Figure 4.1(a) shows an example of the correct molar relationship.

2. *Crown angulation:* We define the *long axis* of a tooth as the vertical line that passes through the approximate center of the buccal surface of the tooth crown (the crown of a tooth is the visible part of tooth above the gumline). Traditionally, the mid-developmental ridge on the buccal side of the anteriors and the premolars is taken to be the long axis. However, this ridge is difficult to compute as it may not be prominent enough, especially on lower incisors. In normal occlusion, the gingival end of the long axis of the teeth should be more distal when compared to their occlusal end. Note that an angulated tooth will take up more space along the archform than in the normal case. Figure 4.1(b) shows an example of the distal angulation of crowns.

3. *Crown inclination:* The crown inclination is the labiolingual or buccolingual inclination of the tooth's crown. When examined from the mesial or distal side, if the gingival end of the crown is more lingual (resp. labial or buccal) than the occlusal end, it is referred to as positive (resp. negative) inclination. The upper (resp. lower) anterior teeth should have a mild positive (resp. negative) inclination, whereas both the upper and lower posteriors should have a positive inclination. The actual angle of inclination varies with each tooth type. Figure 4.1(c) shows an example of crown inclination for some of the teeth.

4. *Rotations:* The teeth should be free of any rotations with respect to the hypothetical archform. This is important for the proper occlusal contact and functionality (e.g., aligned incisal edges are more efficient in cutting). Figure 4.1(d) shows an arch without any rotations.

5. *Tight contacts:* The actual occlusal contacts should be tight without any space between the occluding surfaces of opposing arches. Figure 4.1(a) shows an example.

6. *Occlusal plane:* In most cases the occlusal surface, where the teeth from opposing arches meet, ranges from a flat plane to a slight *curve of Spee* [55]. However, a flat

occlusal plane is recommended as the treatment goal given the fact that the curve of Spee increases naturally with growth and function.

Evaluation of orthodontic cases

Various schemes for the objective evaluation of the quality of occlusion have been proposed in the literature. The goal here is to objectively assign a score to the quality of the finished cases using certain measurements on the arches. Some of these are described in [16, 20]. A characteristic of such an evaluation is that the scores must be based on features on the tooth surfaces that can be computed easily and precisely. Also, they must be applicable to a large variety of cases and at the same time point out minor but important inadequacies in tooth positions. Based on extensive studies of thousands of cases over several years by experts, the ABO has established a scoring system to evaluate the quality of cases presented by students training in orthodontics [47]. This objective score is calculated from a set of intrinsic features on tooth surfaces (e.g., cusps, grooves, marginal ridges, incisal edges, etc.) that can be measured unambiguously in a majority of cases. The properties measured for scoring are very similar to the six keys to normal occlusion and are described below in Section 4.3.2.

Our approach to correct occlusion

Some of the techniques to capture the main factors in orthodontic treatment were described above. For example, Section 4.1 described factors defining normal occlusion, whereas Section 4.3.1 described the common errors that occur in abnormal occlusions. These factors are well-rooted in the experience of experts and are justifiable in terms of tooth functionality, anatomy, and esthetics. Thus, these factors eventually influence the current practice in treatment planning. Our search for techniques to automate alignment and re-establish dental occlusion uses these ideas to guide and formulate methods that work for a majority of cases. Our hypothesis is that the re-positioning of teeth to correct as many of the common errors as possible among those defined in the dental literature such as the ABO model grading system will lead to improved dental occlusions for the case under consideration.

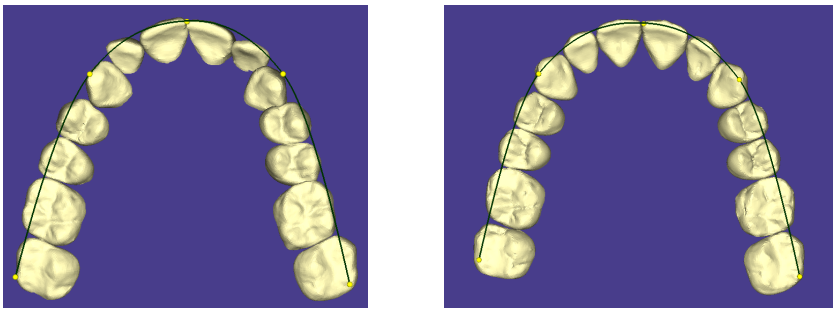
4.3.2 Constraints influencing occlusion

This section describes the formulation of the alignment problem in orthodontics in terms of various constraints that govern the quality of occlusion. As mentioned in Section 4.2, most of the related formulations of alignment problems treat the arches as individual rigid bodies that can only be translated and rotated through space as a single entity. Our formulation is to make the arches more "stretchable" by allowing individual teeth to move differently from one another. Thus, our approach can potentially explore many more arrangements of tooth positions and should lead to better outcomes.

We begin by describing the various constraints that must be satisfied by arches in correct occlusion, as given in the ABO model grading system. Some of the principles are modified to keep the computational model simple while not having a significant impact on the quality. It is worth observing these constraints in the context of the properties of normal occlusions [6].

Alignment constraint (AC): The alignment constraint requires that the incisal edges, canine cusps and the buccal cusps of posteriors on the arches are aligned along a smooth archform as shown in Figure 4.2. The original constraint also requires the grooves of the upper arch to be aligned along a smooth curve. However, we can handle the upper arch similar to the lower arch without affecting the correctness. This is arguably the most important constraint as it affects the primary layout of teeth along the alveolar bone, which in turn influences most other constraints. Also, it determines the esthetics on the anterior region and proper cutting (resp. chewing) function on the anteriors (resp. posteriors). The upper and lower lateral incisors and second molars contribute to the majority of errors, as was observed in real-world test cases.

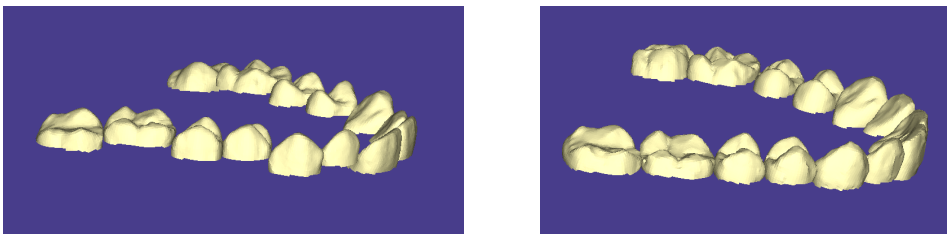
Marginal ridges constraint (MRC): Marginal ridges determine the vertical positioning of the posterior teeth with respect to their neighbors on the same arch. The marginal ridges of all properly formed teeth must be at the same height. Figure 4.3 gives an example of correct and incorrect marginal ridge heights. This ensures that the proper occlusal contacts will be made with the opposing arches, e.g., the distal marginal ridge of the upper first molar occludes with the mesial marginal ridge of the lower second molar as shown in Figure 4.1(a). The most common errors occur in the first and second



(a) (b)

Figure 4.2: Alignment constraint: (a) Malocclusion and incorrect alignment in both anteriors and posteriors. (b) Properly aligned (buccal) cusps, grooves and incisal edges. Also, premolars with undesirable rotations are shown in (a). (The proper alignment here and in Figure 4.3 and Figure 4.4 was obtained using our algorithm in Section 4.5.)

molars.

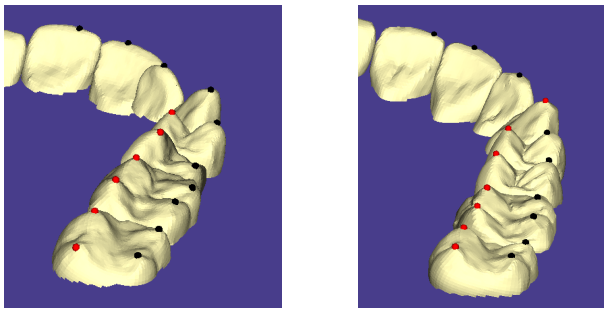


(a) (b)

Figure 4.3: Marginal ridges constraint: (a) Errors in marginal ridges. (b) Correct vertical positioning of teeth leads to the same height of marginal ridges.

Buccolingual inclination constraint (BIC): The buccolingual inclination of a tooth is measured as the angle between the line joining its buccal and lingual cusps and the occlusal plane. The crown inclination can be negative or positive as defined in Section 4.3.1. It is used to determine the crown inclination of the posterior teeth. Proper occlusion in maximum intercuspation requires that the buccal and lingual cusps of the

posterior teeth be at the same height relative to the occlusal plane. Figure 4.4 shows an example. The second molars contribute to the majority of inclination errors.



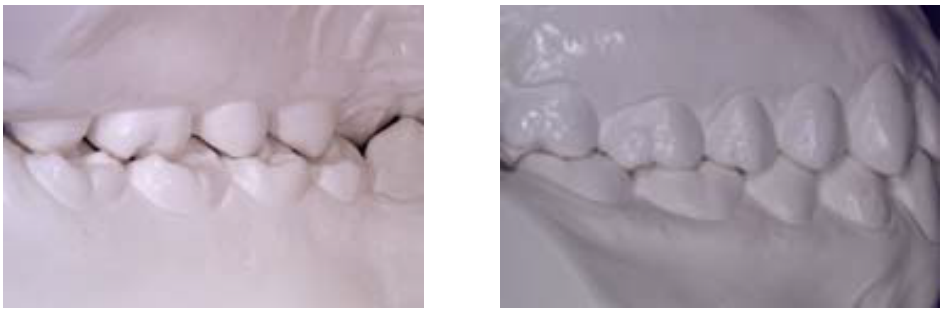
(a) (b)

Figure 4.4: Buccolingual inclination constraint: (a) Errors in buccolingual inclination at the posteriors. (b) Corrected inclination showing buccal and lingual cusps at the same height.

Occlusal contacts constraint (OCC): These constraints determine the extent of occlusal contact in the posterior teeth from opposing arches. Large occlusal contact areas on posteriors are vital to ensure that the chewing function is carried out well. Figure 4.5 shows examples of incorrect and correct occlusal contacts. This constraint is defined on the *functional cusps*, i.e., buccal cusps of the lower and lingual cusps of upper posteriors. The most common inadequacy of contacts is found in the second molars.

Occlusal relationship constraint (ORC): This is used to ensure the anteroposterior relationship of the posterior teeth. This includes the crucial first molar relationship discussed in Section 4.3.1. This constraint extends it and defines the vertical alignment, with respect to the archform, of each upper posterior with a corresponding interproximal contact point or groove of the lower arch. For example, as shown in Figure 4.6(a), the canine cusp of the upper arch must align with the interproximal contact points between the lower canine and the premolar. Similarly, the upper first premolar cusp must align with the interproximal contact point between the lower first and second premolars.

This also leads to the classification of malocclusion errors as a Class I, Class II,



(a) (b)

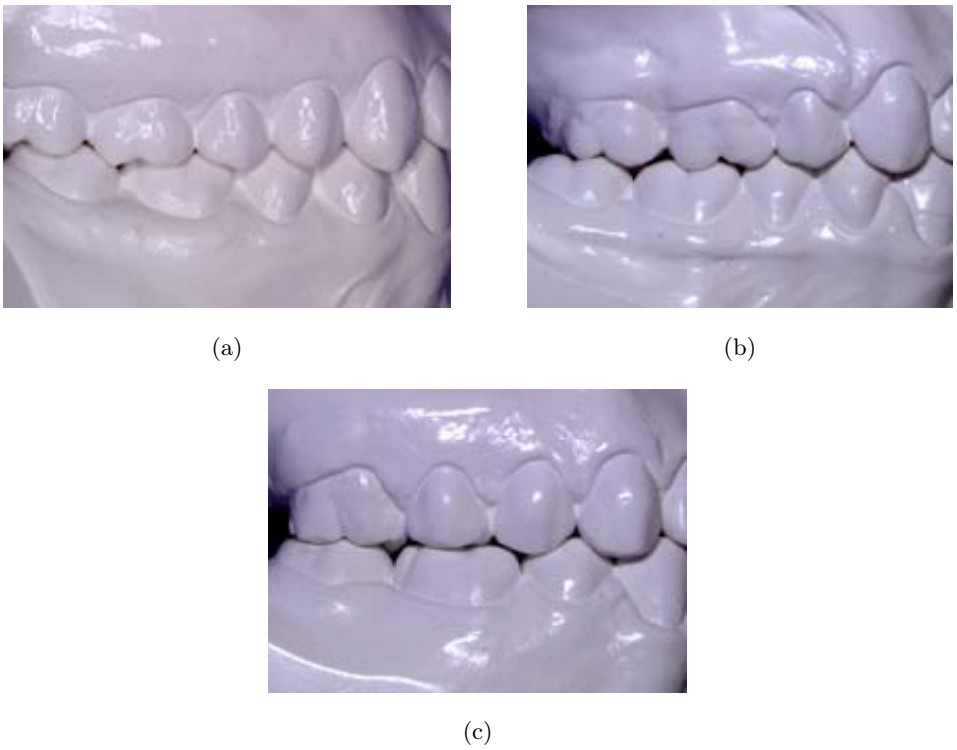
Figure 4.5: Occlusal contacts constraint [47]: (a) Errors in occlusal contacts leads to improper bites at posteriors. (b) Correct occlusal contacts are tight.

or Class III relationship. A Class I relationship is the one described above. A Class II (resp. Class III) relationship occurs when a premolar is extracted from the upper (resp. lower) arch thereby "shrinking" the room on the archform. Therefore, in a Class II relationship, the upper first molar is located ahead of the lower first molar and, thus, the mesiobuccal cusp of the upper first molar aligns with the interproximal contact points of the lower second premolar and the first molar (see Figure 4.6(b)). Similarly, in a Class III relationship, the buccal cusp of the upper second premolar aligns with the buccal groove on the lower first molar (see Figure 4.6(c)).

As mentioned earlier, this thesis only handles occlusion for the Class I relationship. The consideration of Class II and III relationships is beyond the scope of the thesis.

Interproximal space constraint (ISC): This determines the amount of interproximal space between two adjacent teeth on the same arch. Such gaps between teeth are unaesthetic and can lead to food impaction. Figure 4.2(b) shows an example of correct interproximal spacing between teeth.

Overjet constraint (OC): This constraint determines the transverse relationship between teeth from opposing arches when viewed from the mesial or distal side. On posterior teeth, it ensures that the functional cusps occlude with the central grooves of the opposing arches and are captured adequately by the occlusal contact constraint.



(a) (b)

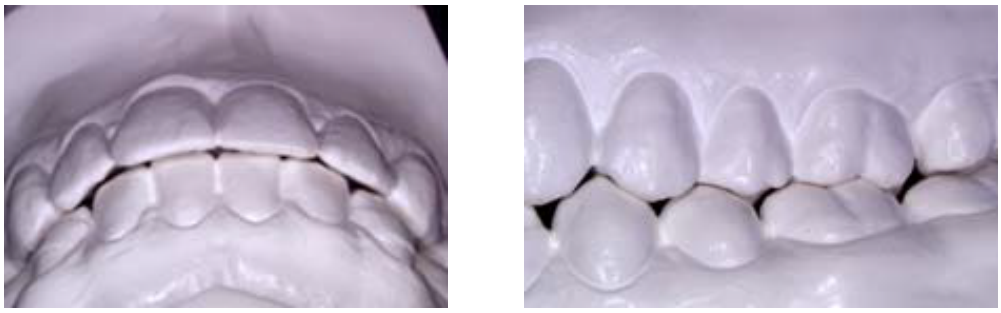
(c)

Figure 4.6: Occlusal relationship constraint and malocclusion classes [47]: (a) Class I. (b) Class II. (c) Class III.

However, on the anteriors, the lower incisal edges must be in contact with the lingual side of the upper anteriors. But this must not be achieved simply by the over-inclination of anteriors of a single arch. There are reasonable correct range of angles for the labiolingual inclination of the anteriors (see Figure 4.7). The most common errors are found among the incisors.

4.3.3 Modeling of constraints: The constraint graph

A conceptual diagram of various constraints acting on different teeth is shown in Figure 4.8. We refer to this as the *constraint graph*. The nodes represent individual teeth and the edges represent a constraint that affects two different teeth. To reduce clutter, Figure 4.8 shows the nodes on one side (say, left) only. However, this does not



(a) (b)

Figure 4.7: Overjet constraint [47]: (a) Anterior teeth. (b) Posterior teeth.

mean that the left and right sides of the arches can be aligned independently, as some constraints such as interproximal space may influence both sides simultaneously (at the central incisors). Some of the constraints relate a tooth to a more global landmark of the arches, e.g., the archform and the occlusal plane. Thus, the archform and the occlusal plane are also represented as nodes that control the alignment and the overjet of teeth, respectively. The alignment, buccolingual inclination, marginal ridges, and interproximal space constraints are the *intra-arch constraints*, i.e., they influence the position of a tooth relative to the other teeth on the same arch. Similarly, the occlusal contact, occlusal relationship, and overjet constraints are the *inter-arch constraints* and determine the quality of occlusion of opposing arches. Note that all the intra-arch and occlusal contact constraints apply to both the arches. Some of these constraints are not shown in Figure 4.8, again to reduce clutter.

Each edge in Figure 4.8 defines a type of constraint between the two teeth on which it is incident, with respect to some feature. The features used at the endpoints of the edge may be of different types (e.g., a cusp of a tooth must make contact with a groove of another tooth) or may be defined using features of multiple teeth. For example, the occlusal relationship constraint defines the relationship between the cusp of the upper canine and the interproximal contact point of the lower first premolar and the adjacent canine. (This interproximal contact point can be computed from two adjacent tooth objects, but is not a direct feature of either of the teeth.) Buccolingual inclination is the only exception in that it affects only a single tooth and, hence, both endpoints of

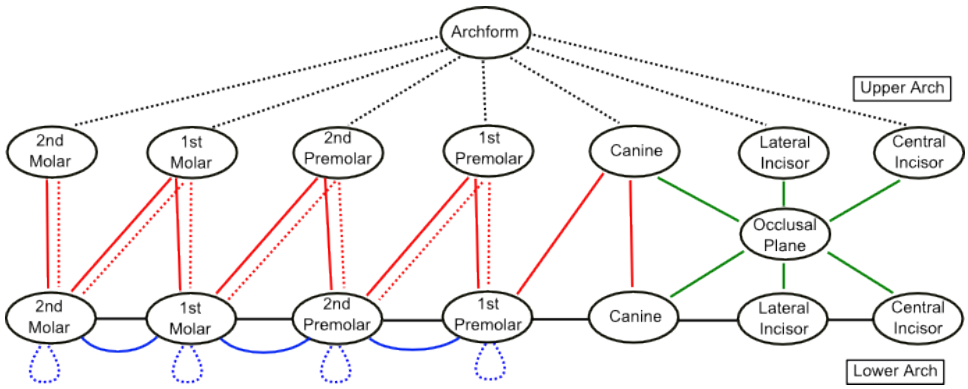


Figure 4.8: A part of the constraint graph with constraints affecting the occlusion of teeth on one side (say, left) of the upper and lower arch. The different teeth are represented by nodes. Alignment constraints (shown dotted black), interproximal space constraints (shown black), marginal ridge constraints (shown blue), and buccolingual inclination constraints (shown dotted blue) constitute the intra-arch constraints. Similarly, occlusal relationship constraints (shown red), occlusal contact constraints (shown dotted red), and overjet constraints (shown green) constitute the inter-arch constraints. Intra-arch constraints are only shown for a single arch to reduce clutter. All the intra-arch and occlusal contact constraints apply to both the arches.

the edge are incident on the same tooth (the edge is a loop). Two edges incident on the same tooth may act on different subsets of its features, e.g., the occlusal relationship constraint influences only the buccal cusps of a molar, whereas the buccolingual inclination influences both buccal and lingual cusps.

There are a few exceptions among the constraints such as the buccolingual inclination mentioned above. These arise due to the observed abnormal anatomical features in orthodontic cases. For example, the lower premolars often have *diminutive* (not well-formed) cusps on the lingual side, thus, resembling the adjacent canine more than the second premolar. Also, due to significant wear, restoration, or unusual anatomy, some of the features like cusps, marginal ridges and cusp ridges may not be well-represented. Thus, the particular constraint (edge) corresponding to these features is not considered while determining the occlusion. This is justified because it is difficult (and may be incorrect) to use the diminutive cusp to influence occlusal contacts or the buccolingual

inclination of the tooth the cusp belongs to.

Also, it has been observed in practice that for a reasonable alignment and occlusion among well-formed teeth, the combination of occlusal relationship, occlusal contact, and alignment constraints subsumes the overjet constraint for the posteriors. (This is why overjet is not shown for posteriors in Figure 4.8.)

4.3.4 Conflicts among constraints

It is easy to see that the satisfaction of some of the constraints described above will lead to conflict among certain others. The conflict among the constraints can be classified, according to the type of tooth movement involved, as horizontal and vertical conflicts.

- **Horizontal conflicts:** Consider the conflict between the alignment and inter-proximal space constraints. The crowding of teeth leads to improper layout of teeth on the archform. In order to correct the alignment, one needs to rotate and align a tooth along the archform. However, this is difficult as there is no room between the teeth on the archform. Thus, some teeth must be moved out on the archform (distally) so that enough room can be created to accommodate the tooth in question in its new position. In addition to these intra-arch constraints, the occlusal relation also is in conflict for the horizontal positioning of the teeth along the archform.

- **Vertical conflicts:** These are constraints that influence the vertical position of the tooth. Due to marginal ridges and buccolingual inclination, the intra-arch constraints try to establish a uniform height for the posterior teeth within the same arch. However, due to occlusal contact and overjet, the inter-arch constraints try to position the teeth so that the contact area is maximized.

In addition to these conflicts, there are influences due to the combination of both horizontal and vertical constraints, e.g., the correct occlusal contact (vertical positioning) depends on the establishment of correct occlusal relationships (horizontal positioning).

4.4 Establishing correct occlusion via constraint-based alignment

In this section, we describe an approach to achieve correct occlusion via the handling of the constraints described in Section 4.3.2, while keeping in mind the conflicts among constraints discussed in Section 4.3.4. We also discuss a general simulation-based alignment model based on a spring-mass system.

4.4.1 Handling of constraints

Section 4.2 discussed the limitations of techniques that view the entire arch as a single object. Next, we look at the possible approaches to constraint-based establishment of dental occlusion. Given the constraints on occlusion, some natural questions are: Can the constraints be handled independent of each other? Also, is it possible to handle the constraints by partitioning them into disjoint groups that can be handled independent of each other? A little contemplation reveals that because of the natural conflicts between the constraints discussed in Section 4.3.4, such a strategy will not work. Also, the approach of first satisfying intra-arch constraints in isolation from the inter-arch constraints followed by handling of inter-arch constraints does not perform well (e.g., due to the vertical conflicts described in Section 4.3.4).

Currently, even the relative importance of individual constraints or of groups of constraints on the global alignment and occlusion problem is not well-understood. There is evidence that, in case of a conflict, some constraints such as AC and OCC are given more importance than others such as MRC. However, these are based on the observations of experts and tend to vary in practice. Finally, unusual dental anatomy may also contribute to further complications. For example, if some molar cusp is only slightly diminutive, it may still be considered for the buccolingual inclination constraint, which may cause non-tight occlusal contact.

4.4.2 Simulation-based approach: A general spring-mass model

The discussion above clearly suggests the need for a more holistic approach to handle the constraints that considers all of them simultaneously. However, such a strategy must

also be flexible enough to incorporate some of the observed wisdom about the relative importance of constraints.

A feasible way to realize the constraints and their influences on teeth is as a system of forces on a set of 3-dimensional (3D) rigid bodies. Consider each tooth as a 3D rigid body with its surface represented by a triangle mesh. The constraints as explained above act on points in 3D space that are defined either by a known tooth surface feature or which can be computed using these available features. Now, each constraint edge can be viewed as a "spring" connecting the two rigid body masses and the influence of the constraint can be viewed as a force exerted by the spring.

Note that the system described above is entirely hypothetical and is just one possible realization of the constraint graph shown in Figure 4.8. In the modeling of teeth as rigid bodies, properties other than their surface are chosen in a normalized fashion without consideration of their actual values (e.g., each tooth is taken to have unit mass). The situation with edges is also hypothetical as they do not have any "real" physical analogue and are only defined conceptually. Therefore, several properties assigned to a constraint edge depend on the solution approach that we employ. Some examples are: force in a spring which requires defining the non-stretched length of the spring and a suitable spring constant (which may reflect the relative importance of a constraint, as a stiff spring exerts more force). More details of a concrete realization of the constraint graph with respect to our actual implementation will be given later in Section 4.5.

Given such a model of the constraint graph (teeth as nodes and edges are stretched springs attached to features), one can visualize a simulation which re-positions the teeth (nodes) such that the overall system achieves a certain optimal state. This optimal state can be defined in many meaningful ways, e.g., the minimization of total energy in the springs, or when no further tooth re-positioning is possible, etc. Various techniques exist to describe simulations on rigid-body systems, e.g., N-body simulations [3] and simulated annealing, that we can adapt to our approach. Note that a good simulation algorithm based on the approach described above must be robust across a range of actual values chosen for the properties (mass, spring constant, etc.).

The input to a simulation are the current teeth (with their features, the archform, and the occlusal plane) and a set of constraints to be satisfied, using which a set of nodes and edges for the constraint graph are created. Given this, the simulation algorithm

proceeds to move the nodes in small steps in response to the forces exerted by the springs, while also updating the node and edge properties. One can also define various boundary conditions for the simulation process, e.g., teeth cannot penetrate each other at any time, maximum distance that a tooth can move, directions in which a given tooth can move, groups of teeth that must move together, etc. An important aspect of this simulation is that it must somehow detect and overcome a local minimum configuration during its execution, such as when teeth become interlocked in a sub-optimal position and are unable to move.

4.4.3 Advantages of the simulation-based approach

The simulation approach is motivated by the process that an expert follows while planning the correct occlusion on real-world cases. We believe that this is close to how an orthodontist plans the alignment, weighing all constraints "in parallel" from a global (arch-level) as well as from a local (tooth-level) viewpoint, thus satisfying both classes of constraints simultaneously.

As discussed in Section 4.1.1, it is difficult even for a human expert to always find the optimal occlusion. Also, it has been observed in this domain that an optimal arrangement cannot exist in a neighborhood of highly sub-optimal arrangements in the "search space". This means that there will be many near-optimal arrangements of teeth that closely resemble the optimal one. In Section 4.5.3, we describe how we can get past some highly sub-optimal solutions by detecting local minimum conditions in our simulation.

Recall that the goal of virtual alignment, as it stands today, is to assist in planning patient-specific treatment in the context of the best possible outcome achievable. The actual treatment may deviate from this plan for a variety of reasons, so the planned alignment may not be necessarily achievable. Given this, an arrangement with a nearly-optimal occlusion is as good as the optimal one. (This is also reflected in the coarse scoring system given in the ABO model grading system [47].)

Also, there are some practical benefits to using a simulation-based approach. It provides fast evaluations of different treatment scenarios, such as the best outcome after a given tooth is extracted, optimization for Class I, II or III occlusion, satisfying a subset of constraints only, etc. This is extremely useful as an educational tool that can quickly

help visualize the final outcome of different strategies in occlusion planning.

Finally, real-world cases sometimes need very specialized alignment planning due to various reasons. However, in such cases, some of the existing dental occlusion errors (evaluated in terms of constraints) may be similar to the errors found in a majority of orthodontic cases. In such a situation, a simulation-based approach can help a user by generating arrangements that are in the vicinity of the optimal occlusion and can be subsequently fine-tuned by an expert.

4.5 Implementation of the simulation: Realization of the constraint graph

Sections 4.4.2 and 4.4.3 described a simulation-based approach to achieve occlusion and provided some rationale about the viability of this approach in comparison to some other techniques. In this section, a concrete implementation of such a simulation is described. This includes the details of representing teeth and their constraints in the constraint graph model. Also, the details of the simulation algorithm along with its various properties such as choice of time steps, boundary conditions, avoidance of local minima, etc. are discussed.

4.5.1 Node properties

As mentioned earlier, each tooth is represented as a node. Thus, each node is modeled as a rigid body and corresponds to a tooth surface with specially chosen points that correspond to the surface features of interest. Each node is simply given one unit of mass. As the emphasis of the simulation is on the relative importance of constraints, there is no clear advantage to having one node weigh more than the other.

There are two special nodes representing the archform and the occlusal plane of the arches. These do not correspond to any tooth surface and are global entities using which derived features, such as interproximal contact points, can be represented. Also, as will be seen later, during the simulation the archform and occlusal plane nodes do not change their position unlike the other nodes (this is an example of a boundary condition).

Next, we describe the state of each node that is maintained during the simulation. The rigid body setup and simulation techniques used here are based on the ideas and

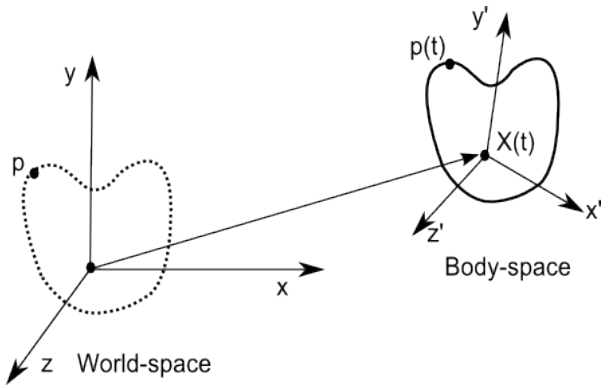


Figure 4.9: The body-space axes (x', y', z') of a rigid body shown in the world-space (x, y, z). At time t, the rigid body is translated in world-space through $X(t)$ and the direction of its body-space axes is $R(t)$. At time t, point p on the rigid body is at $p(t) = R(t)p(0) + X(t)$, where $p(0)$ is the position of p with respect to $X(0)$.

implementation discussed in [10]. It is assumed that the nodes can undergo only translation and rotation transformations and cannot be deformed (change shape). This is important because a feature such as a cusp may not remain a cusp if the tooth surface is deformed. A node N_i has a centroid (center of mass) whose position at time t is denoted as $X_i(t)$ in the *world-space* coordinate system (see Figure 4.9). Each node also has a corresponding local *body-space* which has its origin (0,0,0) at the centroid of that node. Additionally, each rigid body N_i also has a rotation matrix $R_i(t)$ associated with it that describes its orientation in world-space at time t. The matrix $R_i(t)$ can be described using three column vectors as $R_i(t) = (R_i^x(t), R_i^y(t), R_i^z(t))$ where, $R_i^x(t) = (r_{xx}, r_{xy}, r_{xz})^T$, $R_i^y(t) = (r_{yx}, r_{yy}, r_{yz})^T$ and $R_i^z(t) = (r_{zx}, r_{zy}, r_{zz})^T$ correspond, respectively, to the directions of the x-, y- and z-axis of the body-space of N_i with respect to the world-space axes at time t.

Initially, at time $t = 0$, N_i's centroid c_i is positioned at $X_i(0)$ and has the x-, y- and z-axis parallel to the corresponding world axis, i.e., $R_i(t) = I_{3\times3}$, the identity matrix. For simplicity, a point p (in world-space) on the surface of N_i is described in the body-space of N_i. Thus, $p(t)$, the position of p at time t, is calculated as

$$p(t) = R_i(t)p(0) + X_i(t) \tag{4.1}$$

where $p(0) = p - X_i(0)$ is the initial position of p with respect to the body-space origin

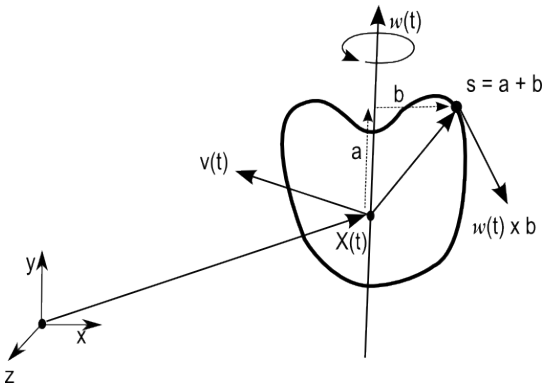


Figure 4.10: A rigid body is shown with its linear velocity ($v(t)$) and angular velocity ($\omega(t)$). The rate of translation of a vector s on the rigid body through world-space is $v(t)$ and its rate of change of orientation due to $\omega(t)$ is given by $\omega(t) \times b = \omega(t) \times (a + b) = \omega(t) \times s$.

$X_i(0)$ of N_i (see Figure 4.9).

The state description of each node has a position ($X_i(t)$) and an orientation ($R_i(t)$). The simulation would also move some nodes (teeth) under the influence of constraint-driven forces. This requires the tracking of the linear velocity and angular velocity of each node during the simulation. The linear velocity $v_i(t)$, is the rate of change of position of the node's centroid at time t ($\dot{X}_i(t)$). Similarly, $\omega_i(t)$ is the angular velocity of the rigid body in world-space (see Figure 4.10). The direction of $\omega_i(t)$ gives the axis of rotation and the magnitude $|\omega_i(t)|$ gives the speed of rotation.

The rate of rotation of the axes of the body-space, denoted as the matrix $\dot{R}_i(t)$, is computed from the angular velocity as described below. Consider a vector s in the body-space of N_i. In the body-space, s can be written as $a + b$ where vectors a (resp. b) are its components parallel (resp. perpendicular) to $\omega_i(t)$. The instantaneous movement of the point at the tip of s due to $\omega_i(t)$ is along a circle in the direction orthogonal to both b and $\omega_i(t)$ and its tangential velocity is given by $|\omega_i(t)||b| = \omega_i(t) \times b = \omega_i(t) \times (a + b)$ (as a and $\omega_i(t)$ are parallel) $= \omega_i(t) \times s$. Thus, the rate of change of a vector $s(t)$ at time t due to $\omega_i(t)$ is $\dot{s}(t) = \omega_i(t) \times s(t)$ (see Figure 4.10).

The rate of change of the column vectors of a rotation matrix $R_i(t)$ can derived

similarly [10].

$$\dot{R}_i(t) = (\omega_i(t) \times R_i^x(t), \quad \omega_i(t) \times R_i^y(t), \quad \omega_i(t) \times R_i^z(t)) \tag{4.2}$$

Given a vector a, define a matrix a^* as

$$a^* = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix}$$

Then, the cross-product $a \times b$ can be written as a^*b, i.e.,

$$a \times b = a^*b = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} = \begin{pmatrix} a_y b_z - b_y a_z \\ -a_x b_z + b_x a_z \\ a_x b_y - b_x a_y \end{pmatrix}.$$

Using this, Equation 4.2 is written as

$$\dot{R}_i(t) = (\omega_i(t)^* R_i^x(t), \quad \omega_i(t)^* R_i^y(t), \quad \omega_i(t)^* R_i^z(t)) = \omega_i(t)^* R_i(t) \tag{4.3}$$

The quantities $v_i(t)$ and $\omega_i(t)$ are maintained in the state of N_i and are computed using the forces and torques acting on N_i. Given a node with linear velocity $v(t)$ and current position $X(t)$, the new position of node at time $t + \Delta t$ is computed using the Euler-forward technique [37] as:

$$X(t + \Delta t) = X(t) + \Delta t \cdot \dot{X}(t) = X(t) + \Delta t \cdot v(t) \tag{4.4}$$

Similarly, the rotation matrix at time $t + \Delta t$ is computed using the current rotation matrix $R(t)$ and the rate of its rotation $\dot{R}(t)$ as:

$$R(t + \Delta t) = R(t) + \Delta t \cdot \dot{R}(t) = R(t) + \Delta t \cdot (\omega(t)^* R(t)) \tag{4.5}$$

Each node may have many external forces acting on it at any given time as a result of the constraints, e.g., attractive forces (occlusal contact) and repulsive forces (minimum interproximal space). A vector $F = (f_x, f_y, f_z)^T$ describes the direction and magnitude of the force. Let there be k forces $F_{i1}, F_{i2}, \ldots, F_{ik}$ acting on rigid body N_i at time t at points $p_{i1}(t), p_{i2}(t), \ldots, p_{ik}(t)$, respectively. Then, the net external force on a node N_i at time t due to all the forces is

$$F_i(t) = \sum_{j=1}^{k} F_{ij}. \tag{4.6}$$

Similarly, the net torque on N_i due to all the forces is

$$\tau_i(t) = \sum_{j=1}^{k} (p_{ij}(t) - X_i(t)) \times F_{ij}. \tag{4.7}$$

The derivation of the linear (resp. angular) velocity via the computation of the linear (resp. angular) momentum using forces (resp. torques) for our simulation is based on [10]. The *state* $S_i(t)$ of a node N_i at a time t is the position, orientation, linear velocity and angular velocity of the node, i.e., $S_i(t) = (X_i(t), R_i(t), v_i(t), \omega_i(t))$. Given a set of nodes representing rigid bodies, one can maintain the history of the nodes' movements during an interval of time under the influence of forces and resulting torques by using a sequence of states.

Collision detection among rigid bodies

It is clear that one of the important constraints that must be respected at any instant of the simulation is that no two teeth (nodes) surfaces must penetrate each other. (Unlike the orthodontic constraints discussed in Section 4.3.2, collision-avoidance is a constraint imposed by the simulation.) This means that the collision between teeth must be evaluated at all time instants of the simulation. Existing solutions to collision detection between two tooth-like non-convex surfaces are computationally expensive [26, 41].

We use a simplified and efficient approach to collision detection among teeth by observing that all collisions can be classified into two groups: (a) inter-arch collisions: those between teeth on different arches (occlusal contact) and (b) intra-arch collisions: those among adjacent teeth on the same arch.

The inter-arch collisions can be checked by simply checking for penetrations on the occlusal region of each tooth involved. Consider tooth objects A and B. For each of these tooth objects, a hierarchical, tree-based bounding box data structure for their 3D mesh vertices is created; these structures are denoted as H_A and H_B, respectively. Thus, the highest level of H_A and H_B corresponds to the bounding-box of all the mesh vertices of A and B, respectively. If the surfaces of A and B penetrate each other, then, the bounding-boxes at the highest level of H_A and H_B intersect. In other words, if two bounding-boxes do not intersect, then the surfaces corresponding to the vertices in their boxes do not penetrate and the search can be terminated. Otherwise, we search

recursively on the corresponding sub-bounding-boxes of these bounding-boxes in H_A or H_B, until a collision is detected, or the set of sub-bounding-boxes is exhausted in one of the trees. This is based on a popular approach to collision detection among 3D surfaces [26], but is expensive because the structures H_A and H_B must be updated (or rebuilt) when teeth vertices are moved during the simulation.

Intra-arch collision checking can be done as follows. A vertex of a tooth mesh can be projected horizontally onto its nearest point on the archform. The projections of all the vertices of a tooth define an interval on the archform, where the left (resp. right) endpoint of the interval is the projection with least (resp. greatest) x-coordinate value. We consider two adjacent teeth as colliding if their corresponding intervals on the archform overlap. This is sufficient because eventually all teeth are expected to be aligned along the arch without any rotations and an overlap of the intervals of two teeth leads to a violation of interproximal space constraint. The projection intervals are computed to find the interproximal contact points between adjacent teeth. Due to this, the intra-arch collision checks are very efficient.

4.5.2 Edge properties

Each edge of the constraint graph defines a constraint between features on nodes. An edge exerts a force on the nodes it connects through the endpoints where it is connected. The various characteristics of different edge types are discussed below.

Recall that the endpoints of an edge can represent an intrinsic feature (e.g., cusp, ridge, groove, etc.) or a derived feature (e.g., the interproximal contact point between two adjacent teeth). The interproximal contact point between two adjacent teeth was defined earlier as the midpoint of the overlap between the intervals representing the projection of each tooth on the archform. This is motivated by the following observation: The occlusal relationship constraint is defined as the alignment of cusps with interproximal contact points when viewed from the buccal side. Thus, it is seen that the occlusal relationship constraint is in fact defined for the alignment of the cusp and the interproximal contact point when both are viewed (using projections) with respect to the archform.

Each edge generates a corresponding force that drives the simulation. According to Hooke's law [37], the force due to an edge E (considered as a stretched or compressed

spring) depends on the current length of the edge (E_L), the relaxed length (E_R), the spring constant (E_K), and some boundary conditions. In our simulation, the relaxed length (minimum energy state) of an edge is set to be very small, typically 0.05 mm. This is so because every constraint when satisfied (in isolation) would lead to a total shrinkage of the corresponding edge. Thus, any spring's length more (resp. less) than its relaxed length generates an attractive (resp. repulsive) force between the connected nodes.

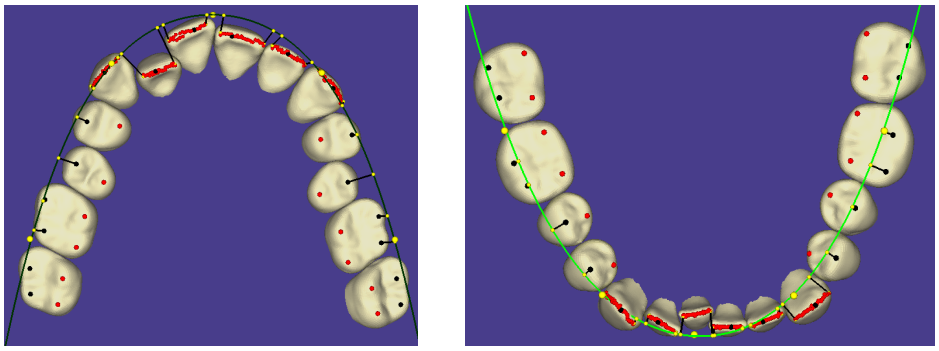
The spring constants E_K are useful in defining the relative importance of each constraint and are chosen to be same for each constraint-type. This is motivated by the fact that a stiff spring (high E_K) holds more energy than a normal spring when both are stretched (or compressed) by the same length. Thus, edges corresponding to more important constraints such as alignment and occlusal contacts may be given a higher E_K than other edges. The actual value of E_K used in our experiments was an integer in the set $\{0, 1, 5, 10, 15, 20\}$. (An E_K value of 0 on an edge will eliminate the constraint type associated with that edge).

We now provide some details on the forces due to each constraint type.

- **Alignment constraint edge:** Figure 4.11 shows examples of alignment constraint edges. The forces move the teeth so that the associated features (incisal edges of anteriors (incisors and canines) and buccal cusps of posteriors) on the teeth align with their closest points on the archform. Note that the anteriors have two endpoints on the incisal edges connected to the archform. Thus, if the tooth experiences any rotation (incisal edge not aligned with archform), the two forces couple to create a torquing effect to rotate the tooth (see the lateral incisor in Figure 4.11(a)).

- **Interproximal space constraint edge:** Figure 4.11 also shows examples of interproximal space constraint edges. These constraints exists between every pair of adjacent teeth and are responsible for creating the needed space on the archform for rectifying any crowding among anteriors. These forces can be either mesial (attractive, closing excess interproximal space) or distal (repulsive, creating more space).

- **Marginal ridge constraint edge:** Figure 4.12 shows an example of marginal



(a) (b)

Figure 4.11: The alignment constraint edges (shown as black line segments) between teeth and archform (shown green). Shown are (a) an upper arch and (b) a lower arch. The endpoints of the constraints are shown as yellow markers. Note the crowding of teeth among the incisors.

ridge constraint edges for some posterior teeth. The forces here are only in the vertical direction and their magnitude is proportional to the difference in the heights of adjacent marginal ridges.

- **Buccolingual inclination constraint edge:** These edges represent purely rotational torques whose purpose is to bring the lingual and buccal cusps to the same height by rotating the tooth. The torque is generated by applying a vertically-oriented force at either the lingual or buccal cusps such that magnitude of the force is proportional to the difference in their heights.

- **Occlusal relationship constraint edge:** An example is shown in Figure 4.13. Note the position of the upper first molar and canine. The vertical angulation of these edges determines the magnitude and direction of the resulting force; more angulation results in greater force. The same applies for the actual distance between the cusp and the interproximal contact point or buccal groove.

- **Occlusal contact constraint edge:** Figure 4.14 shows some of these edges, which are attractive forces bringing the cusps and grooves closer to get tight occlusal contact.

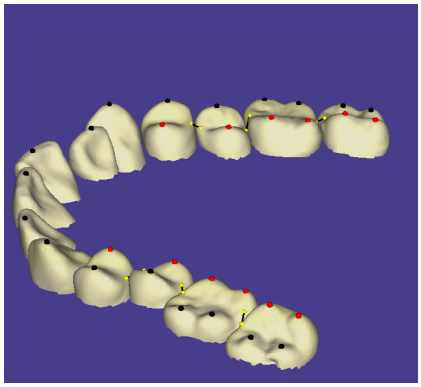


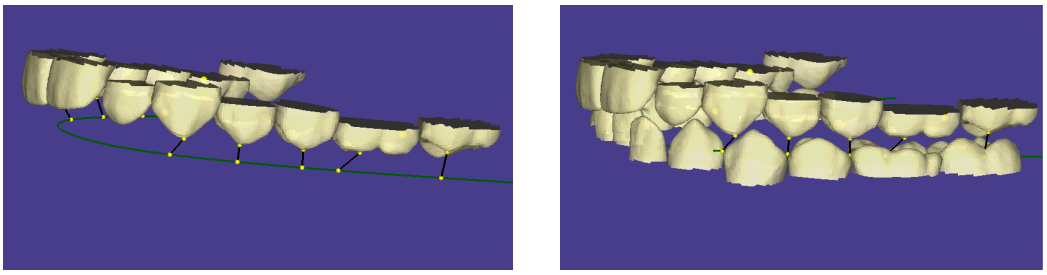
Figure 4.12: The marginal ridge constraint edges (shown as black line segments) between the midpoints of the marginal ridges (shown as yellow markers) of adjacent posterior teeth. The force due to these edges depends only on the difference in the vertical height of their two endpoints.

- **Overjet constraint edge:** In the anteriors, these edges contribute to attractive forces between the incisal edges of lower incisors and the lingual surface of upper incisors. Also, it forces the anteriors into correct inclination by enforcing the known correct inclination angles for the long axis of anterior crowns. This is to prevent over-inclination of the upper or lower anteriors to satisfy the contact part of the constraint.

4.5.3 The simulation algorithm

The constraint graph is denoted as $G(V, E)$, where V and E are the set of nodes and the set of constraint edges, respectively. Recall that the state of a node N_i is $S_i = (X_i(t), R_i(t), v_i(t), \omega_i(t))$. A *configuration* C_t is defined as a snapshot, $S = S_1, S_2, \ldots, S_{|V|}$, of the states of all the nodes at a time instant, t, of the simulation. A chronological sequence of configurations succinctly records the entire trajectory of all tooth movements and is used for the animation of the occlusion plan.

In order to drive the algorithm to produce better occlusion among nodes (teeth), a measure of quality for a configuration is essential. Such a quality measure depends on how well the constraints are satisfied in the given configuration and is a measure of the correctness of the occlusion achieved. The quality of each constraint type is



(a) (b)

Figure 4.13: Occlusal relationship constraint edges on the teeth of an upper arch. (a) Shown with archform only. (b) Shown in relation to the teeth on the lower arch.

given a *score* based on the ABO grading system [47]. This score is derived from the measurements made on the features of the teeth and is complete with a set of special cases (e.g., diminutive cusps are not considered for occlusal contact constraint). The net score of a configuration, C, denoted $\textsc{score}(C)$, is calculated as the sum of the scores of the individual constraints due to all constraint edges in E. The goal is to compute a configuration that minimizes $\textsc{score}(\cdot)$.

There are many boundary conditions imposed on the simulation. Some of these have been discussed earlier, e.g., nodes corresponding to the archform and the occlusal plane are restricted not to move. Also, there may be restrictions on the maximum distance that a node can move in a single time step to avoid abrupt, unnatural movement. There may be different restrictions on the total distance different nodes can move from their original positions, e.g., it may be useful to not move the first molars and canines too much compared to the incisors. Different edge types have different relaxed lengths E_R. These boundary conditions may also evolve over time as the simulation proceeds, e.g., certain teeth are restricted to move together (this is not handled currently by our algorithm and would require merging of nodes in G). Thus, one can appreciate that there are many boundary conditions, a subset of which a user might be interested in applying to a simulation run to study the differences in outcomes produced by initial setups. The simulation algorithm must respect these boundary conditions during its entire execution.

Section 4.5.2 described how forces are computed from each of the constraint edges. An outline of the main simulation procedure is shown in Algorithm 1. The output of this

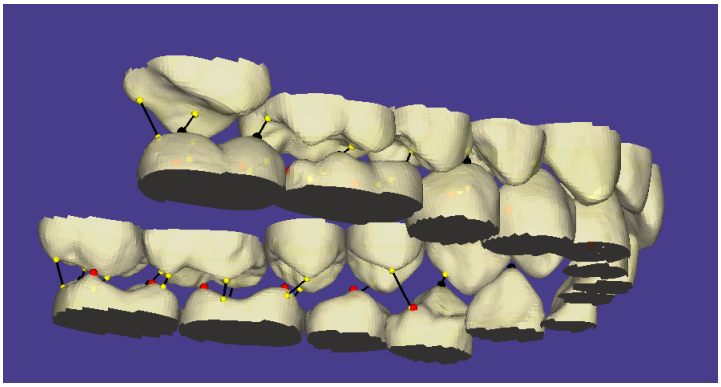


Figure 4.14: Occlusal contact constraint edges shown for some of the posterior teeth. Note the edge connecting the cusp of the molar and the corresponding groove on the opposing arch in the posterior teeth.

algorithm is a sequence of configurations leading to an optimal outcome, i.e., the final configuration, C_{final}, minimizes SCORE$(\cdot)$, subject to boundary conditions. Thus, based on our hypothesis (stated at the end of Section 4.3.1), this leads to the best possible occlusion of the teeth with high probability. (More discussion on optimality is provided later in Section 4.5.4.)

Algorithm 2 computes the next configuration of states such that the energy stored in the constraint edges (which is proportional to their length) is minimized or until no further tooth movement is possible. It is based on the assumption that this minimization leads to better occlusion (see Section 4.4.2). Also, given that a user can express the relative importance of constraints, the edges with larger spring constants will influence the computation of the next state more than those with smaller spring constants. This naturally resolves conflicts among the forces via a energy minimization approach. The parameter δ is used to check if there was any significant change in any tooth's position or orientation to continue the simulation further. This is done by checking if the 1-norm of the position vector $X(t)$ and of the rotation matrix $R(t)$ has changed by more than δ between two consecutive iterations.

Algorithm 1 Constraint-based-simulation

Input: C_{t_0}, t_0; Global input variables: $G(V, E)$, Δt, MaxIterations, Boundary conditions, δ.

Output: Sequence of configurations, $\mathcal{C} = C_{t_0}, C_{t_0+\Delta t}, C_{t_0+2\Delta t}, \ldots, C_{final}$, computed during the simulation.

▷ C_{t_0} is the initial configuration and C_{final} is the configuration with optimal occlusion. Δt is the duration of time step by which simulation is advanced in each iteration and δ is an error threshold (used in Algorithm 2).

1: $t = t_0$, $C_t = C_{t_0}$. ▷ Initialization.
2: $\mathcal{C} = C_{t_0}$.
3: NumIterations = 0
4: **while** (NumIterations ≤ MaxIterations) **do**
5: $(\hat{\mathcal{C}}, \hat{t}) =$ **Simulation-step**(C_t, t). ▷ Run simulation until it stops.
6: $\hat{C} = \hat{\mathcal{C}}.end$. ▷ Best configuration in $\hat{\mathcal{C}}$.
7: $(\mathcal{C}', t') =$ **Check-local-minimum**$(\hat{C}, \hat{t})$. ▷ Overcome the local minimum at $\hat{C}$.
8: $C' = \mathcal{C}'.end$. ▷ Best configuration after local minimum check.
9: **if** SCORE$(C') \leq$ SCORE$(\hat{C})$ **then** ▷ Lower score indicates better occlusion.
10: Append configurations in $\mathcal{C}'$ to $\mathcal{C}$. ▷ Record the simulation history.
11: $t = t'$.
12: $C_t = C'$. ▷ Proceed from C'.
13: **else**
14: Append configurations in $\hat{\mathcal{C}}$ to $\mathcal{C}$. ▷ Record the simulation history.
15: Return $\mathcal{C}$. ▷ Output the best occlusion computed.
16: **end if**
17: NumIterations = NumIterations + 1.
18: **end while**
19: Return $\mathcal{C}$. ▷ NumIterations > MaxIterations

Algorithm 2 Simulation-step

Input: C_{start}, t_{start}. ▷ Global input variables; same as in Algorithm 1.

Output: Sequence of configurations computed during simulation $\mathcal{C} = C_{start}, C_{start+\Delta t}, \ldots, C_{end}$ and t_{end}.

1: $t = t_{start}$, $C_t = C_{start}$. ▷ Initialization (see Sections 4.5.1 and 4.5.2).
2: $\mathcal{C} = C_{start}$.
3: NumIterations= 0
4: **while** (NumIterations $\leq$ MaxIterations) **do**
5: $t = t + \Delta t$.
6: $C_t = C_{t-\Delta t}$. ▷ Create a copy of previous state.
7: Get the active constraints in edge set E.
8: Compute forces and torques on nodes. ▷ Section 4.5.2.
9: Update linear and angular velocity.
10: Compute new states of nodes. ▷ Section 4.5.1.
11: Apply boundary conditions to new states.
12: Record the new states in C_t.
13: **if** Corresponding states in C_t and $C_{t-\Delta t}$ differ by less than δ **then**
14: Return $\mathcal{C}$.
15: **else**
16: $\mathcal{C} = \mathcal{C} \cup C_t$. ▷ Add C_t to simulation history.
17: **end if**
18: NumIterations = NumIterations + 1.
19: **end while**
20: Return $\mathcal{C}$. ▷ NumIterations > MaxIterations

Algorithm 3 Check-local-minimum

Input: C_t, t. ▷ Global variables; same as in Algorithm 1.

Output: Sequence of configurations computed during simulation $\mathcal{C} = C_{start}, C_{start+\Delta t}, \ldots, C_{end}$ and t_{end}.

▷ The positions of nodes in C_t are randomly displaced within the boundary condition restrictions (see Section 4.5.3) and the simulation is run. The best scoring sequence of configurations among such runs is returned.

1: BestScore = SCORE(C_t).
2: $\mathcal{C} = C_t$.
3: $t_{end} = t$.
4: **for** $i = 1$ to k **do**
5: Create $C^{(i)}$ by displacing nodes from their positions in C_t.
6: $(\hat{\mathcal{C}}, \hat{t})$ = **Simulation-step**$(C^{(i)}, t)$. ▷ Run the simulation.
7: $C_{end} = \hat{\mathcal{C}}.end$.
8: S = SCORE(C_{end}).
9: **if** BestScore $\leq S$ **then** ▷ If a better configuration is found.
10: $\mathcal{C} = \hat{\mathcal{C}}, t_{end} = \hat{t}$, BestScore $= S$.
11: **end if**
12: **end for**
13: Return $\mathcal{C}, t_{end}$. ▷ Best among the k runs.

Local minimum configurations in simulation

Any optimization method must completely avoid, or detect and rectify, the situation when the search is stuck in a local minimum configuration. A *local minimum configuration* is reached when no further node movement (translation or rotation) is feasible (possibly due to collision constraints), but the total energy in the edges can be reduced. In other words, the teeth cannot move further, but a configuration with better occlusion exists. Algorithm 3 shows the algorithm used to detect and handle a local minimum configuration in order to continue the search for a global minimum configuration.

Given a configuration in which teeth are interlocked and cannot move further, Algorithm 3 introduces random perturbations to the tooth positions (particularly in the z-direction), and restarts the simulation. This is done a constant number of times (specified as k in Algorithm 3, line 4) and the best resulting configuration (the one with least score) is returned. If the initial configuration was truly a local minimum, then, with high probability, some set of perturbations will lead to a better configuration. However, if the configuration was indeed a near-optimal one, then no further improvement would likely be possible and the initial configuration is returned as the result. Note that the boundary conditions on tooth movements must be respected while creating the perturbations.

4.5.4 Characteristics of the simulation algorithm

The simulation algorithm differs from the traditional rigid body simulation methods in many ways. First, the collision detection among nodes is not handled uniformly for simplicity and efficiency. For example, the collision between a pair of nodes on the same arch, the opposing arches, and a tooth node and archform node are all handled in completely different ways (see Sections 4.5.1 and 4.5.4). Also, the exact values of mass, spring constants, termination conditions are all chosen experimentally to make the simulation work. This section describes some of the characteristics of the algorithm.

Duration of time step

A key factor that controls the accuracy and performance of Algorithm 2 is the magnitude of each time step, Δt, which controls the rate of advancement of the simulation

(see Equation 4.4). Large values of Δt result in large changes in the translation and rotation of the nodes. If these changes are too large, they may miss the optimal solution because the search is very "coarse". However, if Δt is too small, it may take a long time (many iterations) to converge and result in high computational cost. Most real-world simulations have a similar Δt parameter which is often best found experimentally, as we do.

Residual velocities and cycling through forces

Another major issue with the simulation, as described, is that of spurious movement of nodes due to residual velocities. As an illustration, consider a tooth node in Figure 4.11. Suppose that the only constraint acting on the tooth node is the alignment constraint forcing it to align along the archform. As the tooth node starts moving towards the archform, it builds up a certain momentum (and velocity). Now, consider the time when the tooth is aligned with the archform. The alignment constraint is satisfied and, hence, there should be no further tooth movement. However, the collision between a tooth node and the archform node cannot be handled in the same way as collision of two teeth (the archform is a curve and could be on a different plane than the tooth, i.e., the corresponding nodes "collide" only when viewed from top). Thus, even though the alignment constraint is satisfied, there is some amount of residual velocity in the tooth node that moves it past the target position, eventually leading to an oscillatory movement of the tooth about the archform. It is also clear that as one adds more constraints, such behavior will only multiply and get more complicated.

A simple solution to the above problem, without doing any sophisticated collision detection, is to simply reset the linear and angular velocities to zero after a certain number of iterations. This technique, while not affecting the long-term position and orientation of the nodes during simulation, suffices to handle the linear and angular momentum that is built up due to large forces on constraint edges.

In the case of multiple constraints, in addition to the reset of velocities, the simulation also selects a constraint-type that drives the computation of forces and torques on nodes for a small number of iterations. Thus, the entire time interval is divided into very small subintervals during each of which only a single constraint determines the next configuration. Thus, the effect of multiple constraints acting more or less simultaneously

over the time interval is approximated by the cumulative effect of individual constraints acting independently over successive subintervals of very small length. This simplifies the simulation considerably while still ensuring that the effect of no single constraint skews the results unfairly.

Optimality and convergence

Section 4.4.3 has already presented several arguments for why the configurations produced by the simulation approach are close to the optimal occlusion in most cases. Some discussion was also given why this is a reasonable goal given that the optimality of a configuration cannot be verified due to the lack of metrics to evaluate the quality of a given configuration of teeth in a pair of opposing arches.

Also, given the nature of the problem, the convergence of Algorithm 2, i.e., the termination of the simulation upon reaching an optimal configuration, is difficult to prove. To see this, consider a pair of equal and conflicting forces at a very small scale, e.g., interproximal space and occlusal relationship constraints acting on a second molar and influencing its placement along the archform. In such a case, it is possible that in the final stages, the simulation simply moves back and forth between two similarly-scored states, as the teeth move in an oscillatory fashion. This motivates the use of a threshold on the amount of tooth movement to be considered significant and an upper bound on the number of maximum iterations that the simulation can run (which is determined experimentally).

However, our experiments consistently show that the occlusion achieved by the simulation process is quite good and often converges quickly to near-optimal configurations. This is because the chosen constraints (see Section 4.5.2) guide the search for correct occlusion very efficiently.

4.6 Evaluation

This section describes the software tool which implements the simulation algorithm described in Section 4.5.3 and presents the results of the simulation on a set of real-world test cases.

As in previous chapters, the alignment module of the tool consists of an intuitive user

interface to load dental models and carry out the simulation of alignment. Figure 4.15 shows a screenshot of the alignment module tool and describes its functionality. In addition to this, for the purpose of evaluating the outcome of the simulation algorithm, we also use a proprietary software tool [24] that determines the discrepancies in the tooth positions and orientations in given pair of arches.

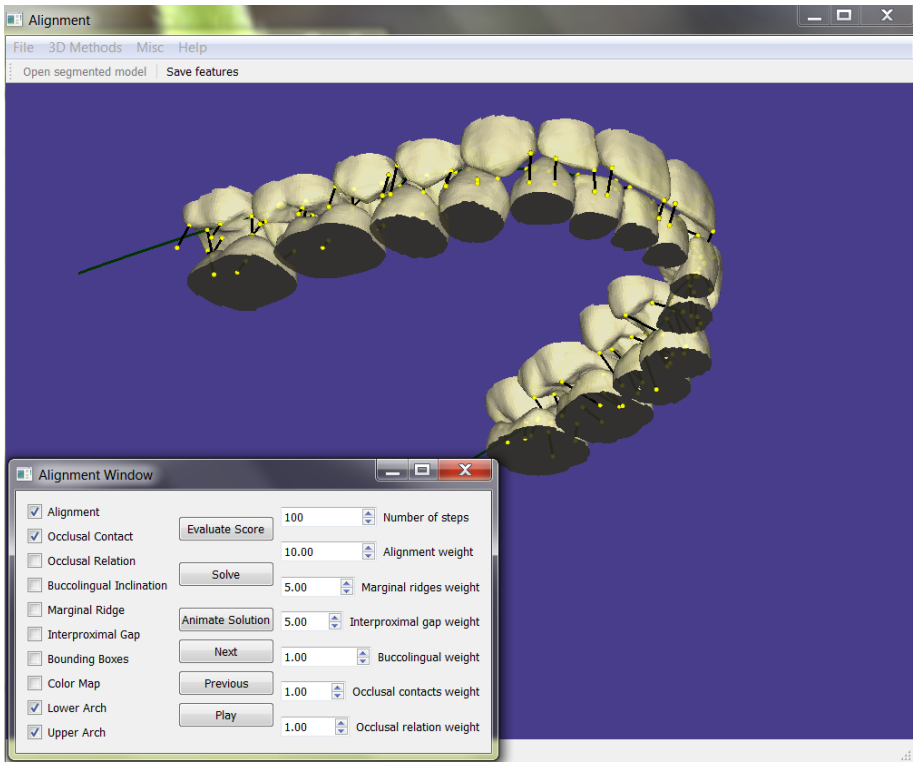


Figure 4.15: Screenshot of the feature evaluation and alignment tool. An upper and a lower arch is shown along with some of the constraint edges (forces) that have been activated for the simulation. The checkboxes on the lower-left window are used to activate or deactivate different constraint edges. The numeric fields on the right side of this window can be used to set the weight of the edges of each constraint-type and the maximum number of iterations in the simulation process. The buttons in the middle are used to start the simulation, view the animation of tooth movement computed during the simulation, and evaluate a score for the quality of the occlusion achieved.

4.6.1 Results and analysis

The simulation algorithm described in Section 4.5.3 was evaluated on a set of 8 lower and 8 upper arches. These presegmented arches were selected from a database of actual cases treated at the University of Minnesota's dental clinic. The models were obtained through the SureSmile® software [52] and cover a range of difficulty, including malocclusions, improperly formed teeth, and varying anatomy on posteriors. For each of these cases that we studied, the arrangement of teeth from both before and after the alignment planning phase (the latter determined by an expert) are given. (These are known as the pre-treatment and post-treatment configurations, respectively.)

The process of computing an alignment to improve occlusion is carried out by using the alignment module of our software tool described above. Note that some of the simulation parameters such as the timestep (Δt) and the error threshold (δ) are set to a value that performs well for all types of cases, as predetermined experimentally. Also, it was observed that a broad range of weight assignments for the constraint edges (representing their spring constants) performed well on a majority of the models. (An example of such a weight assignment is to set the weight of AC edges to 10, the weights of MRC, ISC, and OCC edges to 5, and all the remaining edge weights to 1.)

As mentioned in Section 4.1.1, there are very few well-established metrics to evaluate the quality of occlusion for a given dental model. In practice, an expert often relies on visual inspection for a qualitative analysis. However, there are some quantitative methods to grade the quality of occlusion by making certain measurements on surface features of the teeth as described in Section 4.3.2, and checking for commonly occurring orthodontic errors. Our implementation is based on the techniques mentioned in Section 4.3.2, and is used to evaluate a score for each dental model (see Section 4.6.1). Also, the availability of the post-treatment arrangement of the same teeth enables us to measure the discrepancies in the tooth positions computed by the simulation and the positions established by an expert (see Section 4.6.1).

Score-based analysis

Each active constraint is evaluated on the current position and orientation of the involved teeth. If a constraint is not satisfied, an integer score in the range $[0, 2]$ is assigned to it.

For example, if the height of the adjacent marginal ridges differ by less than 0.5 mm, no score is assigned. However, if this difference is between 0.5 mm and 1.0 mm (resp. 1.0 mm and 2.0 mm), a score of 1 (resp. 2) is assigned. Thus, this provides a coarse grading of the quality of occlusion in terms of the error in constraints. Thus, a similar score is assigned when an incisal edge or a buccal cusp is out of alignment from the neighboring teeth (AC), and so on. More details on these score values can be found in [47]. The total of all the scores of all the constraint edges defined on the arches is taken as the total score.

As mentioned previously, a low score for a model indicates better quality of occlusion. Figures 4.16-4.19 show some examples of models with frequently occurring errors. Also given are graphs that show the score due to each constraint and the total score. These graphs reveal how changes in the score of a constraint affect the other scores.

Figure 4.16 shows a normal orthodontic case with a small amount of crowding at the anteriors, whereupon the simulation process brings about a gradual and steady drop in the scores of all the constraint types. Figure 4.17 shows an example with too much space between the anterior teeth of the upper and lower arch. This is difficult to handle because of the conflict between the ISC and ORC. As a result, even though the teeth can be taken to their corresponding archform very quickly (AC is satisfied within a few rounds), the ORC and OCC scores are significantly affected.

Also, the lower second molars in Figure 4.17 have very little crown area available, especially on the lingual side. This, in turn, makes it difficult to identify the cusps and use them in determining the correct occlusal contacts and their relationships. Such problems with second molars are in general difficult to handle automatically because of the wide variety of the errors that occur. Thus, this is a situation where a user's experience will be needed to establish an occlusion that is acceptable.

An example where an upper arch anterior tooth is "trapped" in the lingual region of the lower anterior teeth is shown in Figure 4.18 (note the left lateral incisor on the upper arch). This situation may also happen in the posterior region where a tooth cannot move because of other teeth in its neighborhood. Such problems can be resolved in an efficient fashion by ensuring that there is enough room for the teeth to move horizontally, without collisions with the opposing arch teeth, at least for the initial few steps of the simulation. This usually suffices because the AC (in general, the intra-arch constraint) edges have

large spring constants associated with them, and are able to play a more significant role especially in the initial iterations of the simulation. This scope for horizontal movement is provided by moving each tooth vertically away from the other arch so that there is a horizontal plane separating the two arches. This initial abrupt movement of teeth is responsible for the sudden increase in the scores in the first two or three iterations. Finally, Figure 4.19 shows a case where there is sufficient crowding on the archform on the upper anteriors.

Surface-matching-based analysis

Next, we describe a surface-matching-based analysis of the final occlusion established by the simulation algorithm. As mentioned earlier, this is done by means of the *e*model compare 8.1 software that takes as input two arrangements of the same set of teeth, one computed by the simulation and the other determined by an expert, and determines the discrepancies in the position of each tooth in one arrangement relative to the corresponding tooth in the other arrangement. This is achieved by finding a close match, which minimizes the total separation of the vertices of the surfaces in question. The surface-based method has advantages and disadvantages when compared to the score-based method discussed above. The score-based method is somewhat coarse as it assigns real-valued measurements to integer-valued "buckets", whereas the surface-based method is more sensitive small differences in the tooth positions and to factors such as the density of meshes. However, the surface-based method may provide too much detail in comparing the quality of occlusion even when it suffices to establish correct occlusion globally rather than on a tooth-by-tooth basis. Finally, the score-based method does not rely on the availability of a treatment plan created by an expert.

The errors are reported in terms of the translations and rotations needed to align each tooth with its image in the other arrangement. However, often the two arrangements are not in the same frame of reference and, due to this, a global registration of the two arrangements must be established to offset any global discrepancies. For example, consider two adjacent posterior teeth whose marginal ridges are not at the same height. In order to satisfy this constraint, the higher-placed tooth can be moved vertically down, or the lower-placed tooth can be moved up. Even though the net effect on the occlusion in both cases would be the same, the two arrangements produced by the alternative moves

will be off vertically from each other, when considered in a global frame of reference.

For each of the cases, we compare the arrangement resulting from the simulation to the post-treatment arrangement established by an expert. First, the two arrangements are registered globally, after which individual teeth are compared. Tables 4.1 and 4.2 present the results on teeth in 8 upper and 8 lower arches, respectively. The mean values of their error are shown along with their standard deviation.

The discrepancies are computed based on a coordinate-system located at each tooth parallel to the archform. This enables us to assign meaningful descriptions to the differences in the tooth positions and the orientations about different axes found by the surface matching algorithm. Thus, the errors are presented relative to an orthogonal coordinate system which has its axes parallel to the mesiodistal line, buccolingual line, and the vertical z-axis. The corresponding errors in orientations are taken as the error in tooth inclination (torque), tooth angulation (tip), and tooth rotation (spin about the z-axis).

The low order errors in the mesial-distal and buccal-lingual columns indicate that the simulation does align the teeth to the archform as expected of the strong intra-arch forces. The correct vertical positioning of the teeth is little more challenging because of the conflicts among MRC and OCC. On the other hand, the errors are relatively higher in terms of the torque, tip and rotation angles. We note that there no significant difference in the discrepancies between the upper and the lower arch teeth. Also, no significant differences were observed between teeth from the left and right side of the arches.

Note that discrepancy values with a large variance were observed for the tooth angulation comparison (shown in columns labeled as "Tip" in Tables 4.1 and 4.2). This is due to both the lack of reliable tooth features to describe the correct crown angulation and the lack of constraints to enforce the correct crown angulation in the simulation process.

4.7 Conclusion

This chapter presented a technique to establish dental occlusion via a simulation-based approach that models the alignment process as a spring-mass system, where teeth are

considered to be masses connected by springs that represent dental constraints. These constraints are obtained from the experience of practitioners and well-known approaches to objectively classify commonly occurring orthodontic errors. A set of constraints guide the simulation algorithm to find an arrangement that best satisfies the given constraints subject to certain boundary conditions. As part of future work, Chapter 5 discusses some enhancements to the simulation algorithm and some advanced modeling techniques to make the virtual alignment process closer to the natural treatment process.

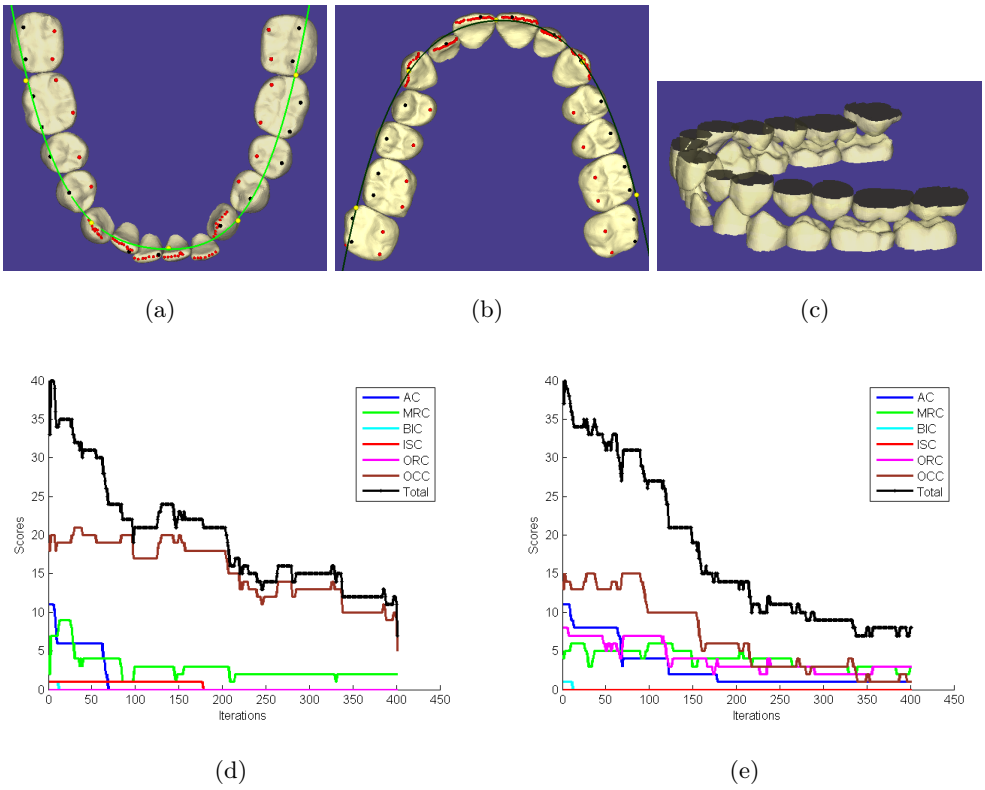


Figure 4.16: The simulation of teeth under the influence of constraint-based forces. The lower arch (a), upper arch (b) and their combined occlusion from the buccal side (c) is shown. The simulation history and the scores due to different constraints are shown for the lower arch (d) and the upper arch (e). The black line at the top represents the total score due to all the constraints. The x-axis in the graph represent the iteration number during the simulation.

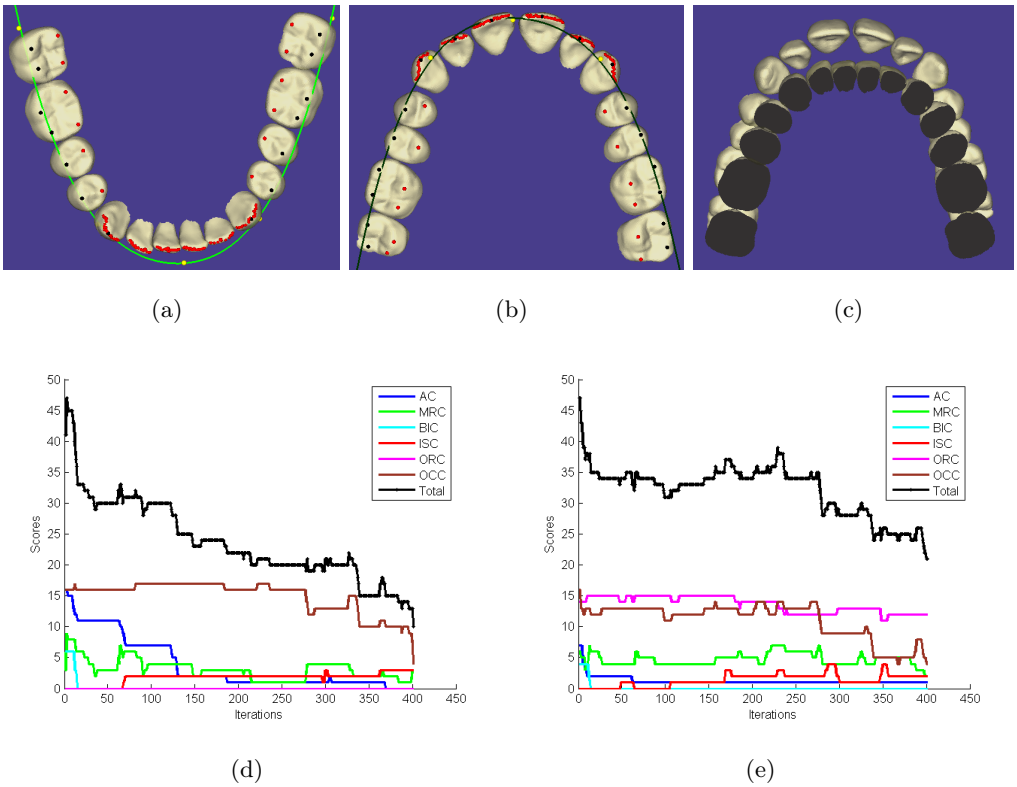


Figure 4.17: The simulation of teeth under the influence of constraint-based forces. (a) Lower Arch. (b) Upper Arch. (c) Wide space between the lower and the upper anteriors. Simulation history of the lower arch (d), and the upper arch (e). (See the caption of Figure 4.16 for a detailed explanation.)

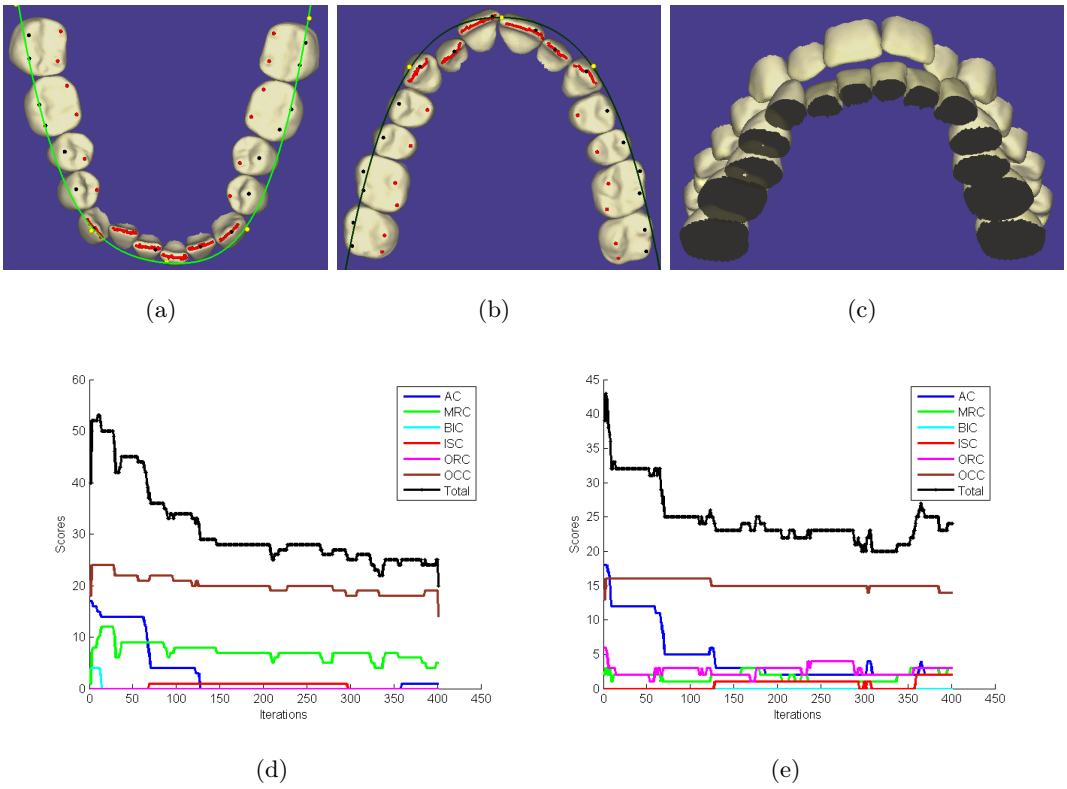


Figure 4.18: The simulation of teeth under the influence of constraint-based forces. (a) Lower Arch. (b) Upper Arch. (c) The left lateral incisor in upper arch is trapped among the lower anteriors. Simulation history of the lower arch (d), and the upper arch (e). (See the caption of Figure 4.16 for a detailed explanation.)

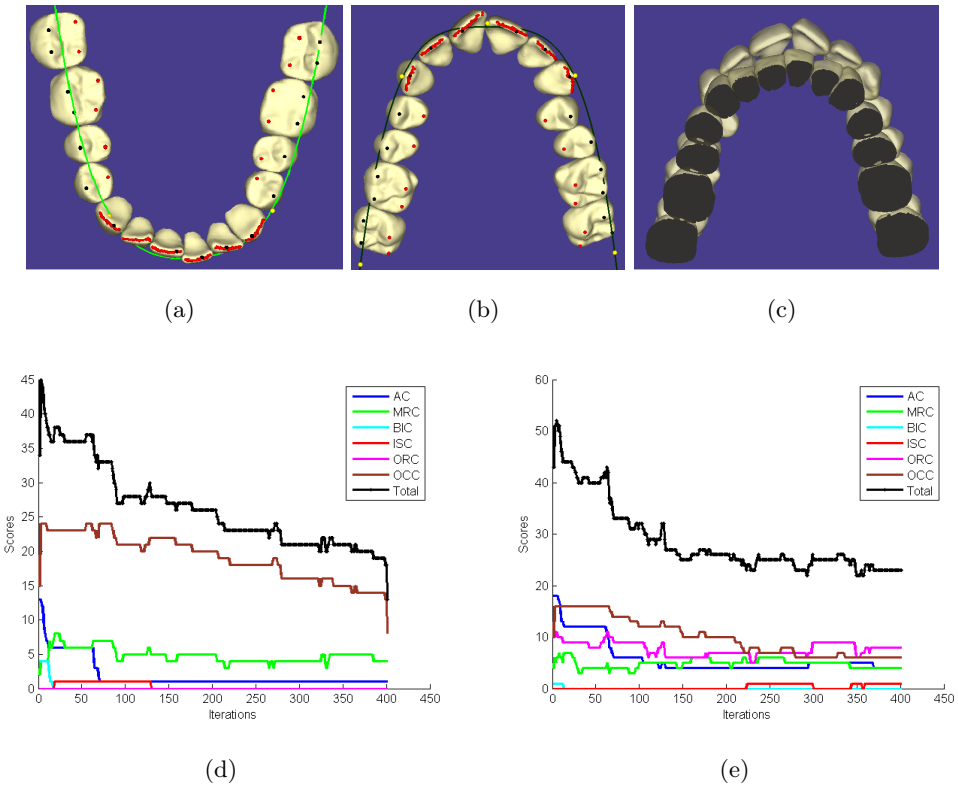


Figure 4.19: The simulation of teeth under the influence of constraint-based forces. (a) Lower Arch. (b) Upper Arch. (c) Crowding among the anteriors. Simulation history of the lower arch (d), and the upper arch (e). (See the caption of Figure 4.16 for a detailed explanation.)

Tooth type	Mesial-Distal (mm)	Buccal-Lingual (mm)	Vertical (mm)	Torque (°)	Tip (°)	Rotate (°)
Second molar	-0.05 ±1.94	-0.88 ±1.41	1.25 ±0.85	-9.53 ±10.11	-2.45 ±10.38	-2.22 ± 4.97
First molar	-0.06 ±1.28	-0.44 ±0.61	0.57 ±0.78	-2.62 ± 4.64	-2.07 ± 4.47	-0.97 ± 3.58
Second premolar	-0.39 ±2.68	-0.06 ±1.02	1.28 ±3.07	-2.60 ± 4.69	-11.66 ±30.33	2.15 ±12.36
First premolar	-0.03 ±1.31	0.18 ±0.75	1.91 ±0.94	2.18 ± 4.75	-0.31 ± 6.01	-4.93 ± 7.21
Canine	-0.66 ±1.19	0.09 ±0.79	1.00 ±1.38	2.60 ± 4.41	-9.55 ±18.90	-5.08 ±11.21
Lateral incisor	-0.11 ±1.46	1.15 ±0.99	0.25 ±1.90	3.99 ± 5.92	-7.62 ±20.94	-7.53 ±25.42
Central incisor	0.09 ±0.89	1.26 ±0.75	0.08 ±0.75	7.13 ± 3.49	-0.24 ± 7.28	0.48 ±10.20

Table 4.1: Discrepancies in position and orientation of the upper arch teeth across all the 8 cases (16 teeth of each type), in the outcome of the simulation compared to the arrangement established by an expert. The results are shown as mean values ± standard deviations. Negative values indicate a tooth position more distal, lingual or vertical, or with less buccolingual inclination, less mesial tip or buccal surface rotated more distally than the correct tooth in the post-treatment arrangement.

Tooth type	Mesial-Distal (mm)	Buccal-Lingual (mm)	Vertical (mm)	Torque (°)	Tip (°)	Rotate (°)
Second molar	-0.16 ±1.13	0.36 ±0.72	0.71 ±1.09	-1.67 ±5.85	-1.57 ±5.03	11.90 ± 6.02
First molar	-0.33 ±0.78	0.36 ±0.52	-0.15 ±1.01	-2.70 ±3.74	-0.38 ±2.52	11.15 ± 4.29
Second premolar	-0.40 ±1.09	0.08 ±0.75	0.43 ±1.21	-0.16 ±4.34	4.48 ±9.31	4.96 ±10.72
First premolar	0.06 ±1.21	0.19 ±0.95	-0.03 ±1.10	1.37 ±5.92	3.72 ±6.17	-7.27 ±10.65
Canine	0.15 ±1.17	-0.86 ±0.80	0.07 ±1.66	2.09 ±4.51	0.73 ±7.83	3.13 ± 9.45
Lateral incisor	-0.12 ±1.12	0.34 ±1.04	0.78 ±1.51	-0.73 ±7.01	6.15 ±5.61	10.45 ±16.28
Central incisor	-0.05 ±0.81	0.14 ±1.33	0.61 ±1.19	-3.40 ±4.72	4.01 ±6.56	4.20 ± 5.81

Table 4.2: Discrepancies in position and orientation of the lower arch teeth across all the 8 cases. (See the caption of Table 4.1 for a detailed explanation.)

Chapter 5

Conclusions and future work

5.1 Summary of the thesis

Digital orthodontics is a rapidly growing field as orthodontists migrate towards computer-based systems for treatment planning. Although virtual simulation tools have the potential to produce very effective treatment plans, there is a gap between the number of orthodontic patients who get their digital models made as part of the treatment process and the number of patients who are actually treated using virtual plans. This is mainly because all currently available tools still require a lot of time-consuming manual interaction from the orthodontist. Our goal in this thesis has been to bridge this gap, by automating the major tasks in virtual treatment planning.

First, in Chapter 2, we presented an algorithm to segment a 3D dental arch into individual tooth objects. The main challenges in dental segmentation arise from insufficient or noisy information on the gum-teeth boundary or the interproximal contact region. These are, in turn, due to malocclusions and crowding of teeth found in the dental arches of orthodontic patients. Our solution overcomes these limitations by first dividing the problem into two subproblems (Gum-Teeth Separation and Tooth-Tooth Separation) and then solving these sequentially using separate methods for each subproblem.

Next, in Chapter 3, we described techniques for the problem of identifying certain key features on tooth surfaces that are essential for downstream processing during the alignment of teeth. Our methods use buccolingual and mesiodistal cross-sections of the

tooth surface to facilitate a "guided" extraction of incisal edges, central grooves, marginal ridges, cusp ridges, and occlusal surface boundary. We also described a watershed-based cusp identification algorithm.

Finally, in Chapter 4, we presented a technique for the alignment of teeth to achieve the best possible dental occlusion. Given the teeth in the upper and lower arches, we obtain a near-optimal dental occlusion via a simulation-based approach that models the alignment process as a spring-mass system where teeth are considered to be masses connected by springs representing dental constraints which are, in turn, modeled via the features identified in Chapter 3. These constraints are obtained from the experience of practitioners and the well-known approaches to objectively classify the commonly occurring orthodontic errors. The set of chosen constraints guide the simulation algorithm to find an arrangement that best satisfies all the given constraints subject to certain boundary conditions.

All of the algorithms developed in this thesis have been implemented in a software tool. The tool is written in C++ in the environment provided by Qt Open-Source Edition 4.5 [42]. It includes intuitive graphical user interface for each of the three main modules (segmentation, feature identification, and alignment). All modules have been evaluated by domain experts on actual clinical data.

5.2 Future work

This section describes some directions for future work in the field of virtual treatment planning in orthodontics. Most of the interesting future directions lies in the alignment step and the step that succeeds it, i.e., *placement* of brackets and wires. Due to wide variations in real-world cases, we believe that a significant amount of improvement is difficult to achieve in the segmentation and feature identification steps.

5.2.1 Improved alignment of teeth

The alignment step has many potential areas for improvement and extension to make it more useful when considered in terms of its deployment in a clinical setting. First, the current implementation of the simulation-based approach as described in Section 4.5, does not handle well the dental models with missing teeth, and Class II or Class III

occlusal relationships. Second, as we mentioned earlier, there are no established metrics to evaluate the quality of occlusion in a given dental model and this presents significant challenges in designing an algorithm to automatically find the arrangement with the "optimal" occlusion. Third, the simulation can be made more biologically-realistic by using the tooth's root information from an alternative source such as a CT-scan, instead of relying only on tooth crown information from laser-scans. Finally, the simulation process itself will be greatly enhanced through the use of modeling techniques that take into account the physical properties of the jaw bone and the teeth. This will bring the virtual treatment planning and simulation very close to the actual treatment process and will, thus, help generate accurate, patient-specific treatment plans. We envision the use of the finite-element method as a key technique here.

5.2.2 Placement of orthodontic devices

The placement step, which involves correctly placing the orthodontic devices (brackets, wires, etc.) on the teeth, was not addressed in this thesis. This is an important step in a real-world treatment planning system, and current solutions involve extensive user-input to determine the correct positions of brackets. There is wide variety of commercially available orthodontic devices which differ in shapes and designs, which in turn influences their properties during the treatment process. Thus, the lack of accurate computational models to simulate the long-term effects of orthodontic devices makes the automation of the placement step more challenging.

5.2.3 Use of machine learning techniques

Another valuable, but largely unexplored aspect of virtual tools in treatment planning is the availability of a large number of existing orthodontic cases that have been planned and treated by the experts. Given the widespread use of digital technology to create and virtually plan the treatment process, a large number of "solved" orthodontic cases are recorded in the digital archives of dental clinics. These archives, in addition to containing the virtual treatment plans created by an expert for a given input case, also record information on tooth position during and after the actual treatment (typically recorded

as an unsegmented surface-scan of the arches). One can envision the creation of machine learning-based computational models of tooth movement in orthodontic treatment planning that are founded on the statistical observations on a large number of existing cases.

As an example, consider a mathematical model to track the movement (trajectory) of certain important features (cusps) on the tooth crowns during various stages of treatment. This can be used to create a classification of the patterns of movement and relate these to the arch types and initial configuration of teeth. These statistical models may be used to predict the tooth movement in a new case based on similar ones in the database.

References

[1] Guide for setting up teeth on F/F dentures. `http://tinyurl.com/83p45dx`.

[2] 3Shape OrthoAnalyzer Software^{TM}. `http://tinyurl.com/d26agep`.

[3] S. J. Aarseth. *Gravitational N-Body Simulations: Tools and Algorithms*. Cambridge University Press, 2003.

[4] A. Agathos, I. Pratikakis, S. Perantonis, N. Sapidis, and P. Azariadis. 3D mesh segmentation methodologies for CAD applications. *Computer-Aided Design & Applications*, 4(6):827–841, 2007.

[5] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun. Anisotropic polygonal remeshing. *ACM Trans. on Graphics*, 22(3):485–493, 2003.

[6] L. F. Andrews. The six keys to normal occlusion. *Am. J. Orthod.*, 62(3):296–309, 1972.

[7] E. H. Angle. *Treatment of malocculsion of the teeth*. University of Michigan Library, 1907.

[8] M. Attene, B. Falcidieno, and M. Spagnuolo. Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer*, 22(3):181–193, 2006.

[9] M. Attene, S. Katz, M. Mortara, G. Patane, M. Spagnuolo, and A. Tal. Mesh segmentation - A comparative study. In *SMI '06: Proc. of the IEEE Intl. Conf. on Shape Modeling and Applications*, pages 7–18, 2006.

[10] D. Baraff. Physically based modeling. SIGGRAPH course notes, 2001.

[11] E. A. Begole. Application of the cubic spline function in the description of dental arch form. *J. Dent. Res.*, 59(9):1549–1556, 1980.

[12] S. Braun, W. P Hnat, and D. E. Fender. The form of the human dental arch. *Angle Orthod.*, 68:29–36, 1998.

[13] Y. Chang, J. J. Xia, J. Gateno, Z. Xiong, X. Zhou, and S. T. C. Wong. An automatic and robust algorithm of reestablishment of digital dental occlusion. *IEEE Trans. on Medical Imaging*, 29(9):1652–1663, 2010.

[14] D. Chetverikov and Z. Szabo. A simple and efficient algorithm for detection of high curvature points in planar curves. *Lecture Notes in Computer Science*, 2756/2003:746–753, 2003.

[15] D. B. Diggs. *The quantification of arch form.* PhD thesis, University of Washington, 1962.

[16] D. Eismann. Reliable assessment of morphological changes resulting from orthodontic treatment. *Europ. J. Orthod.*, 2:19–25, 1980.

[17] T. Gatzke and C. M. Grimm. Estimating curvature on triangular meshes. *Intl. J. of Shape Modeling*, 12(1):1–28, 2006.

[18] N. Germane, S. J. Lindauer, L. K. Rubenstein, J. J. Revere Jr, and R. J. Isaacson. Increase in arch perimeter due to orthodontic expansion. *Am. J. Orthod. Dentofacial Orthop.*, 100(5):421–427, 1991.

[19] J. Goldfeather and V. Interrante. A novel cubic-order algorithm for approximating principal direction vectors. *ACM Trans. on Graphics*, 23(1):45–63, 2004.

[20] E. Gottlieb. Grading your orthodontic treatment results. *J. Clin. Orthod.*, 9:156–161, 1975.

[21] L. T. Hiew, S. H. Ong, and K. W. C. Foong. Optimal occlusion of teeth using planar structure information. *Machine Vision and Applications*, 21(5):735–747, 2010.

[22] D. D. Hoffman and W. A. Richards. Parts of recognition. *Cognition*, 18(1-3):65–96, 1984.

[23] D. D. Hoffman and M. Singh. Salience of visual parts. *Cognition*, 63(1):29–78, 1997.

[24] B. Hultgren, R. Isaacson, and M. Marshall. Digital Model Comparison software (*e*model compare 8.1), 2011.

[25] Invisalign®. `http://www.aligntech.com`.

[26] P. Jiménez and F. Thomas. 3D collision detection: A survey. *Comp. & Graph.*, 25:269–285, 2001.

[27] R. G. Keim, E. L. Gottlieb, A. H. Nelson, and D. S. Vogels III. 2008 JCO study of orthodontic diagnosis and treatment procedures, part 1: Results and trends. *J. Clin. Orthod.*, 42(11):625–640, 2008.

[28] R. Klette and A. Rosenfeld. *Digital geometry.* Morgan Kaufman, 2004.

[29] T. Kondo, S.H. Ong, and K.W.C. Foong. Tooth segmentation of dental study models using range images. *IEEE Trans. on Medical Imaging*, 23(3):350–362, 2004.

[30] Y. Kumar, R. Janardan, and B. Larson. Automatic feature identification in dental meshes. Submitted to a journal, 2012.

[31] Y. Kumar, R. Janardan, and B. Larson. Automatic virtual alignment of dental arches in orthodontics. Submitted to a journal, 2012.

[32] Y. Kumar, R. Janardan, B. Larson, and J. Moon. Automatic feature identification in dental meshes. *Wksp. on Mesh Processing in Medical Image Analysis (MeshMed)*, 2011.

[33] Y. Kumar, R. Janardan, B. Larson, and J. Moon. Improved segmentation of teeth in dental models. *Comp.-Aided Design. and Appl.*, 8(2):211–224, 2011.

[34] B. H. Le, Z. Deng, J. Xia, Y. Chang, and X. Zhou. An interactive geometric technique for upper and lower teeth segmentation. In *MICCAI (1)*, volume 5762 of *Lecture Notes in Computer Science*, pages 968–975, 2009.

[35] Y. Lee, S. Lee, A. Shamir, D. Cohen-Or, and H. Seidel. Intelligent mesh scissoring using 3D snakes. In *Proc. Comp. Graphics and Appl.*, pages 279–287, 2004.

[36] Y. Lee, S. Lee, A. Shamir, D. Cohen-Or, and H. Seidel. Mesh scissoring with minima rule and part salience. *Computer-Aided Geometric Design*, 22(5):444–465, 2005.

[37] D. L. Logan. *A First Course in the Finite Element Method*. Fifth Edition, Cengage Learning, 2012.

[38] M. Mokhtari and D. Laurendeau. Feature detection on 3D images of dental imprints. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 2359, pages 619–630, September 1994.

[39] A. P. Mangan and R. T. Whitaker. Partitioning 3D surface meshes using watershed segmentation. *IEEE Trans. on Vis. and Comp. Graphics*, 5(4):308–321, 1999.

[40] M. Meyer, M. Desbrun, P. Schröder, and A. Barr. Discrete differential geometry operators for triangulated 2-manifolds. *VisMath.*, pages 35–54, 2002.

[41] C. L. Ming and S. Gottschalk. Collision detection between geometric models: A survey. *In Proc. of IMA Conf. on Math. of Surfaces*, 1:1–20, 1998.

[42] D. Molkentin. *The Book of Qt 4: The Art of Building Qt Applications*. No Starch Press, San Francisco, CA, USA, 2007.

[43] J. Moon. Evaluation of software developed for automated segmentation of digital models. Master's thesis, University of Minnesota, 2011.

[44] J. Moon, B. Larson, R. Janardan, and Y. Kumar. Evaluation of software developed for automated segmentation of digital models. *IADR/AADR/CADR 89th General Session and Exhibition*, 2011.

[45] H. Noroozi, T. H. Nik, and R. Saeeda. The dental arch form revisited. *Angle Orthod.*, 71:386–389, 2001.

[46] American Board of Orthodontics. `http://tinyurl.com/n8bbwl`, 2009.

[47] American Board of Orthodontics. Grading system for dental casts and panoramic radiographs, 2010. `http://tinyurl.com/ygstyfu`.

[48] D. L. Page, A. F. Koschan, and M. A. Abidi. Perception-based 3D triangle mesh segmentation using fast marching watersheds. In *CVPR '03: Proc. of the Intl. Conf. on Computer Vision and Pattern Recognition*, pages 27–32, 2003.

[49] S. Pulla, A. Razdan, and G. Farin. Improved curvature estimation for watershed segmentation of 3-dimensional meshes. Manuscript, 2001.

[50] C. Rössl, L. Kobbelt, H. Seidel, and I. Stadtwald. Extraction of feature lines on triangulated surfaces using morphological operators. *AAAI Symp. on Smart Graphics*, pages 71–75, 2000.

[51] M. Sarfraz, A. Masood, and M. R. Asim. A new approach to corner detection. *Computer Vision and Graphics*, pages 528–533, 2006.

[52] SureSmile®. `http://www.suresmile.com/`.

[53] *e*model®8.5. `http://www.geodigmcorp.com`.

[54] T. Weinkauf and D. Günther. Separatrix persistence: Extraction of salient edges on surfaces using topological methods. *Computer Graphics Forum (Proc. SGP '09)*, 28(5):1519–1528, July 2009.

[55] J. B. Woelfel and R. C. Scheid. *Dental Anatomy: Its Relevance to Dentistry*. Lippincott Williams & Wilkins, 2011.

[56] Y. Tian-ran, D. Ning, H. Guo-dong, C. Xiao-sheng, C. Hai-hua, L. Wen-he, Y. Qing, and L. Peijun. Bio-information based segmentation of 3D dental models. *Intl. Conf. on Bioinformatics and Biomedical Engineering*, pages 624–627, 2008.

[57] M. Zhao, L. Ma, W. Ta, and D. Nie. Interactive tooth segmentation of dental models. *27th IEEE Intl. Conf. on Engineering in Medicine and Biology*, pages 654–657, 2005.