From CT to NURBS: Contour Fitting with B-spline Curves

Olya Grove, Khairan Rajab, Les. A. Piegl and Susana Lai-Yuen

University of South Florida, {olyagrove,khairan,lpiegl}@gmail.com, laiyuen@eng.usf.edu

ABSTRACT

This paper deals with the challenging task of computing accurate contours from CT and MRI scans using B-spline curve approximation. To date bio-modeling and visualization have been performed primarily on voxel and facet (triangle) based models. On the other hand, traditional CAD has reached a level of sophistication where just about any object can be designed, prototyped and manufactured using well-refined CAD modeling and manufacturing tools. NURBS, the *de facto* standard to represent geometry in CAD systems, have been the building blocks of CAD modeling and will be used in this paper to perform a critical task for bio-fabrication of human components. Although at first glance it may seem that contour fitting is a trivial task, the details presented in this paper reveal that traditional techniques are not readily adaptable to medical data and several fundamental algorithms had to be completely rewritten to account for the characteristics of human organs. The emphasis of this research is placed on accuracy and shape fidelity (precise surgical operation), speed (real-time simulation), smoothness (the discrete volumetric or faceted data should be replaced by smooth curves), and data reduction (the large amount of image data must be reduced by at least 60-80%).

Keywords: computed tomography, image processing, bio-CAD, NURBS curves.
DOI: 10.3722/cadaps.2011.3-21

1 INTRODUCTION

Medical diagnosis has entered a new dimension with the introduction of computed tomography (CT) and magnetic resonance imaging (MRI) into modern medicine. Images, representing slices of the human body, allow the physician to examine parts of the body and plan surgery or other forms of treatment. On the other hand, the advent of computers in design engineering opened yet another horizon resulting in sophisticated CAD systems. These systems are able to automate the entire manufacturing process from early conceptual design, through the various phases of the manufacturing processes, to archival and documentation. It comes as no surprise that the medical community has been eyeballing engineering automation with the intention of attempting to use the myriad of techniques to model, visualize and perhaps re-engineer parts (or all) of the human body. Bio-CAD [26,27] is an emerging field that supports the medical community with techniques of tissue engineering, part modeling and visualization, surgical simulation and planning, as well as manufacturing of various accessories and prosthetics.

This paper deals with the following challenging problem. Given a CT slice of an organ, generate a NURBS curve that accurately approximates an organ boundary in the image. This curve is an intermediate step in the direction of turning the set of CT slices into a volumetric NURBS model. At first glance the problem appears to be easy to solve, given the large number of image processing and curve fitting techniques readily available. However, a closer look reveals that techniques that served the CAD community well for decades simply fall apart when applied to medical data. The processed CT data is jaggy, has a lot of noise and the shape it represents is full of intricate details not present in traditional CAD data that defines car panels or airplane wings. The method presented herein requires a number of different techniques both from image processing as well as CAD. The crux of the problem is proper data preparation as no approximation method will ever produce satisfactory results if the input data is overly noisy or contains incorrectly ordered voxels (pixels). The sections below give the necessary details to reproduce the algorithm that takes CT data and turns it into a smooth NURBS automatically.

2 MEDICAL IMAGE DATA PREPROCESSING

The development of image based 3D computational models that represent human anatomy and physiology have been a very active area of research [31]. Capable of depicting both the entire body and select organs accurately, these models have an array of applications such as dosimetry calculations in nuclear medicine, visualization and surgery planning, prosthesis design and manufacturing, and so on.

Image based models are created from high-resolution cross-sectional digital images using imaging modalities such as magnetic resonance imagining (MRI) and computed tomography (CT). Once the medical data set is acquired, it is segmented in order to identify and classify various types of organs and tissues, and their boundaries.

Pixel data from a sequence segmented medical images can be stacked up and represented by volume arrays, in order to create a 3D voxel model. Voxel models, currently considered the most faithful representation of human anatomy, are excellent for visualization and are suitable for rapid prototyping. Voxel models, however, suffer from a number of disadvantages such as huge storage requirements and aliasing. Because voxel is the smallest unit, tissues and structures less than pixel in size cannot be faithfully represented [3]. Most importantly, voxel models lack geometric representation and as a result, cannot be used in engineering design.

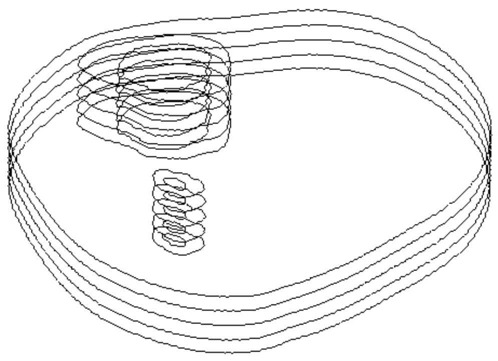


Fig. 1: B-Spline curves fitted to contours extracted from CT scan stack.

An alternative method to construct a 3D model from medical image data is to reconstruct a surface from serial parallel contours extracted from images (Fig.1). Data points representing the contours of the region interest are extracted and fitted with B-Spline curves, and the sequence of curves is lofted to generate the 3D surface model. The accuracy of the fitted surface depends largely on the quality of the extracted contours and the B-Splines fitted to the contour data. For example, in Figure 1, boundaries of tibia and fibula and the outer boundary of the leg are extracted and fitted with B-Spline curves using the proposed method, for 5 consequent CT scans.

NURBS models offer a myriad of advantages over voxel models. Resolution independent, these models can be easily resized and converted to voxel models. NURBS models are compact, concise and mathematically rigorous. Having been used extensively in engineering design, NURBS models enjoy a wide range of interrogation tools and lend themselves nicely for simulation and deformation analysis and manufacturing, which makes them extremely useful in biomedical research.

2.1 Contour Detection

In order to successfully reconstruct a 3D CAD model from a medical imaging dataset, it is important to accurately extract features of interest from the images. Segmenting medical datasets is a challenging task due to the size of the dataset, the complexity of anatomical shapes, sampling artifacts, and noise, which results in disconnected and inaccurate representations of the organ contours [13].

Although medical image segmentation is a relatively established and mature research field, fully automatic segmentation of medical data remains an unsolved problem [3]. Due to inherent noise in the data, complexity of anatomical shapes, overlap of gray level values between neighboring tissues and organs and lack of distinct boundaries, segmentation is often performed manually. Manual segmentation is time consuming and requires a trained experienced radiologist to correctly identify and outline each tissue. Even among experienced radiologists, there is a lack of agreement with respect to segmentation [31], with discrepancies as large as 24% for difficult to segment organs such as esophagus [3]

Currently, semi automatic segmentation is most commonly employed – these techniques rely on a limited manual input from the user, thus allowing processing a large number of contiguous images needed for generating a 3D model at a reasonable time and with an accepted degree of accuracy.

Semi-automatic segmentation relies on either image thresholding or edge detection algorithms and involves a certain amount of manual pre- and post-processing. Image segmentation and edge extraction for further 3D model construction is often done using a commercial software package (e.g. MIMICS), which involves manually thresholding and editing each slice individually, as the result of automatic edge detection often provides a poor result (Fig. 2).

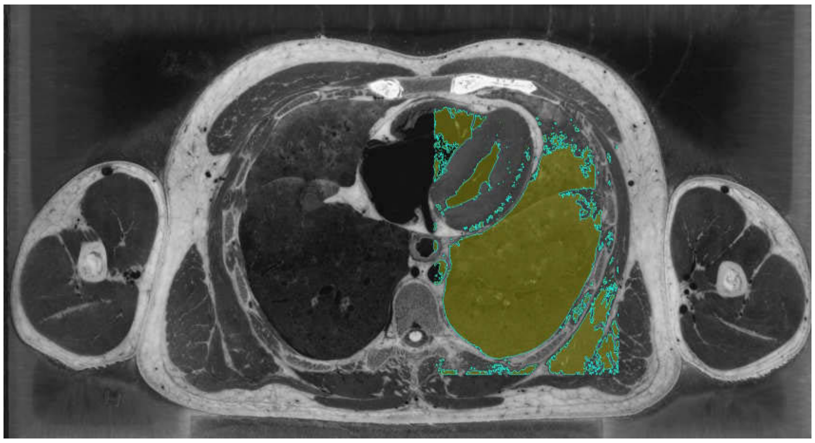


Fig. 2: Automatic edge detection using MIMICS.

2.2 Contour Tracing

In this research we used Canny edge detector [1,2] to find the contour of an image of interest (Fig. 3). Edge-based segmentation [4,25,29,33] relies on edges discovered by an edge detecting operator to segment regions of interest in the image. Edges in an image usually manifest locations of discontinuities in intensities, texture, etc.

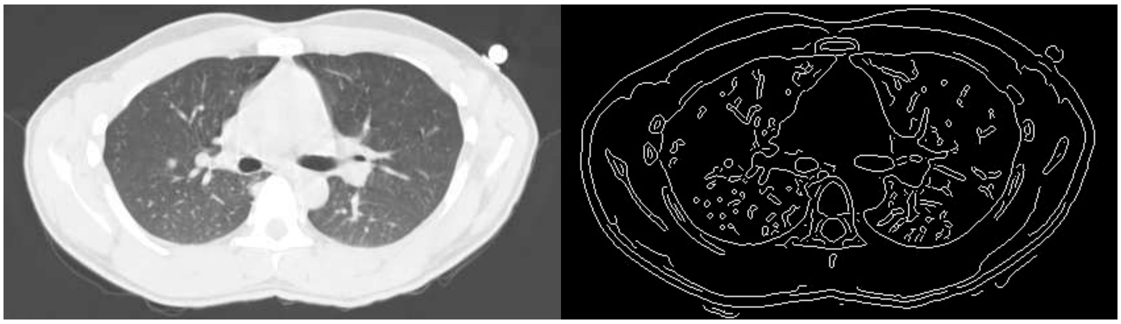


Fig. 3: MRI scan of the lung (a), binary image after applying Canny edge detector (b).

After Canny edge detector is applied to the original image, this resultant binary image forms an edge map (Fig. 3b) used as input to the contour-tracing algorithm [11,18,21,24,29].

The algorithm is based on depth-first-search (DFS). DFS is often used in image traversing, and has been applied to segmentation and path finding in medical images [5,23]. While DFS produces a spanning tree of all the vertices reached during a graph search, in our application the goal is to generate an orderly sequence of a closed boundary pixel positions. Just as in DFS, three categories are used to describe the vertices: *undiscovered*, *discovered*, and *visited*.

An edge map of the original image is used as input to the contour tracing algorithm. The algorithm creates a table of the same size as the input image and sets all cells to -1, to describe all pixels in the corresponding input image as *undiscovered*. Once an edge pixel is located (white pixel in a binary image, where 0/black denotes background and 255/white denotes an edge), it is marked as *discovered* and *current* and the corresponding table cell is changed to 0. The algorithm then proceeds to discover the neighbors of the *discovered* pixel, by recursively applying the algorithm. Eight neighbors are examined in a clockwise fashion: if only one neighbor is discovered, its status is changed from *undiscovered* to *discovered*. Once all neighbors are traversed, current pixel x and y coordinates are added to the array of ordered boundary pixels. If two or more neighbors are discovered, the algorithm further investigates each member to identify dangling pixels and chooses one of the neighbors to proceed.

Although Canny edge detection algorithm produces ridges that are one pixel wide due to the non-maximal suppression step, discontinuities are still possible and can negatively affect the ordering of the pixels needed for the fitting [9,32]. Therefore, while Canny edge detection algorithm finds edges automatically, minimal post-processing might be required in order to get rid of remaining few dangling pixels or fix discontinuities.

2.3 Contour Smoothing

After the contour has been traced and cleaned we are ready to fit a NURBS curve to the pixel data. While it seems a matter of selecting one of the many fitting capabilities from existing libraries, several fitting attempts that have failed to produce acceptable results revealed that traditional CAD techniques are not appropriate to deal with image data, at least not without significant modifications.

In the traditional design engineering practice one would fit a curve to a sizable data set in predominantly two ways [19]:

- Start with an interpolation, i.e. the curve passes through all the data points, and then remove as many knots (control points) as allowed by the tolerance. This method did not work because the interpolating curve had too many wiggles that were not removable even with a high tolerance, see Figure 4.
- Start with an approximation with some number of control points and increase the degree of freedom (knots and control points) until the tolerance requirement is met. This method did not work either because:
  - it was nearly impossible to guess the number of start degrees of freedom to get a reasonably good initial approximation;
  - intricate details could not be reproduced even if the number of control points were increased drastically;
  - as the number of control points were increased, the curve approached an interpolating status and the unwanted wiggles started to appear; and
  - the knot vector (see below), that is set based on the distribution of the points turned out to be inadequate.

It became evident that the only way a CAD-based technique would work is if the data is smoothed out and decomposed into segments of similar complexity/shape. While the literature offers a few data smoothing methods [8,30], they were not adequate to be applied in a CAD-based application.

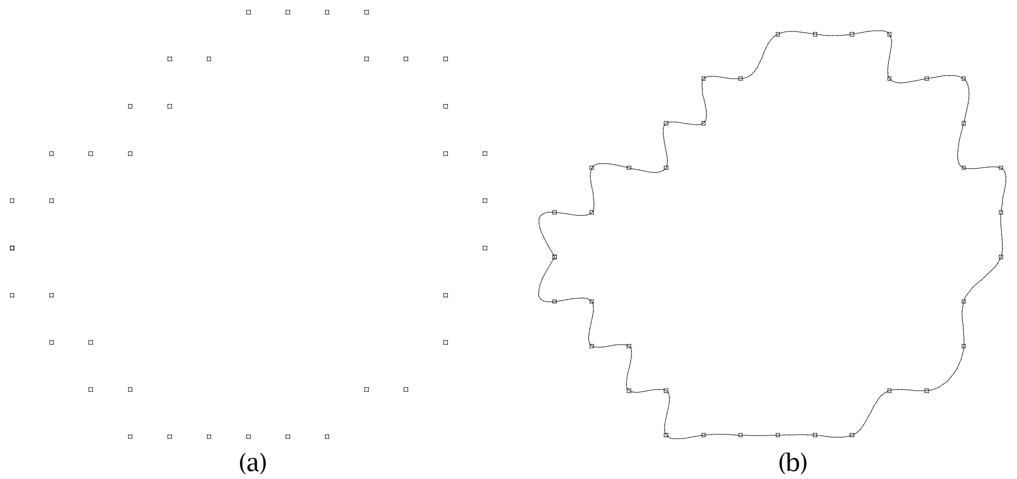


Fig. 4: Jaggy data set (a), interpolation (b).

We need a smoothing method that takes pixel data, i.e. the jaggy point set, and turns this data into a smooth point set. That is, the method must be able to alter the data set so that the interpolating curve becomes as smooth as the user wants it to be. Our method is based on a variation of Laplacian smoothing applied to a polygon. Given an ordered point set $Q_0,...Q_m$, and the level of required smoothing lev, the algorithm below generates a new, smoother, set of points.

```
Procedure Smooth(Q[0..m], lev)
begin
    for $k = 1$ to $lev$
      for $i = 1$ to $m - 1$
        $Q_i \leftarrow \frac{1}{4}Q_{i-1} + \frac{1}{2}Q_i + \frac{1}{4}Q_{i+1}$
      endfor
      if ( closed )
          $Q_0 \leftarrow \frac{1}{4}Q_{m-1} + \frac{1}{2}Q_m + \frac{1}{4}Q_1$
          $Q_m \leftarrow Q_0$
      endif
    endfor
end
```

That is, the algorithm recursively computes a new position for each point until the required level is reached. Several examples are shown in Figures 5-6 where the curve shown is the interpolating curve to the smoothed data points. It is evident that even one level of smoothing improves the data considerably. It is also interesting to note that as the level of smoothing increases, the new data set converges to a point (for closed data sets).

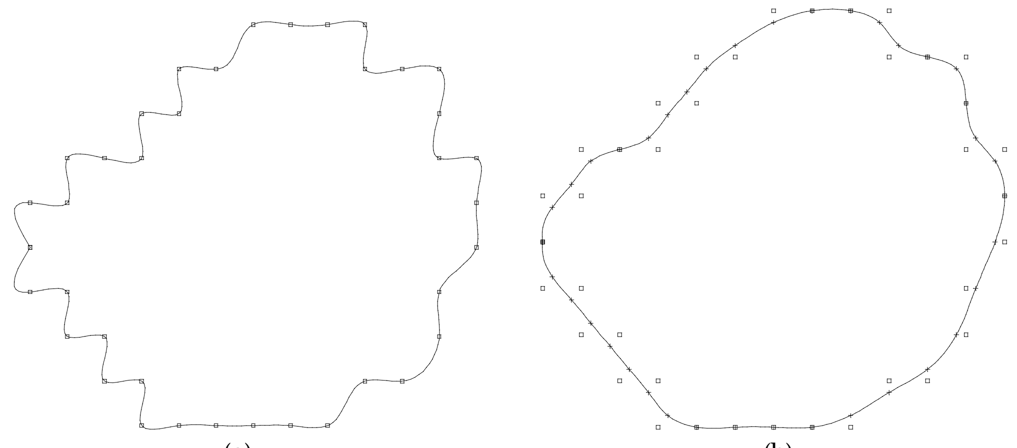


Fig. 5: Level of smoothing zero (a) and one (b).

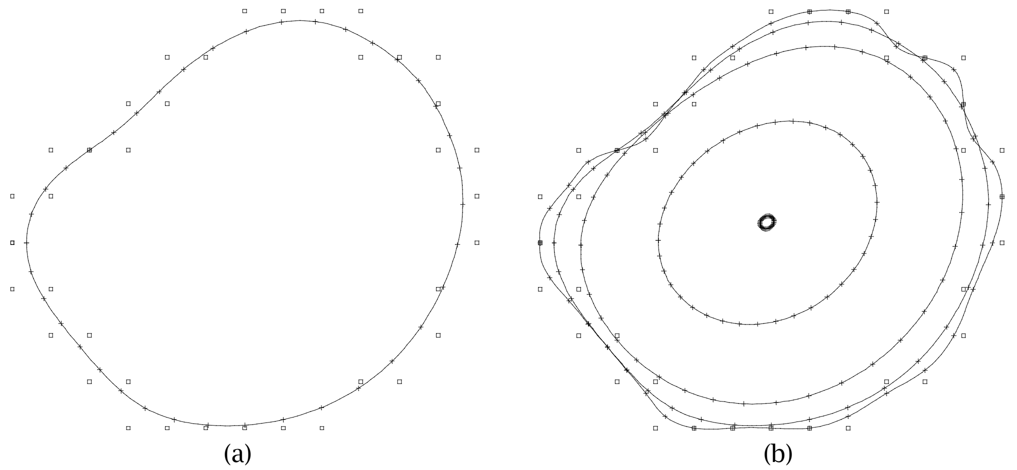


Fig. 6: Level of smoothing five (a), convergence with levels 1, 5, 20, 100 and 500 (b).

An important question remains to be answered: when to stop smoothing, i.e. when is the data set smooth enough? There are two types of answers to this question: a technical and a practical one. The technical answer goes like this. For each smoothed point take a small neighborhood, e.g. 3-5 points on either side. Fit a curve, e.g. a circle, to this small data set and check the error. If the error is within a certain deviation, stop smoothing. While this is technically correct, it has two disadvantages: (1) computationally expensive, and (2) it opens up new questions as to what is the right neighborhood and the acceptable deviation.

We opted to go with the practical approach: use a fixed level of smoothing based on the type of the data set. Medical data is fairly predictable within the organ type it represents so it is fairly easy to set up a knowledge base that gives the user the proper level of smoothing for successful curve fitting. In our examples all data was smoothed by 3-5 levels of smoothing.

3 POINT DATA SEGMENTATION FOR CURVE FITTING

The smoothed data set obtained in the previous step is now used to assist in finding an optimal NURBS curve that approximates the data set to within a user supplied tolerance and has relatively few number of control points. Before we can proceed, some B-spline fundamentals will be needed. A B-spline curve of degree p is defined as follows [19]:

$$C(u) = \sum_{i=0}^{n} N_{i,p}(u) P_i$$

The points P_i represent control points and $N_{i,p}(u)$ are the normalized B-spline functions defined over the knot vector

$$U = \{\underbrace{u_0 = \cdots = u_p}_{p+1}, u_{p+1}, \ldots, u_n, \underbrace{u_{m-p} = \cdots = u_m}_{p+1}\}$$

Unless otherwise stated, the knot vector is always clamped, i.e. the first and the last knots are repeated with multiplicity $p+1$. Now, given the data points $Q_0, \ldots Q_m$, we are seeking a B-spline curve that approximates the data in the least-square sense. This requires the computation of the parameters $t_0, \ldots, t_m$ where the points are assumed, the choice of the degree p, the highest index of control points n, and the knot vector U. The parameters are computed using the chord-length method [19], the degree will be chosen to be two or three, and the highest index and the knot vector will be computed in the section that follows. This leaves only the control points to be determined that can be done by solving the following system of linear equations:

$$\left(N^T N\right) P = R$$

$$N = \begin{bmatrix} N_{0,p}(t_0) & N_{1,p}(t_0) & \cdots & N_{n,p}(t_0) \\ N_{0,p}(t_1) & N_{1,p}(t_1) & \cdots & N_{n,p}(t_1) \\ \vdots & \vdots & \cdots & \vdots \\ N_{0,p}(t_m) & N_{1,p}(t_m) & \cdots & N_{n,p}(t_m) \end{bmatrix} \quad R = \begin{bmatrix} N_{0,p}(t_0)Q_0 + N_{0,p}(t_1)Q_1 + \cdots + N_{0,p}(t_m)Q_m \\ N_{1,p}(t_0)Q_0 + N_{1,p}(t_1)Q_1 + \cdots + N_{1,p}(t_m)Q_m \\ \vdots \\ N_{n,p}(t_0)Q_0 + N_{n,p}(t_1)Q_1 + \cdots + N_{n,p}(t_m)Q_m \end{bmatrix}$$

Choosing the right knots (see below), the matrix N becomes diagonally dominant with semi-bandwidth less than $p+1$, and the system of equation can be solved without pivoting [19].

Each B-spline curve can be thought of as the collection of Bezier segments that are automatically joined with C^{p-1} continuity, assuming that there are no multiple interior knots. This B-spline-to-Bezier relationship is what our method takes advantage of to determine the appropriate degrees of freedom (number of control points) and the right knot locations. That is, the smoothed data set will be segmented using a set of Bezier curves so that on each segment the error is less than the given tolerance.

A Bezier curve of degree p is completely determined by its control points as follows [19]:

$$B(u) = \sum_{i=0}^{p} B_{i,p}(u) P_i \qquad B_{i,p}(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i}$$

Using these curves least-squares Bezier fitting is applied to obtain a set of C^0 Bezier curves that approximate the data to within a tolerance. The Bezier least-squares fitting is a special case of B-spline fitting and the system to be solved simplifies to:

$$\left(B^T B\right) P = R$$

$$B = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ B_{0,p}(t_1) & B_{1,p}(t_1) & \cdots & B_{p,p}(t_1) \\ \vdots & \vdots & \cdots & \vdots \\ B_{0,p}(t_{m-1}) & B_{1,p}(t_{m-1}) & \cdots & B_{p,p}(t_{m-1}) \\ 0 & 0 & \cdots & 1 \end{bmatrix} \qquad R = \begin{bmatrix} Q_0 \\ B_{1,p}(t_0)Q_0 + B_{1,p}(t_1)Q_1 + \cdots + B_{1,p}(t_m)Q_m \\ \vdots \\ B_{p-1,p}(t_0)Q_0 + B_{p-1,p}(t_1)Q_1 + \cdots + B_{p-1,p}(t_m)Q_m \\ Q_m \end{bmatrix}$$

where the parameters $t_0,\ldots,t_m$ are computed using the chord-length method [19]. Now, given the Bezier fitting technique with a given degree, the point set is segmented using a combination of Bezier fitting and binary search type index splitting.

```
Input:
    Q[0], …,Q[m]: data points
    t[0], …, t[m]: parameters data points are assumed at
    required tolerance of the approximation
Output:
    parameters/data points where the junction points are assumed
Algorithm:
    start = 0; end = m;
    while end > start do
        left = start; right = end;
        do
            n = end-start;
            fit a Bezier curve to the points Q[start],…,Q[n];
            success ← check the parametric error;
            if success = yes then right = end;    else left = end;
            end = (left + right)/2;
        while left ≠ end
        start = end; end = m;
        save the parameter/index of the end point for the output parameter list
    endwhile
```

That is, the algorithm takes $Q_0,...Q_m$ and $t_0,...,t_m$ as input, uses the longest Bezier arc to locally approximate a subset of these points, and returns the index set $j_1,...,j_k$ so that $Q_{j_1},...,Q_{j_k}$ are the junction points of the piecewise Bezier approximation. Several examples are shown in Figures 7-8.

Note how the local Bezier curves capture the shape characteristics of the each subset. i.e. each subset is simple enough to approximate the points by a single Bezier arc. This in turns indicates the degrees of freedom required to pass a NURBS curve to the data, as well as indicates where the knots should be placed to faithfully reproduce local characteristics.

4 B-SPLINE CURVE FITTING

B-spline curves can be fitted to the smoothed data or to the original data [6,7,10,14-16,20,22,34,35]. In many CAD applications curve interpolation followed by knot removal is an effective technique to approximate a large set of smooth point set (right side of the flow chart in Figure 1). Unfortunately, this technique does not work with noisy data so it is replaced with its smoothed version, which may introduce error. This leaves only one option: approximate the original data given the segmentation result of the smoothed data. That is, given $t_0,...,t_m$, the segmentation indexes $j_0,...,j_k$ (note that there are k Bezier segments), the highest index of control points will be set as $n = k * p$, where p is the degree of the curve. This is the minimum degree of freedom necessary to approximate the curve to within the given tolerance (as per the Bezier segmentation).

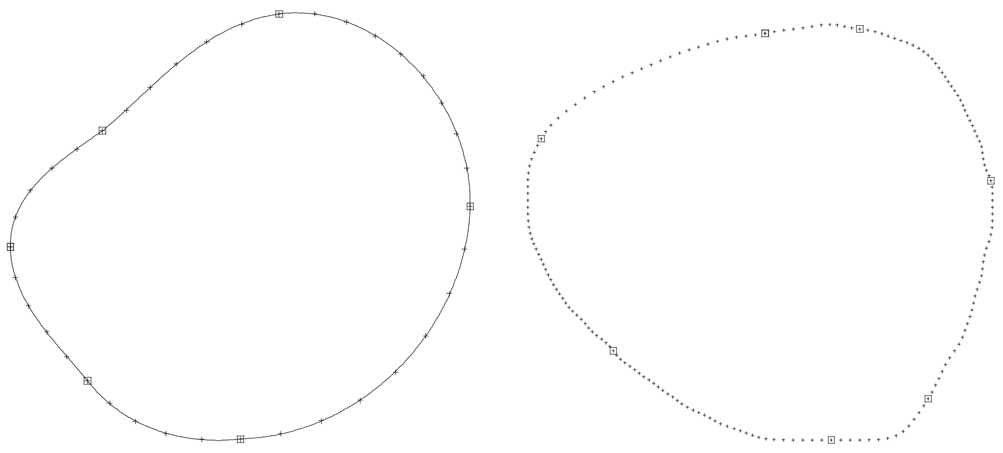


Fig. 7: Piecewise C^0 Bezier approximation: tol = 0.0008 (a), segmenting points: tol = 0.002 (b).

The knot vector is computed in two steps. First, a new set of parameters $s_0,...,s_n$ are computed from the original parameters and the junction indexes $j_0,...,j_k$ as follows. For each $\left[t_{j_l}, t_{j_{l+1}}\right]$ compute

$$\begin{aligned}&s_r = t_{j_l}\\&s_a \leftarrow (p-1) \text{ average of } t_{j_l}, t_{j_l+1},...,t_{j_{l+1}-1}\\&s_{r+p} = t_{j_{l+1}}\end{aligned}$$

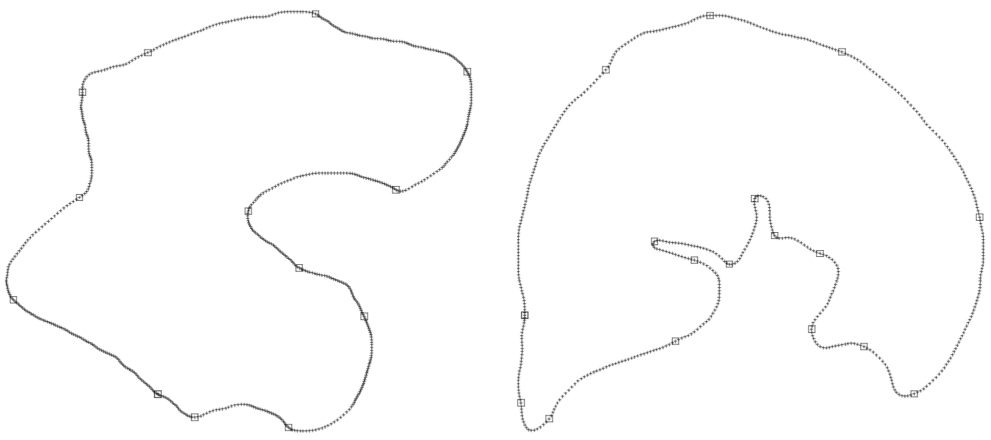


Fig. 8: Segmentation examples: level = 5, degree = 3, tolerance = 0.002.

where the k^{th} average of a set is defined as follow: cluster the set into k subsets and compute the average of each subset. If $k=1$, then this produces the usual average of a set. If $k=2$, the set is divided into two (equal) subsets and their averages are computed resulting in two averages. If $k=3$, one gets 3 averages, etc. The formula above computes exactly $p-1$ averages per segment resulting in $n+1$ new parameters $s_0,...,s_n$. These new parameters are then used to compute the knots based on the well-known formula of De Boor [19]:

$$u_0 = \cdots = u_p = 0$$
$$u_{j+p} = \frac{1}{p}\sum_{i=j}^{j+p-1} s_i \qquad j = 1,...,n-p$$
$$u_{n+1} = \cdots = u_{n+p+1} = 1$$

Now, we have the highest index n, the parameters $t_0,...,t_m$ and the knots $u_0,...,u_{n+p+1}$, a B-spline curve approximation is computed as formulated above, i.e. the matrix equation is solved for the missing control points. This approximation capability can be extended in a number of ways:

- Approximation with end derivatives specified [20].
- Approximation to closed data set [20].
- Approximation with floating end conditions, i.e. the end points of the curve do not coincide with the end points of the data set.
- Approximation with end point constraint, i.e. the end points of the curve coincide with the end points of the data set Q_0 and Q_m.

Space limitation would not allow us to provide details of these variations. If the curve is closed, a smooth closure is required, if the curve is open then either floating or constraint end condition can be applied.

The next important step after the initial fitting is to check the error. Ideally this is done by projecting all points to the curve and computing the perpendicular distances [35]. Point projection is expensive (requires first and second derivatives) and is error prone and hence has not been implemented. A simpler method is to check the parametric error, i.e.

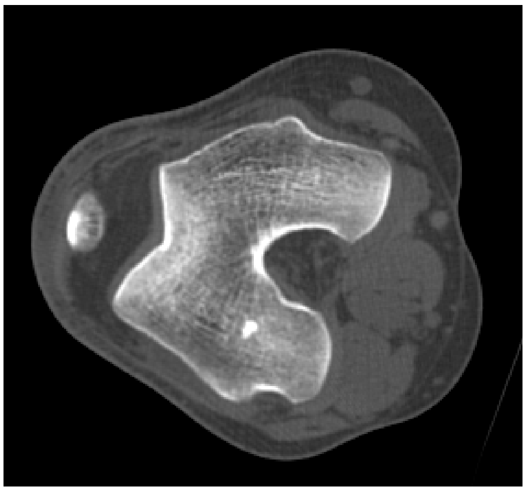
$$\Delta = \max_i \left| Q_i - C(t_i) \right| \quad Q_i \text{ is assumed at } t_i$$

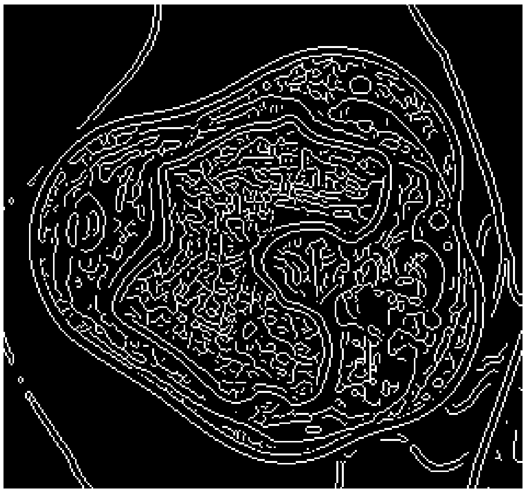
If the error test is passed, the approximation has been completed. Otherwise, at certain areas the curve has not faithfully reproduced the shape and more degrees of freedom are needed to comply with the accuracy requirement. There are two options at this point: (1) introduce additional degrees of freedom locally, i.e. add knots to the segments that were not approximated well enough [16], and (2) do a global decomposition with a smaller tolerance. We elected to implement the second method, i.e. compute a new segmentation with a smaller tolerance. The logic behind this is that the piecewise C^0 Bezier segments capture the shape information quite well, however, going from the C^0 Bezier to the C^{p-1} NURBS may require more freedom given the continuity of this curve. Also, the piecewise Bezier curve provides a shape dependent decomposition and a distribution of parameters that reflects the intricate details inherent in the data set. Practical experience shows that half to quarter of the given tolerance always produces a smooth B-spline curve that is within the required accuracy and captures all desired details.

The last step in the fitting process is data purging, i.e. eliminating all unnecessary knots (control points). This is done via knot removal whose details are found in [19].

5 EXAMPLES

The first example is shown in Figure 9. The boundary data is obtained from the bone CT shown in Figure 9(a). On the left (Fig. 9(c)) fitting was obtained after one level of smoothing, whereas the right side (Fig. 9(d)) shows a fit after five levels of smoothing. Both curves provide perfect coverage of the pixel domain (the left figure shows both the pixels and the curve superimposed). The data set has 638 points, level one smoothing resulted in a B-spline fit with 252 control points (60% data reduction), whereas level five produce only 146 control points (77% data reduction).


(a)


(b)

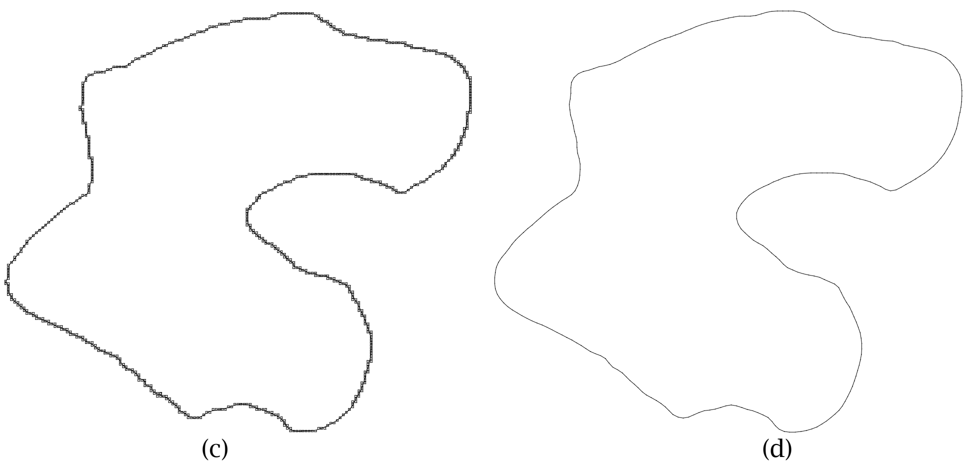
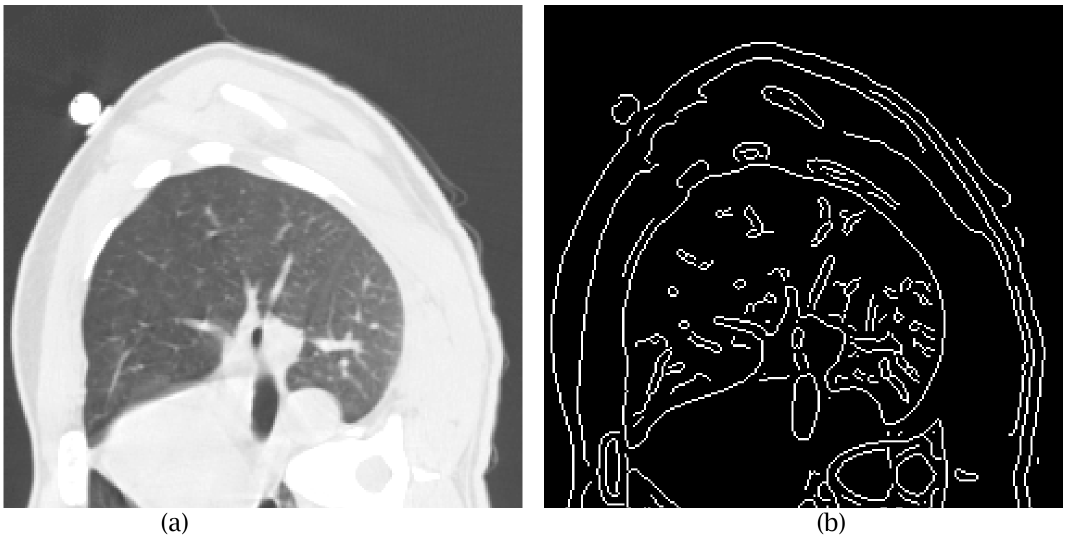
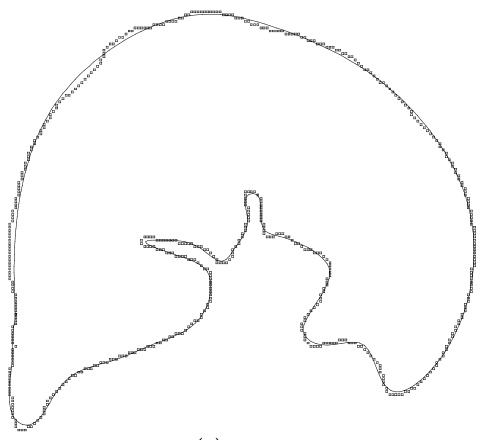


Fig. 9: Bone example: original image (a); edge detection (b); fitting with tolerance = 0.001 and level = 1, contour points and curve are shown (c); level = 5, curve only (d).

Figure 10 illustrates a fitting case with a large and a smaller tolerance used. The data was obtained from the lung CT shown in Figure 10(a). Note how the approximation misses the details due to the large tolerance (Fig. 10(c)). A tighter tolerance (an order of magnitude smaller) produced a perfect contour curve (Fig. 10(d)). The data contained 555 points, the 0.01 tolerance produced a B-spline fit with 39 control points (92% reduction, although the curve is not acceptable), whereas 0.001 tolerance increased the number of control points to 193 (65% data reduction). Note that in the engineering practice a data set of 555 points requires about 10% or less control points because the data is smooth and contains no intricate details. In the medical practice this kind of data reduction is hardly achievable given the noise of the data and the very complicated shape it represents.




(c)

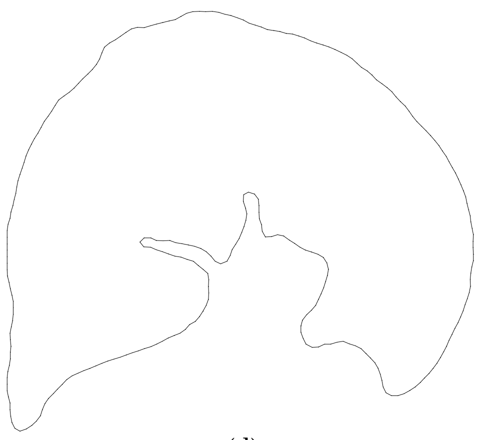
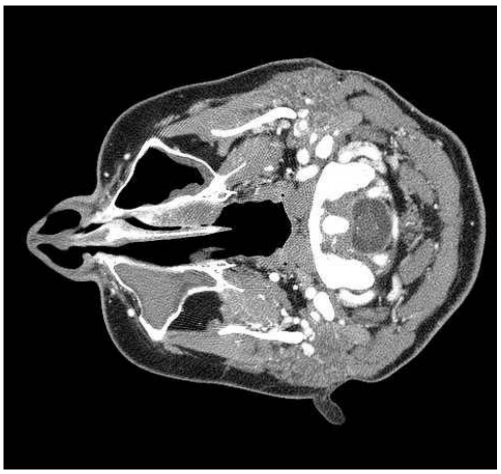
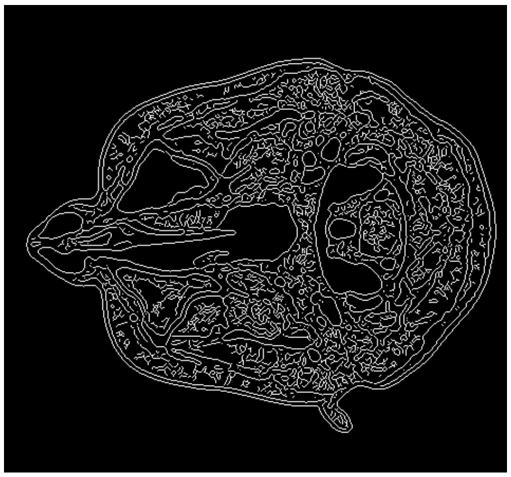

(d)

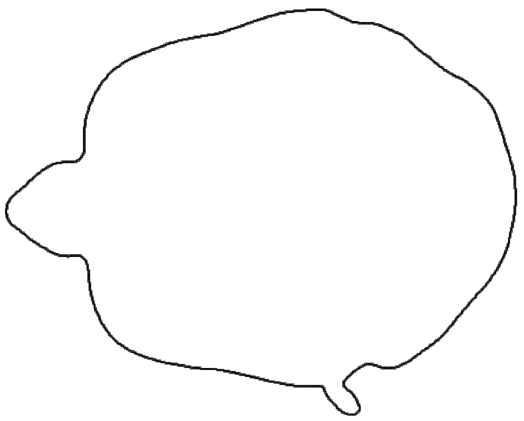
Fig. 10: Lung example: original image (a); edge detection (b); fitting with level = 1, tolerance = 0.01, contour points and curve are shown (c); tolerance = 0.001, curve only (d).

Figures 11-17 demonstrate a variety of applications of the method on data sets obtained from the brain, head, kidney and bones. All examples used $lev = 3$ smoothing, tolerances 0.01 or 0.001 and the degree $p = 3$, and m, n denote the highest indexes of data points and control points, respectively. The table below summarizes the data reduction (%) capabilities.

Fig	9	10	11	12	13	14
m	554	534	1159	1318	949	294
tol	$10^{-2}/10^{-3}$	10^{-3}	10^{-3}	10^{-3}	10^{-3}	$10^{-2}/10^{-3}$
lev	1	3	3	3	3	3
n	38/192	123	81	177	113	21/78
%	92/65	76	93	86	88	92/73


(a)


(b)


(c)

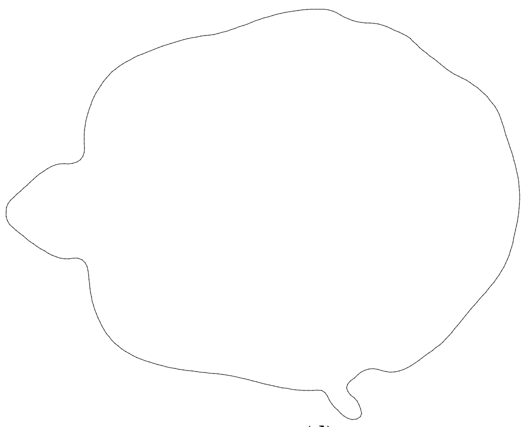
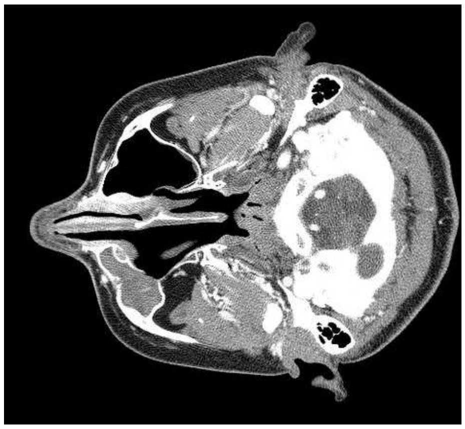

(d)

Fig. 11: Head example: original image (a); edge detection (b); fitting with level = 3, tolerance = 0.001, contour points and curve are shown (c); tolerance = 0.001, curve only (d).


(c)

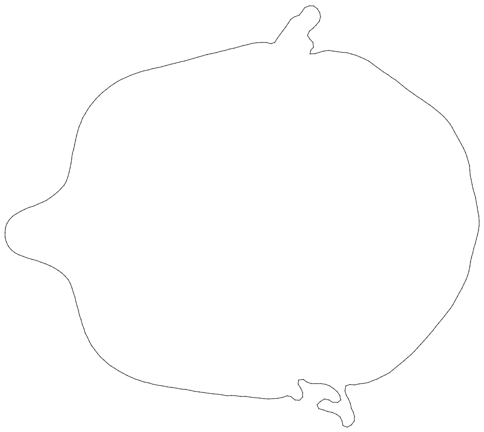
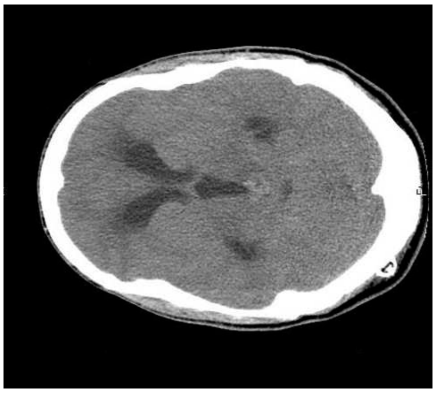
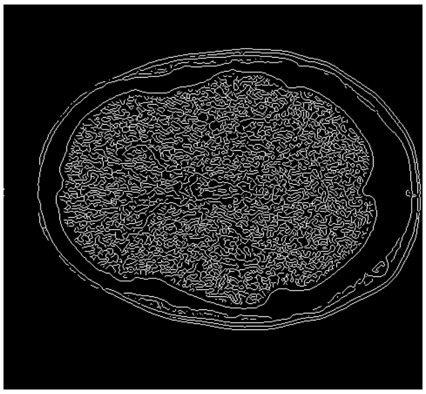
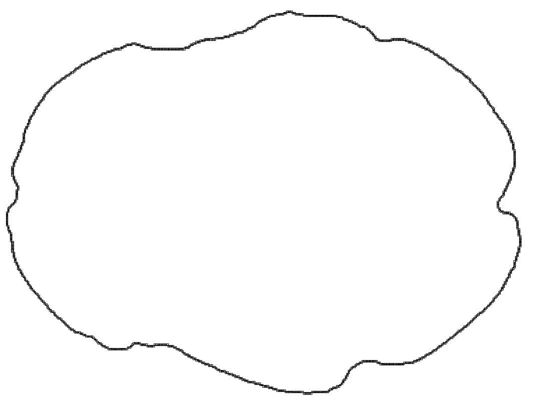

(d)

Fig. 12: Head example with a more complicated cut: original image (a); edge detection (b); fitting with level = 3, tolerance = 0.001, contour points and curve are shown (c); tolerance = 0.001, curve only (d).


(a)


(b)


(c)

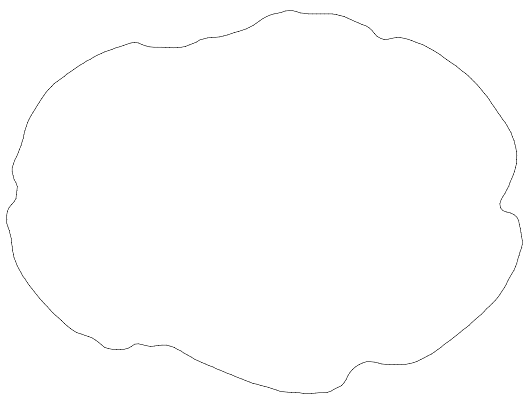
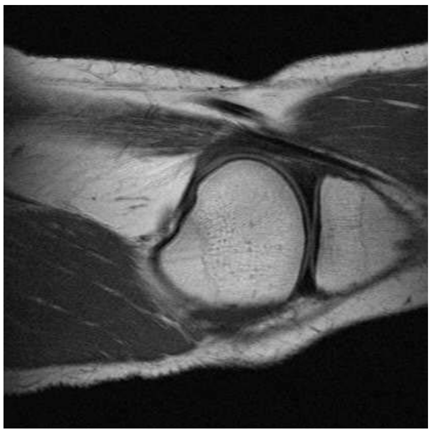
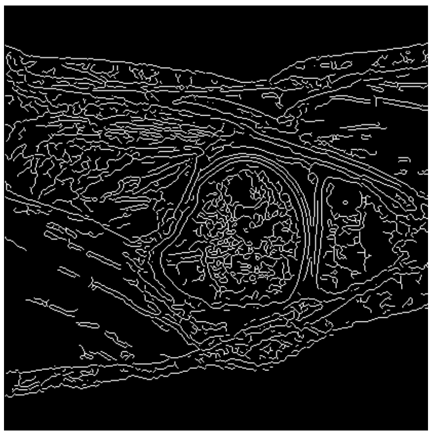
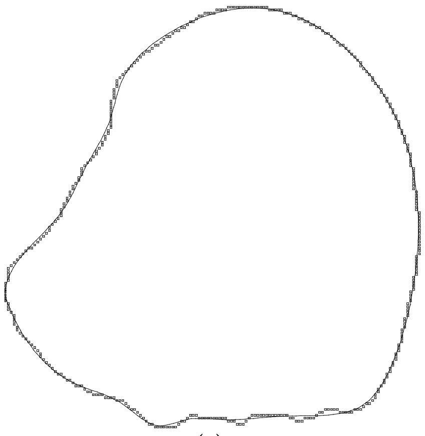
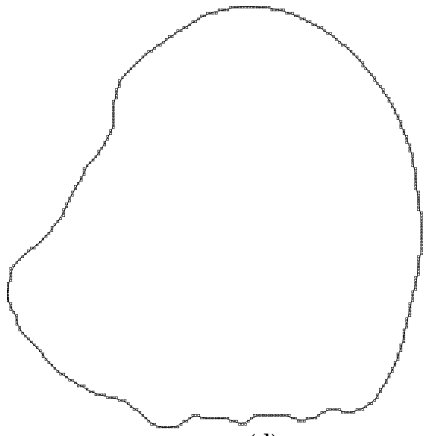

(d)

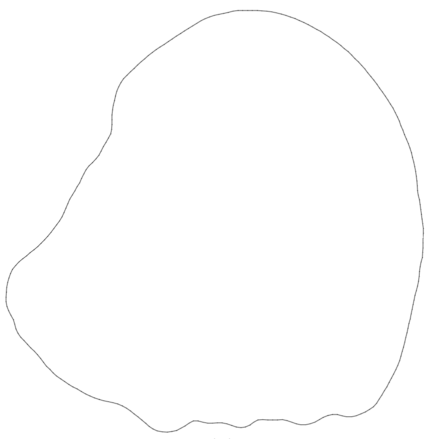
Fig. 13: Head example: original image (a); edge detection (b); fitting with level = 3, tolerance = 0.001, contour points and curve are shown (c); tolerance = 0.001, curve only (d).


(a)


(b)


(c)

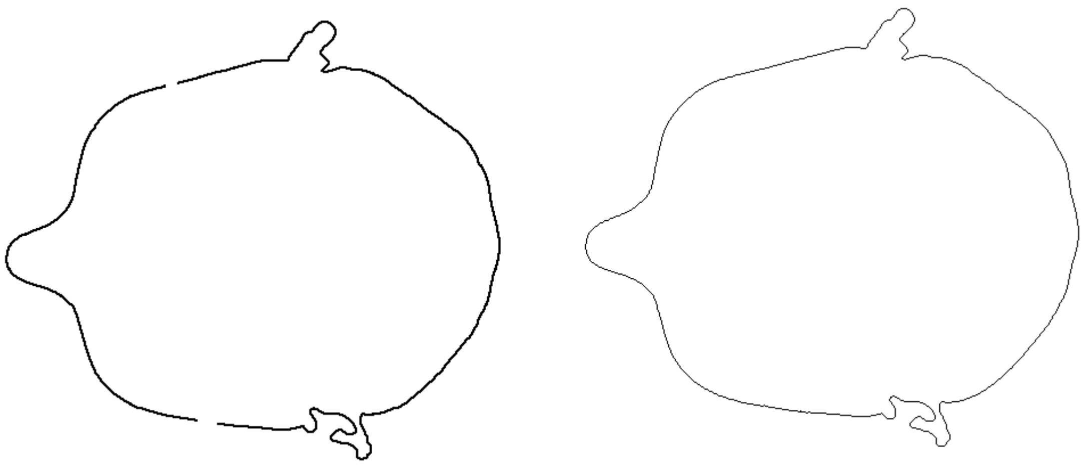

(d)



(e)

Fig. 14: Kneecap example: original image (a); edge detection (b); fitting with level = 3, tolerance = 0.01, contour points and curve are shown (c); tolerance = 0.001, points and curve (d); curve only (e).

Data sets with small discontinuities in data contours can be fitted with a closed B-spline curve, as illustrated in Figure 15.



(a) (b)

Fig. 15: Edge points with small discontinuities (a) and B-spline fitted to the dataset (b).

When the contrast between neighboring pixel intensity values is high, existing automatic contour detection and fitting methods perform rather well. However, in the presence of small intricate details abundant in medical datasets, these methods do not represent these shapes adequately, which can lead to incorrect overall presentation of anatomy as shown in Figure 16. Faithful capture of the overall shape is of uttermost importance in medical applications such as custom implant design and calculation of volume and surface area of organs for medical dosimetry [3].

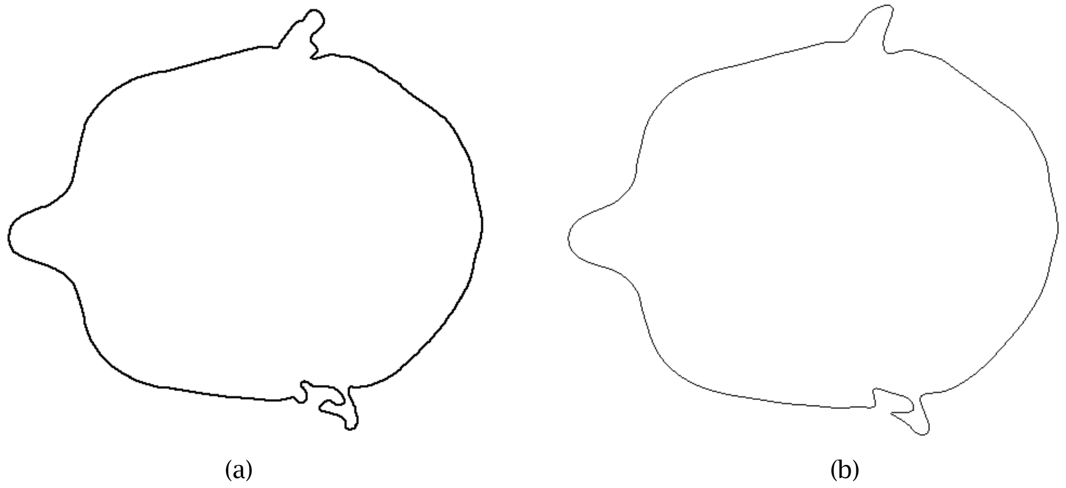


Fig. 16: Original dataset (a) approximated with a least-squares B-Spline curve (b).

In order to capture all the important detail, some methods attempt to interpolate the pixel data. Since pixel dataset is jaggy and very dense, the resultant curve not is smooth and required a high number of control points (Fig. 17). For example, for the dataset (Fig. 16a) with 1,319 data points, MIMICS fitting capability curve produced a kinked B-Spline curve with 2172 control points (Fig. 17b), while our proposed method fitted a smooth B-Spline curve with 156 control points (Fig. 12d). Such high number of control points is not acceptable in Bio-CAD modeling: anatomical cross-sections or CT axial images are taken, on average, at 1mm, which results in a large image dataset. Lofting through contours already containing a high number of control points will lead to exponential increase of overall control point count and subsequent numerical problems.

Figure 18(d) demonstrates a segment of the B-Spline contour, zoomed in to better illustrate the jaggedness of the fitted curve.

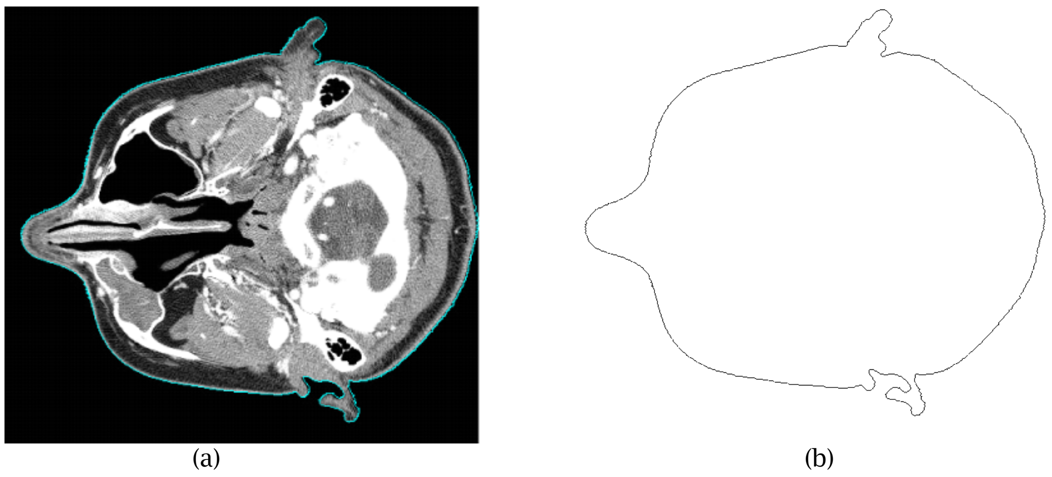
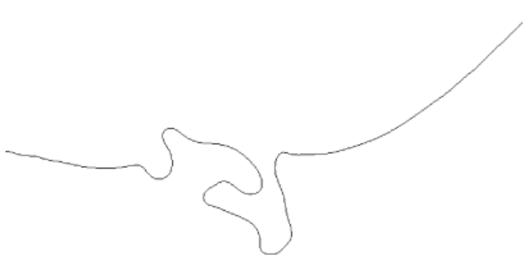
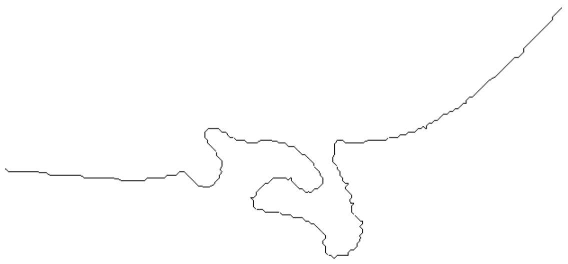


Fig. 17: Border detected (a) and fitted with a B-Spline (b) automatically using MIMICS.



(c)



(d)

Fig. 18: Segment of fitted contour: proposed method (a) and MIMICS fitting capability (b).

6 CONCLUSIONS

B-spline approximation to contour CT and MRI data was presented in this paper. The method introduced relies on B-spline technology that served the engineering design community for decades. It turns out that, due to the tremendous noise and the amount of image data, traditional techniques are inadequate to produce smooth and accurate contours to pixel data. The method presented requires several steps, a large array of B-spline capabilities, in-depth knowledge of NURBS-based computing and significant experience using various forms of fitting techniques. However, the resulting multi-stage technique has high shape fidelity, it is fast (real time even on an average Windows desktop) and is able to reduce the data size by about 60-80%.

7 ACKNOWLEDGEMENTS

We would like to thank Charles C. Nofsinger at the Department of Orthopedics, USF, for providing us with CT scan data used extensively in this project, and for assisting us throughout the project. The cross-sectional images from the Visible Human Project® were also used in our research and illustrated in the paper, provided courtesy of the National Library of Medicine (NLM). This research was supported by the National Science Foundation under grant No. DMI-0758231, awarded to the University of South Florida. All opinions, findings, conclusions and recommendations expressed in this paper are those of the author and do not necessarily reflect the National Science Foundation or the University of South Florida.

REFERENCES

[1] Aarnik, R. G.; De la Rosette, J. J. M. C. H.; Debruyne, F. M. J.; Wijkstra, H.: A preprocessing algorithm for edge detection with multiple scale resolutions, Proc. Engineering in Medicine and Biology Society, 2, 1996, 903-904.
[2] Canny, J. F.: A computational approach to edge detection, IEEE Trans Pattern Analysis and Machine Intelligence, 8, 1986, 679-698.
[3] Caon, M.: Voxel-based computational models of real human anatomy: a review, Radiation and Environmental Biophysics, 42(4), 2004, 229-235.
[4] Davis, L. S.: A survey of edge detection techniques, Computer Graphics and Image Processing, 4, 1975, 248-270.
[5] Gelman, S. A.; Tardif, T.; Zrimec, T.: A knowledge-based approach to 3-D reconstruction of human cerebral vasculature, Computers in Biology and Medicine, 28, 1998, 405-413.
[6] Goldenthal, R.; Bercovier, M.: Spline curve approximation and design by optimal control over the knots, Computing, 72, 2004, 53-64.
[7] Holzle, G. E.: Knot placement for piecewise polynomial approximation of curves, Computer-Aided Design, 15(5), 1983, 295-296.

[8] Hu, J.; Yu, D.; Yan, H.: A multiple point boundary smoothing algorithm, Pattern Recognition Letters, 19, 1998, 657-668.

[9] Koplowitz, J.; Plante, S.: Corner detection for chain coded curves, Pattern Recognition, 28, 1995, 843-852.

[10] Li, W.; Xu, S.; Zhao, G.; Goh, L. P.: Adaptive knot placement in B-spline curve approximation, Computer-Aided Design, 37, 2005, 791-797.

[11] Liow, Y. T.: Contour tracing algorithm that preserves common boundaries between regions, CVGIP Image Understanding, 53(3), 1991, 313-321.

[12] Marr, D.; Hilreth, E.: Theory of edge detection, Proc. Royal Society of London, B207, 1980, 187-217.

[13] McInerney, T.; Terzopoulos, D.: Deformable models in medical images analysis: a survey, Medical Image Analysis, 1, 1996, 91-108.

[14] Park, H.; Kim, K.; Lee, S. C.: A method for approximate NURBS curve compatibility based on multiple curve refitting, Computer-Aided Design, 32, 2000, 237-252.

[15] Park, H.: An error bounded approximate method for representing planar curves in B-splines, Computer Aided Geometric Design, 21, 2004, 479-497.

[16] Park, H.; Lee, J.-H.: B-spline curve fitting based on adaptive curve refinement using dominant points, Computer-Aided Design, 39, 2007, 439-451.

[17] Pavlidis, T.: Algorithm for shape analysis of contours and waveform, IEEE Trans. PAMI, 2, 1980, 301-312.

[18] Philip, K. P.; Dove, E. L.; Chandran, K. B.: A graph search-based algorithm for detection of closed contours in images, Proc. IEEE Engineering in Medicine and Biology Society, 1990, 728-729.

[19] Piegl, L.; Tiller, W.: The NURBS Book, Springer-Verlag, New York, NY, 1997.

[20] Piegl, L.; Tiller, W.: Least-squares NURBS curve approximation with arbitrary end derivatives, Engineering with Computers, 16, 2000, 109-116.

[21] Ren, M.; Yang, J.; and Sun, H.: Tracing boundary contours in a binary image, Image and Vision Computing, 20(1), 2002, 125-131.

[22] Rogers, D. F.; Fog, N. G.: Constrained B-spline curve and surface fitting, Computer-Aided Design, 21, 1989, 641-648.

[23] Sabra, W.; Khouzam, M.; Chanu, A.; Martel, S.: Use of 3D potential field and an enhanced breadth-first search algorithms for the path planning of microdevices propelled in the cardiovascular system, Proc. of the 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference, Shanghai, China, September 1-5, 2005.

[24] Sappa, A.-D.; Vintimilla, B.-X.: Edge point linking by means of global and local schemes, IEEE Conf. on Signal Image Technology and Internet-based Systems – SITIS, 2006, 551-560.

[25] Sonka, V. H. M.; Boyle, R.: Image Processing, Analysis, and Machine Vision, Brooks/Cole, 1998.

[26] Sun, W.: Bio-CAD, Computer-Aided Design, 37(11), 2005, 1095-1096.

[27] Sun, W.; Starly, B.; Nam, J.; Darling, A.: Bio-CAD modeling and its applications in computer-aided tissue engineering, Computer-Aided Design, 37(11), 2005, 1097-1114.

[28] Testi, D.; Zannoni, C.; Cappello, A.; Viceconti, M.: Borer tracing algorithm implementation for the femoral geometry reconstruction, Comp. Meth. & Programs in Biomedicine, 65(3), 2001, 175-182.

[29] Trier, O. D.; Jain, A. K.; Taxt, T.: Feature extraction methods for character recognition - a survey, Pattern Recognition, 29, 1996, 641-662.

[30] Yu, D.; Yan, H.: An efficient algorithm for smoothing linearization and detection of structural feature points of binary image contours, Pattern Recognition, 30, 1997, 57-69.

[31] Zaidi H.; and Tsui B.M.W.: Review of Computational Anthropomorphic Anatomical and Physiological Models, Proceedings of the IEEE, 97(12), 2009, 1938 – 1953.

[32] Zhu, P.; Chirlian, M.: On critical point detection of digital shapes, IEEE Trans. PAMI, 17, 1995, 737-748.

[33] Ziou, D.; Tabbone, S.: Edge detection techniques – An overview, Pattern Recognition and Image Analysis, 8, 1998, 537-559.

[34] Vassilev, T. I.: Fair interpolation and approximation of B-splines by energy minimization and point insertion, Computer-Aided Design, 28, 1996, 753-760.

[35] Wang, W.; Pottmann, H.; Liu, Y.: Fitting B-spline curves to point clouds by curvature-based squared distance minimization, ACM Transaction on Graphics, 25, 2006, 214-238.