Improved Segmentation of Teeth in Dental Models

Yokesh Kumar[1], Ravi Janardan[1],Brent Larson[2]and Joe Moon[2]

[1]Computer Science, University of Minnesota, {kumaryo,janardan}@cs.umn.edu
[2]Developmental and Surgical Sciences,University of Minnesota, (larso121, moonx162}@umn.edu

ABSTRACT

The identification of different teeth in a human dental model is an essential part of virtual treatment planning in digital orthodontics. An algorithm that is almost fully automatic is presented to segment a 3-dimensional dental model (a triangle mesh) into individual tooth objects, along with its implementation in a software tool that can be deployed in clinical settings. The proposed technique avoids the limitations of previous approaches by first separating gums from teeth and then separating individual teeth. The only user intervention needed in most cases is the one-time setting of a certain curvature threshold via an intuitive slider, coupled with real-time visual feedback. The method is found to be robust across a range of real-world models, even in the presence of malocclusions and noise.

Keywords:3-Dmesh, tooth segmentation, digital orthodontics.
DOI: 10.3722/cadaps.2011.211-224

1 INTRODUCTION

Virtual 3-dimensionaltreatment simulation has recently become possible in orthodontics, but its application in routine clinical patient care has been limited. This is due to the need for significant intervention to segment the dental arch model into individual tooth objects and to (re)align these tooth objects on the arch according to prescribed criteria.

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 2-dimensional (2D) to 3-dimensional (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 be reasonably 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 [3].)

Virtual treatment simulation has been incorporated as part of new orthodontic treatment techniques such as Invisalign® and SureSmile®. 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 and software for orthodontics, such as *e*model® *8.0*, from GeoDigm Corp. Fig 1(left)shows the use of *e*model® *8.0 to* segment an upper incisor by manually defining a region of the digital model and Fig1(right)shows the manual alignment of the four upper incisors after segmentation.

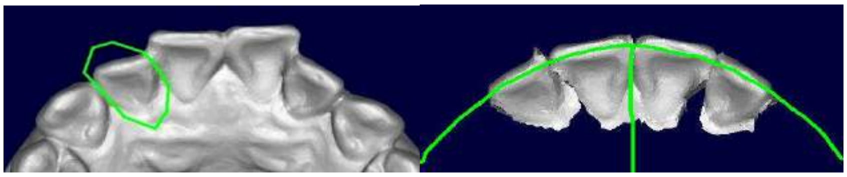


Fig.1: Manual segmentation (left) and alignment (right). (All figures in the paper are best viewed in color.)

Unfortunately, currently available tools for segmentation 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 [8]indicates that actively practicing orthodontists are using simulations, or set-ups, only 2.5% of the time and yet18% of patients now have digital models made as part of the information gathering process prior to treatmentand the use of digitals models is accelerating. This discrepancy between the use of digital models and the number of patients that 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 completing the segmentation and alignment process.

In this paper, we study the fundamental problem of automaticallysegmenting the teeth in a 3D model of the human jaw into individual tooth objects.We present an almost fully automatic algorithm to segment the model (represented as a triangle mesh) into individual tooth objects. We also present an implementation of the algorithm in a software tool that is suitable for deployment in a clinical setting. Our technique avoids the limitations of previous approaches by first separating gums from teeth and then separating individual teeth. In most cases, the only user intervention needed is the one-time setting of a certain curvature threshold via an intuitive slider, coupled with real-time visual feedback. (On some difficult cases, additional user interaction may be required, as discussed in Section 4.)A problem of independent interest that we also investigate is the automatic discrimination of gum-tissue from the teeth, which is crucial for model visualization.

The starting point for our work is a 3D triangular mesh representation of the digital model obtained from a laser scan of a plaster model built from dental impressions. This is the data modality used predominantly in the clinic. Therefore, our segmentation problem is quite differentfrom the traditional image segmentation approaches that work with 2D views of the dental model.

1.1 Related Work

Related work on segmentation can be classified as 2D methods (e.g., [9]), which work with a constant number of plan-views/projections, or 3D methods (e.g., [16,17]), which are based on general mesh segmentation techniques such as in [1,2]. The 2D method in [9] requires significant user input to handle malocclusions and imperfections in the gum-teeth boundary and the loss of spatial information that accompanies the projection of the mesh to 2D makes it difficult to find good separating boundaries for the teeth. The 3D methods in [16,17] are driven by the negative minima rule [7,11] and proceed by finding *feature lines* on the mesh (curves of minimum negative surface curvature which serve as boundaries between teeth).

There are several limitations to the existing 3D methods. For example, the approach in [16] requires the user to patch broken feature lines interactively via mouse clicks, while the approach in [17] requires the user to click on many interstitial points between teeth to connect broken feature lines. Moreover, existing approaches are not good at handling noise in the gum regions because these techniques are simply “extensions” of generic meshsegmentation algorithms applied to dental models and do not use any domain-specific information about the characteristics of dental models. This translates to unacceptably long delays and increased costs in the clinic. Ideally, the majority of the cases should be handled completely automatically, without user intervention once the data file is uploaded and initial parameters are set; interactive, user-driven segmentation should be reserved only for the most difficult or unusual cases.

Very recently, 3D snake-based operators have been applied to the problem of tooth segmentation [10]. This approach is superior to thosein[16,17] and produces much better results by exploiting some domain-specific information about dental models like the position of teeth/gums in the jaws and the presence of cusps or vertices with high positive curvature on most of the tooth objects.However, there is not much information available about the amount of manual interaction needed in the intermediate stages of this algorithm. It is also worth mentioning that this algorithm and the one that we present in this paper weredeveloped simultaneously and independently, and both papersidentify similar shortcomings in the previous approaches.

1.2 Challenges of Dental Segmentation

Robust automatic dental segmentation presents several computational challenges some of which have been noted in [10, 16, 17]. First, although negative minima-based feature regions help identify the boundaries of interest, i.e., the gum-teeth boundary (*gumline*) and tooth-tooth boundary (*interstitial points*), their computation requires reliable curvature estimation on the mesh [5].The curvature thresholds used to identify meaningful feature regions vary across different models even after normalization and, hence, cannot be predetermined. Thus, feature regions must be refined using morphological operators [14] or manually[16, 17].

Second, morphological operators work well only when the gaps in feature lines to be repaired are small (a fewvertices wide). Moreover, in an automated setting, they are applied globally on all feature regions and, hence, can affect portions of some already well-formed feature regions when attempting to repair others.

Third, dental models from laser scans sometimes have scanning errors due to malocclusions. These lead to poor accuracy in the inter-teeth boundaries captured by the scan and make their detection via curvature thresholding difficult. Also, it is hard to distinguish between gums and teeth in some places.

Finally, automatic segmentation must also account for noise in the model, especially in the upper jaws, in the gumline, and in the molar ridges.

2 OUR APPROACH TO SEGMENTATION

We proceed by solving the following problems:

1) **Gum-Teeth separation.**Classify the gums and teeth vertices of the mesh correctly.
2) **Tooth-Tooth separation.**Identify the separating 3D boundary curves between adjacent teeth.

We first solve 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 demands more precision and work with different kinds of noise on the surface compared to the Gum-Teeth separation (the distinction will become clear in our discussions below).

Gum-Teeth separation has other desirable applications in orthodontics. It results in better visualization of dental models, with gums and teeth in their natural coloras shown in Fig.3(f). Also, the absence of gums enables more accurate study of the dynamics of simulated tooth movement over time under the influence of forces and collisionswith other teeth. Below we summarize the various steps involved in our approach to segmentation of dental arch into individual teeth.

GUM-TEETH SEPARATION

S1) Remove the bases and compute the curvature estimates at the vertices of the mesh and obtain feature regions as regions of high-curvature vertices(Figure 2(a-c)).

S2) Refine the feature regions and use a "flood-fill" method to separate gums from teeth and form connected teeth components (Figure 2(d)).

S3) Extract and repair the gumline, and re-initialize the teeth and gum components of the mesh(Figure 2(d-f)).

TOOTH-TOOTH SEPARATION

S4) Split the gumline into lingual (tongue-side) and labial (outside) chains and detect interstitial points (corners) on gumline (Figure 3(a-b)).

S5) Match interstitial points from lingual and labial chains in pairs and construct separating curves between adjacent teeth as the shortest path (along the mesh surface) between each pair of matched interstitial points (Figure 3(c-e)).

S6) 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 (Figure 3(f)).

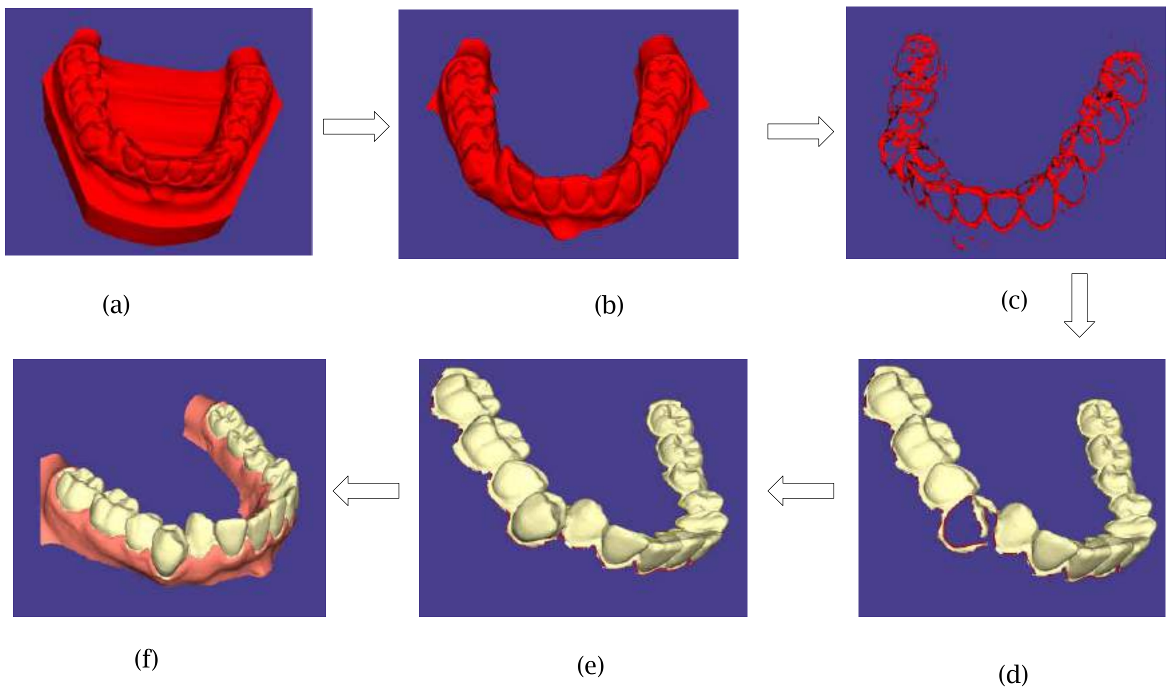


Fig.2: Stages in Gum-Teeth separation: (a) 3D model, (b) removal of bases, (c) curvature thresholding, (d) results of flood-fill (gums not shown) and gumlinedetection (gumline is the red curve), (e) gumline repair, and (f) final gum-teeth separation.

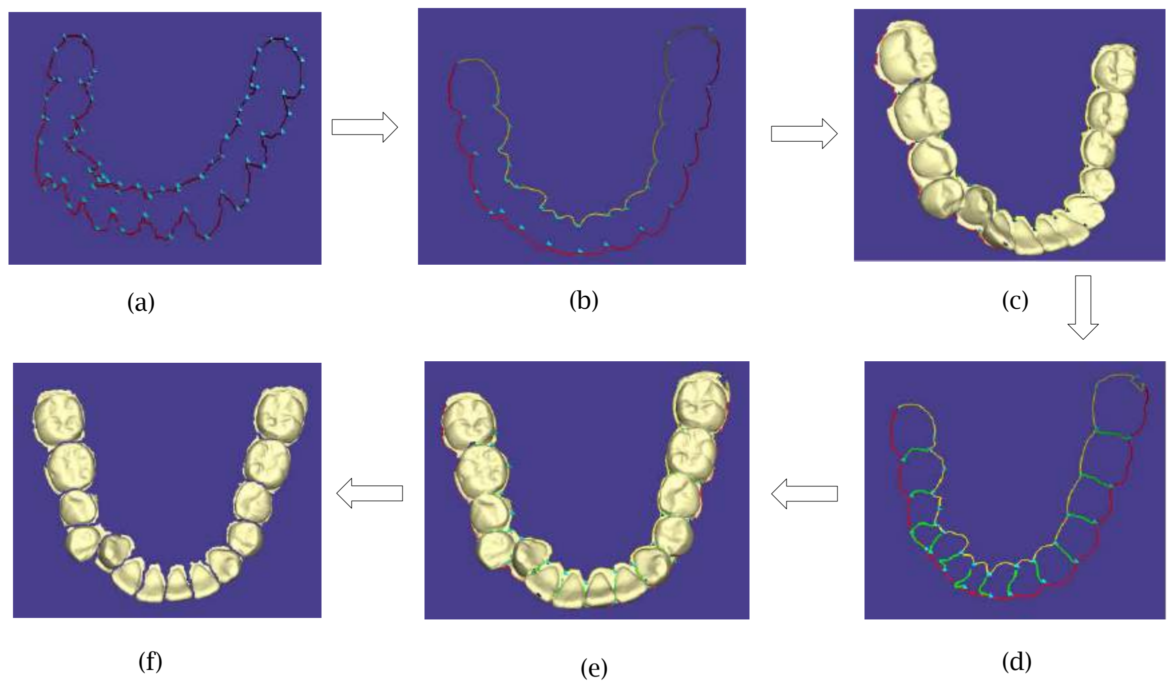


Fig.3: Stages in Tooth-Tooth separation: (a) find corners on gumline, (b) split the gumline into lingual and labial sides (shown as red and yellow curves), and retain concave corners, (c) concave corners on surface of teeth, (d) separator curves between matched corners, (e) separator curves on teeth, and (f) teeth component separated into individual tooth meshes.

2.1 Gum-Teeth Separation

After removingthe bases (Figure 2 (b)), we begin with the computation of curvature estimates at each vertex of the mesh[12]. After this we filter out the vertices with principal maximum curvature below a certain threshold to form feature regions according to the *negative minima rule*[7]. This threshold was found to vary in different models and, thus, a single value does 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(c) shows the feature regions after curvature 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 visually the changes in features regions in real-time. This is the only interactive part in our system and it is far less time-consuming than the user interaction component of previous systems. From our studies on a wide 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.

Although the exact curvature threshold varies widely across different models, for a given model, there is often awide enough range of curvatures for which the results are similar. For example, after normalization of curvatures to map them to[0,1], the thresholds of 0.4 and 0.6 producedthe best results on two different models A and B, respectively.We observed that a threshold of 0.4±0.15 and 0.6±0.1 works just as well on A and B, respectively. So, even though a user has to set a value of curvature via the slider, there is enough flexibility so that it is easy to use.

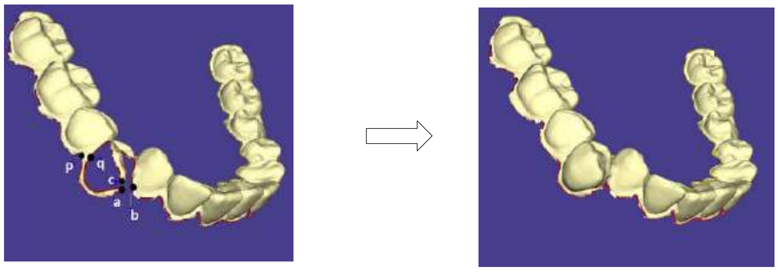
2.1.1 *Feature Region Computation and Flood-fill*

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 the teeth, defining the natural boundary between the teeth and gums, i.e.,thegumline. However, due to scanning errors, sometimes the gumline is not represented prominently in the model[10]. When the gaps in the feature regions are small (a few vertices wide), the morphological operators from[14]can be used to "close" the contour. 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.

Given a set of feature regions, 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 reached as being in the gum regions. This BFS flooding is constrained to not cross any feature vertices. As a result of this "flood-fill" BFS, we get connected regions representing gums and teeth.If there are missing teeth or wide gaps between adjacent teeth, we may get more than one teeth component, and hence, several simply-connected gumline curves around each teeth component. (If there are multiple teeth components (and gumline components), thealgorithmhandles these individually.) Also, the noisy feature regions in the gums 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 Section 4).

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(d)whichillustrates missing portions of teeth that are 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 (our implementation requires at least four unvisited vertices in the 2-neighborhood of a vertex to advance the BFS at that vertex). Figure 2(f) shows the output of the flood-fill in case when the gumline is prominently represented in feature regions.

Furthermore, even in the case when flood-fill 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 2.1.2). To this end, we also use the vertices on the ridges of anteriors and cusps of molars as feature vertices. This prevents the BFS from advancing 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 and are represented quite prominently in the mesh.



(a) (b)

Fig.4: Gumline repair process: (a) defective gumline (shown as red curve) and intervals, and (b) teeth and gumline after repair (gums are not shown).

2.1.2 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 4(a). Our goal is to repair such an initial estimate of the gumlineto obtain the curve shown in Figure 4(b) which is a much better approximation to the true gumline of the model. Our experiments show that this is often required in real-world dental models. Considering both these curves at the places where they differ, notice that the "good"gumline (Figure 4(b)) seems much smoother in contrast to the "bad" one (Figure 4(a)) and can be easily distinguished from the former. Thus, our working 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.

2.1.2.1 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 $C = v_0, v_1, \dots, v_{n-1}$ (v_0 is the first and last vertex on the closed 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 $[v_i, v_j] = v_i, v_{i+1}, \dots, v_j$. Figure 4(a) 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:

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

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.(Since L and M can vary depending on the mesh, in terms of the distance along the curve, L (resp. M) can be chosen to be a window of length 2 mm (resp. 20 mm) along the curve.) 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 $v_i, v_{i+1}, \dots, v_j$ on C denoted as P_{ij}. Let the shortest path from v_j to v_i be $S_{ji} = v_j, v_{k1}, v_{k2}, \dots, v_i$. Concatenate S_{ji} with P_{ij} to obtain the closed circuit $Z_{ij} = v_{i+1}, \dots, v_j, v_{k1}, v_{k2}, \dots, v_i$. 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*). For example, in Figure 4(a), $[p, q]$ 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). Thus, a high score indicates that the corresponding interval is more invalid and must be repaired.

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 (see Section 2.1.2.4).

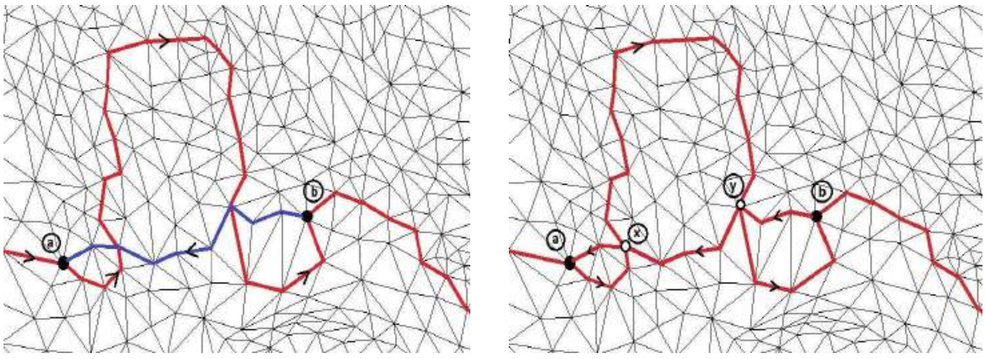
Some examples of candidate intervals in Figure 4(a) are $[p,q]$, $[a,c]$, $[c,b]$and$[a,b]$. It is clear that we must prefer to repair $[a,b]$ among these intervals. Note that if we choose $[p,q]$ prior to $[a,b]$, we may not even get the correct gumline after repairing. Also, repairing $[a,c]$ and $[c,b]$ 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 $\mathcal{J} \subseteq \mathcal{C}$ 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.)

Problem **IIM**is solved in Section2.1.2.3 and we show how to repair the set $\mathcal{J}$ thus found in Section2.1.2.4. We begin with discussion on the key subproblem of classifying intervals as Type-T or Type-G in Section 2.1.2.2.

2.1.2.2 Deciding the Type of Interval

We orient the edges of the pathP_{ij} (resp. S_{ji}) as going away from v_i (resp. v_j) to get a natural predecessor/successor relation among the vertices on the path (Figure 5(a) shows the orientation of the edges). (See Section 2.1.2.1for definitions.)



(a) (b)

Fig.5: Deciding the type of interval: (a) An interval $\boldsymbol{P_{ab}}$ on the gumline (shown in red) and the shortest path $\boldsymbol{S_{ba}}$ (shown in blue), and (b) A self-crossing $\boldsymbol{Z_{ab}}$ obtained by concatenating $\boldsymbol{P_{ab}}$ with $\boldsymbol{S_{ba}}$. We can see that $\boldsymbol{Z_{ab}}$ is decomposed into edge disjoint circuits: (1) $\boldsymbol{a \to x \to a}$, (2) $\boldsymbol{y \to x \to y}$, and (3) $\boldsymbol{b \to y \to b}$. Note that the order of vertices in the original curves from (a) is preserved in the new curves as well.

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: First we decompose the given Z_{ij} into closed edge-disjoint sub-circuits $Z_{ij}^{(1)}, Z_{ij}^{(2)}, \ldots, Z_{ij}^{(k)}$ such that these are not self-crossing. An example of such a decomposition is shown in

Figure 5(b). 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 to form e_{ij}.

2.1.2.3 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 $\mathcal{J} \subseteq \mathcal{C}$ that are pairwise non-overlapping such that the sum of the lengths of the intervals in $\mathcal{J}$ is maximized. A set of non-overlapping intervals is called a *feasible* set. Below we describe a technique to find $\mathcal{J}$. (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 $\mathcal{J}$ 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 4(a), 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 in $\Theta(N \log N)$ time.

2.1.2.4 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. Figure 6show some examples of invalid intervals and the result their repair.

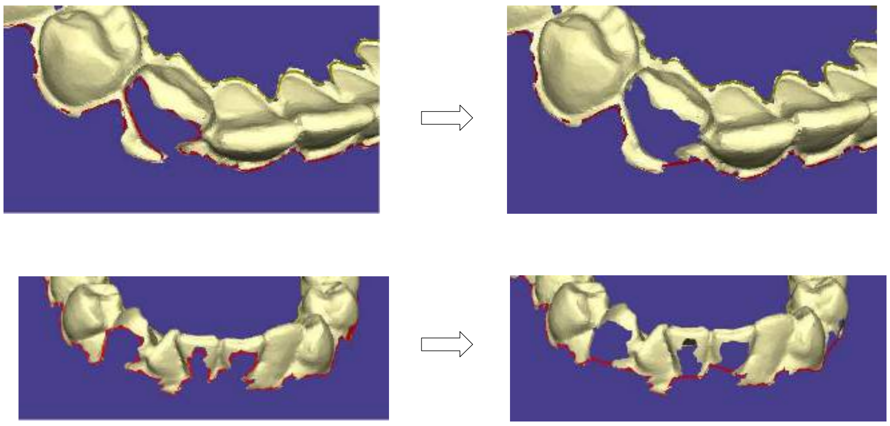


Fig.6: Example of gumline repair on two different models. The left sideshows two defective gumlines which are corrected as shown on the right side.

2.2 Tooth-Tooth Separation

At this stage, we have separated gums from teeth and extracted the gumline. The connected gumline is split at its vertices at theextreme upper-left and upper-right extents to create the lingual and labial sections. Our next task is to separate the teeth components on the mesh into individual tooth objects. This is done in three stages:

2.2.1 Detecting Interstitial Points on Gumline

The *corners* on the 3D gumline curve are vertices of high-curvature (w.r.t the curve, not the mesh), i.e., where the curve bends sharply. (See [4,15] for somecommonly useddefinitions of corners in curves and techniques to compute them.)The interstitial points between teeth are included among the *concave* corners of the gumline curve and we find them by extending techniques for corner detection in 2D curves [15]. The algorithm is quite robust to local irregularities on the gumline (see Figure 3(a-b)).

2.2.2 Matching Interstitial Points

Once the interstitial points on the gumline have been found, we match the ones from lingual side to their corresponding ones on labial side. A practical requirement of our matching algorithm is that it must produce as few false positive pairings as possible, because false positives lead to spurious separators. We use an algorithm based on Euclidean nearest neighbors to match corners on the chain with fewer corners to a subset of corners on the other chain.

2.2.3 Drawing Separating Curves

We 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 the interstitial points detected are not too far away from ground truth, these separating curves pass through the valley regions between adjacent teeth. This gives a very natural separation between adjacent teeth as shown by the green curves in Figure 3(d-e) (see also Table 1 (row 4) for more results). The teeth components are divided by cutting along these curves to get isolated tooth objects as shown in Figure 3(f).

3 SOFTWARE TOOL FOR SEGMENTATION

We have implemented the algorithms described in this paper in a tool written in C++ in the environment provided by Qt Open-Source Edition 4.5 [13]. It consists of a simple, intuitive user interface that can be used to load dental models and segment them. Figure 7 shows the screenshot of the tool with all its commands. 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 algorithms given in Section2. 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 for the setting of the curvature threshold, which is quite intuitive and common in such systems (we discussed briefly in Section2.1.1 why this may be necessary for any automatic dental segmentation approach).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.

Thus, this system provides a tool which can be used in a real-world clinical environment with minimal user interaction. (The software executable is available from the authors upon request.)

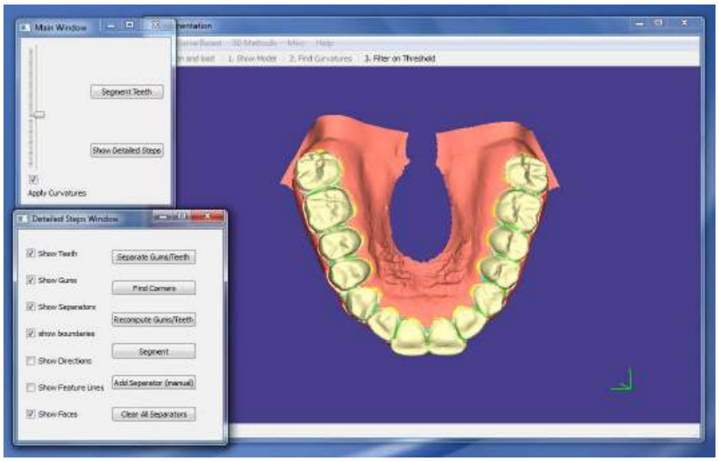


Fig.7: Screenshot of the segmentation tool. The window on the right displays the graphics. The upper-left window has the controls to setup a curvature threshold and initiate teeth segmentation. Thelower-left window, with detailed steps and view controls, can be opened by using the “Show Detailed Steps” button on the window above.

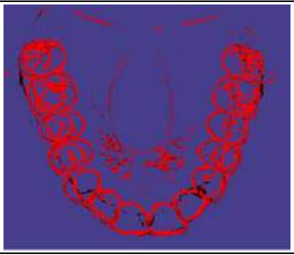
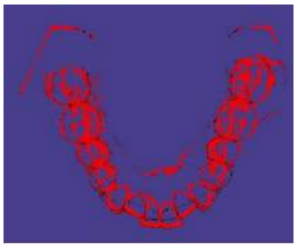
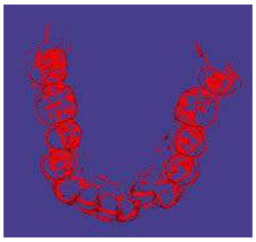
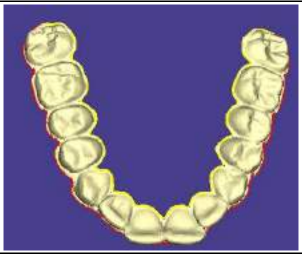
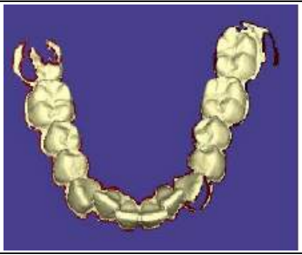
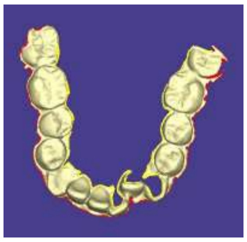
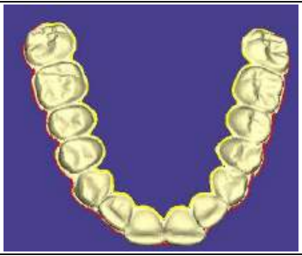
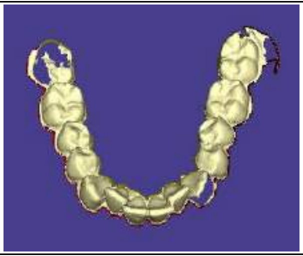
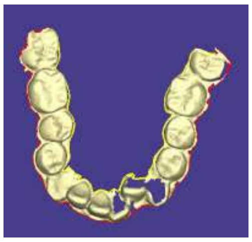
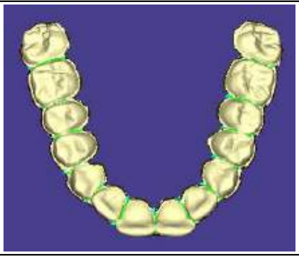
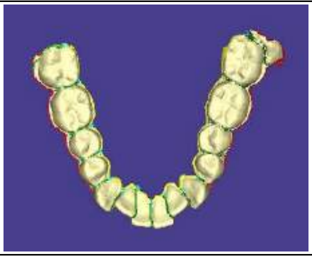
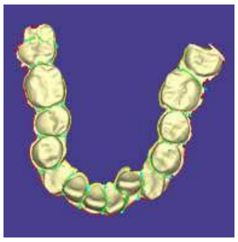
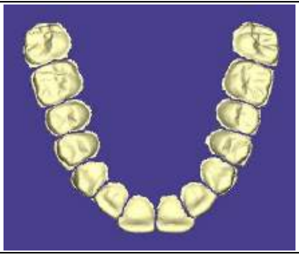
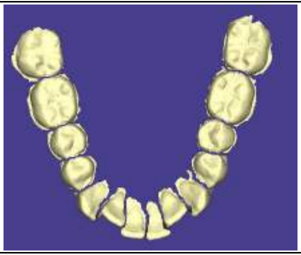
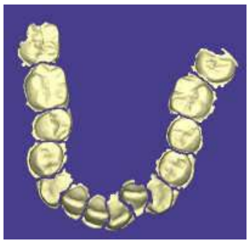
4 EVALUATION OF RESULTS

We have tested our approach on 26 upper and lower jaw models of varying complexity. The datasets included laser-scans of plaster models obtained from two different commercial scanners. Our system performed with reasonable accuracy on most of the models, requiring user interaction only to set the curvature threshold on most of them (as expected). The software system is currently being tested in a clinical setting to further improve its efficiency and user experience.

The approach closest to our work is that of [10]. However, the software for that algorithm is proprietary and not in the public domain, so the authors were unable to provide us with a copy for comparison purposes. Also, with regard to the evaluation of segmentation process, it is worth mentioning that currently there are no better metrics to evaluate the result of the segmentation other than to get it inspected visually by an orthodontist. (This is generally true for any mesh segmentation algorithm which tries to find logical parts in a 3D mesh model.)

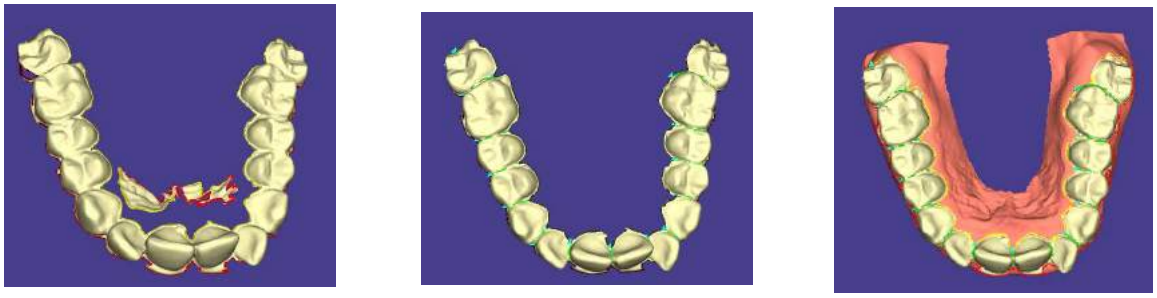
Table 1shows the process of segmenting three different models of increasing difficulty (model (a) being the easiest followed by (b) and then (c), in that order). The segmentation process is triggered by the “Segment Teeth” button after the user has set an appropriate curvature threshold while viewing the corresponding feature regions shown in Table 1 (row 1). Following this the system executes each step without any manual intervention and produces the results shown in the images in Table 1(row 5). It is worthwhile observing that the entire approach is quite robust in handling the malocclusions that are common in orthodontic practice.Among these models, the one in Table 1(a) is the easiest with well-represented gumline which requires no gumline repair at all. The model in Table 1(b) does have some regions of poorly-formed gumline which requires gumline repair.

The model in Table 1(c) is a complex case with severe scanning errors and malocclusions. Observe that it is difficult to estimate even the archform of such a model correctly. Inspite of this, 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 provided within the interface.

		
		
		
		
		
(a)	(b)	(c)

Tab.1: Results of segmentation tool on three different models shown in columns (a), (b), and (c). Row 1: curvature thresholding. Row 2: gumline detection. Row 3: gumline repair. Row 4: finding interstitial points and creating separators. Row 5: separation into individual tooth objects.

A typical problem with the laser-scanned models from plaster casts is the noise in the upper jaw (Figure 8(a)). 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 [10, 17]. In contrast to previous approaches [16, 17] 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 8(a)). We have designed heuristics that handles these spurious components. One such technique, which we described in Section 2.1.1, uses a lower bound on the number of vertices in a valid tooth object and merges with the gums any component that has fewer vertices. However, sometimes these noisy 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 this step is shown in Figure 8(b-c).



(a) (b) (c)

Fig.8: Noisy high-curvature regions in an upper jaw: (a) some gum regions are erroneously labeled as teeth (shown as small white patches), (b) spurious teeth component merged with the gums, and (c) final output of gum-teeth separation.

5 CONCLUSIONS

Digital orthodontics is a rapidly growing field as orthodontists migrate towards computer-basedsystems for treatment planning and simulation. Although this type of virtual simulation has the potential to produce very effective treatment plans, there is a gap between the number of orthodonticpatients who get their digital models made as part of the treatment process and the numberof patients who are actually treated using virtual plans. This is mainly because all currentlyavailable tools still require time-consuming manual interaction by the orthodontist.Our goal here has been to attempt to bridge this gap, by addressing the key problem of automatic segmentationof teeth in a 3D dental model. Such separation of teeth in a 3D model into individual toothobjects is the starting point for any digital approach to treatment planning.

The approach to tooth segmentation presented here is observed to work well onreal-world dental models, obtained from actual patients, and considerably reduces the humanintervention needed on a majority of the models. The previous approaches to this problem, whichmainly used an approach based on the morphological operators along with feature regions, have certain inherent limitations when applied to laser-scanned dental models in orthodontics. Thisis due to flat gumlines and scanning errors, which in turn are due to malocclusions in thepatient's jaws. Our solution overcomes these limitations by first dividing the problem intotwo subproblems (Gum-Teeth Separation and Tooth-Tooth Separation) and then solving thesesequentially.

6 ACKNOWLEDGMENT

We thank the three reviewers for useful comments that helped improve the paper.

7 REFERENCES

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

[2] Attene, M.; Katz, S.; Mortara, M.; Patane, G.; Spagnuolo, M.; Tal, A.: Mesh segmentation - A comparative study, SMI '06: Proceedings of the IEEE International Conference on Shape Modeling and Applications, 2006, 7.

[3] American Board of Orthodontics. http://tinyurl.com/n8bbwl. 2009.

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

[5] Gatzke, T.; Grimm, C. M.: Estimating curvature on triangular meshes, International Journal of Shape Modeling, 12(1), 2006, 1-28.

[6] GeoDigm Corporation, http://www.geodigmcorp.com, *emodel* Software.

[7] Hoffman, D. D.; Singh, M.: Salience of visual parts, Cognition, 63(1), 1997, 29-78.

[8] Keim, R. G.; Gottlieb, E. L.; Nelson, A. H.; Vogels III, D. S.: 2008 JCO study of orthodontic diagnosis and treatment procedures, Part 1: Results and trends, Journal of Clinical Orthodontics, 42(11), 2008, 625-640.

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

[10] Kronfeld, T.; Brunner, D.; Brunnett, G.: Snake-based segmentation of teeth from virtual dental casts, Computer-Aided Design & Applications, 7(2), 2010, 221-233.

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

[12] Meyer, M.; Desbrun, M.; Schröder, P.; Barr, A.: Discrete differential geometry operators for triangulated 2-manifolds, VisMath., 2002.

[13] Molkentin, D.: The Book of Qt 4: The art of building Qt applications, No Starch Press, San Francisco, CA, USA, 2007.

[14] Rössl, C.; Kobbelt, L.; Seidel, H.; Stadtwald, I.:Extraction of feature lines on triangulated surfaces using morphological operators, AAAI Symposium on Smart Graphics, 2000, 71-75.

[15] Sarfraz, M.; Masood, A.; Asim, M. R.: A new approach to corner detection, Computer Vision and Graphics, 2006, 528-533.

[16] Tian-ran, Y; Ning, D.; Guo-dong, H.; Xiao-sheng, C.; Hai-hua, C.; Wen-he, L.; Qing, Y.; Peijun, L.: Bio-information based segmentation of 3D dental models, International Conference on Bioinformatics and Biomedical Engineering, 2008, 624-627.

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