2D Contour Smoothing and Surface Reconstruction of Tubular CT-Scanned Anatomical Structures

George Biros
Biomedical Engineering Program
Carnegie Mellon University
Pittsburgh, PA 15213
biros@andrew.cmu.edu

A Thesis
Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

December 1996

I am always doing that which I can not do,
in order that I may learn how to do it.

Pablo Picasso

Advisors:

Branislav Jaramaz, Ph.D.
Center for Orthopaedic Research
Shadyside Hospital
5200 Centre Avenue, Suite 309
Pittsburgh PA 15232
branko@cs.cmu.edu

Prof. Omar Ghattas
Computational Mechanics Laboratory
Department of Civil and Environmental Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
oghattas@cs.cmu.edu
http://www.cs.cmu.edu/~oghattas

Contents

Acknowledgments

I am deeply grateful to my advisors Dr. Ghattas and Dr. Jaramaz for their guidance and help throughout the course of this project. Their advice and comments were essential and motivating to help me accomplish my work. I would like to thank Dr. Jaramaz, Dr. DiGioia and all the staff of the COR for they created an extremely friendly and supportive environment to me. I am thankful to all my fellow students in the Computational Mechanics Laboratory and especially my friend Aggelos Tsikas for his valuable support. I also want to thank Dr. Kallivokas for his encouragement and suggestions.

I owe so much to Ms. Hilda Diamond, director of the Biomedical Engineering Program. She was always there when I needed her help. Finally I have to thank Fulbright Foundation, NSF Foundation and Dr. Takeo Kanade. It would be impossible to complete my work without their financial support.

Abstract. In this work several aspects of the computer modeling of anatomical structures are considered. A short review of the current status in the topic of surface reconstruction is presented. The goal of this study is to use CT-scan data in order to construct a finite element model and then perform analysis of the mechanics of the total hip replacement procedure. Imaging software is used to extract the contours of the object. Two methods are tested for surface reconstruction; triangulation and B-splines. Public domain software (INRIA[1] nuages) is used to obtain triangulated C^0 surfaces and then commercial software to complete the analysis. Meshing and Boolean operations with triangulated surfaces failed in many cases so the B-splines approach was persued. Known algorithms for curve smoothing and C^2 surface reconstruction were implemented in C programming language. For each contour a B-Spline representation of the original data set is created and then a knot removal algorithm is applied to smooth the curves and reduce the number of control points. A tensor product B-Spline surface is created based on the smoothed curve set. Output of the curves and surfaces is in IGES[2] format. These files were pipelined to mesh generation using PATRAN P3/MSC[3] and ANSYS 5.2[4] for the femur and the acetabulum.

Keywords. free-form surfaces, surface reconstruction, medical imaging, meshing, finite elements, biomechanics, B-Splines, NURBS, solid modeling, curve smoothing, contours, CT-Scan

1 Introduction

1.1 Biomechanics

The medical community has reached a point where a precise, patient specific, simulation of surgical procedures is highly in demand. The motivation for this work is the need for quantitative analysis of both anatomical structures and their interaction with medical devices.

The scope of biomechanics is to quantify and clarify tissue/tissue and tissue/device mechanical interactions. The complexity of the human body is such that analytical methods fail to answer these questions. They are good for providing qualitative information and insight into the problems under consideration. Two are the main paths; experimental and computer aided numerical investigation. Experimental methods have several drawbacks such as:

- Usually only surface information can be obtained and very often the area of interest lies inside the body.
- Calibration of measuring devices is difficult. Often measurements are performed on soft tissue and it is hard to specify a zero strain setting.
- It is difficult to obtain specimens and even more difficult to perform the experiments *in vivo*.

Nevertheless experimental methods are very important. The constitutive equations of living tissues are still unknown although much progress has been done. Using experiments one can explore various continuum

[1] Institut National de Recherche en Informatique et en Automatique, Grenoble, France
[2] Initial Graphics Exchange Specification
[3] The MacNeal-Schwendler Corporation, 815 Colorado Boulevard, Los Angeles, CA 90041
[4] ANSYS,Inc. ,201 Johnson Road, Houston, PA 15342-1300

models and come up with one which matches the tissue's response. Experimental methods are also essential when trying to validate numerical results. Furthermore it is an active area of research by itself since reliable techniques must be developed in order to measure various variables inside the body without harming the tissues.

Numerical analysis is the only alternative when dealing with the above-mentioned problems. Having a constitutive law, one can investigate the physics of the body in a very efficient way. Many "what if" situations can be explored and thus theories about the function of the body can be scrutinized and invasive techniques optimized. Our interests focus on the area of orthopaedic biomechanics. We investigate hip replacement surgical procedures.

The analysis of the hip replacement mechanics can be divided into two parts. The first is the behavior of the pelvis when inserting the acetabular cup. The second is the insertion of the femoral implant in the intramedullary canal of the femur.

An crucial issue in total hip arthoplasty is reduction or elimination of the need for a revision surgery. This kind of operation is commonly required 10 to 15 years after the initial hip replacement, usually due to bone-implant bond loosening. The etiology for bone loosening is usually biocompatibility, stress shielding, initial instabilities and inadequate bone ingrowth. In order to optimize the parameters of the surgery based on the particular needs, a patient specific analysis is necessary. This analysis will lead to optimal implant size and positioning.

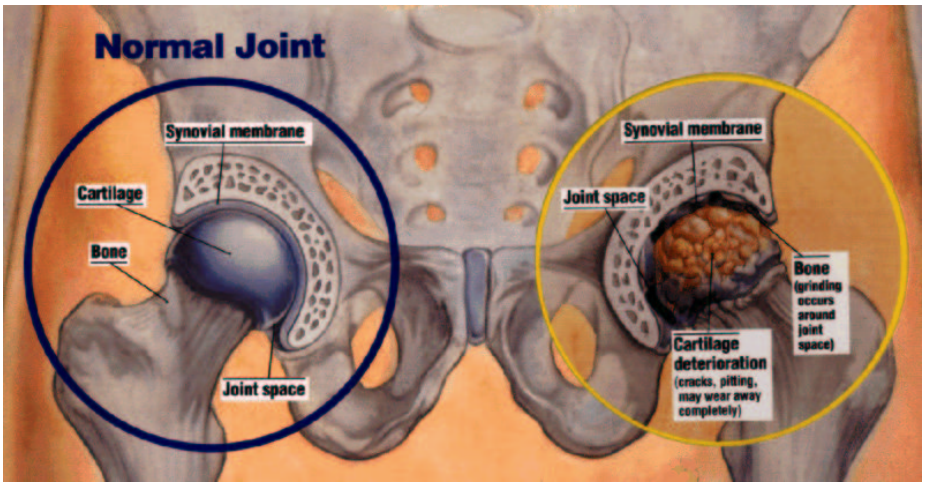
1.2 Finite Elements

Because of the complex interactions of many mechanical and geometric variables at the bone-prosthesis interface, finite element analysis (FEA) must be used in order to simulate the insertion of the implant. This simulation becomes important in the case of cementless procedures. Prior to bony ingrowth, the cementless femoral component must rely on its contact with the cortical endosteum to achieve stability. Finite elements can be used to optimize the size and orientation of the implant and minimize the initial instability and time [4].

FEA of the hip is a very difficult problem. It is almost impossible to take all the real parameters into consideration since the problem will become impossible to solve. Here are some aspects one has to consider:

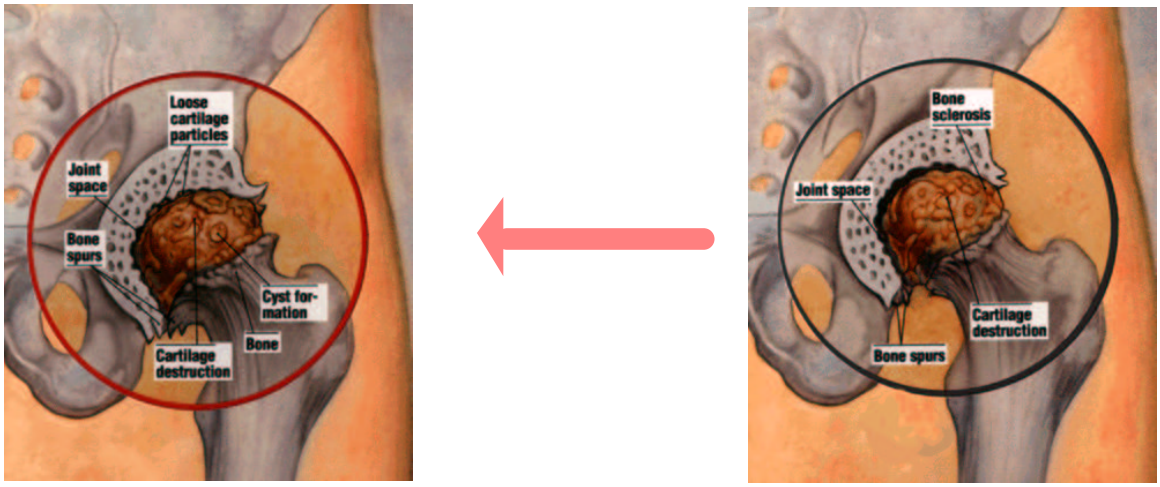
Material properties.

- Nonlinearity.
    1. Tissue admits finite deformations.
    2. Trabecular bone plastifies.
    3. Bone adaptivity and remodeling must be taken into account.
- Trabecular bone is orthotropic.
- Inhomogeneity
    1. Upper femur consists mainly of cancellous bone.



Stage 1

OSTEOARTHRITIS OF THE HIP



Stage 3 **Stage 2**

Stage 1: Deterioration of cartilage begins
Stage 2: Grinding of bones and deformation of joint begins
Stage 3: Joint suffers excessive damage.

Figure 1: Osteoathritis of the hip is one of the main pathologies that create the need for total hip replacement.

TOTAL HIP ARTHOPLASTY - (THA)

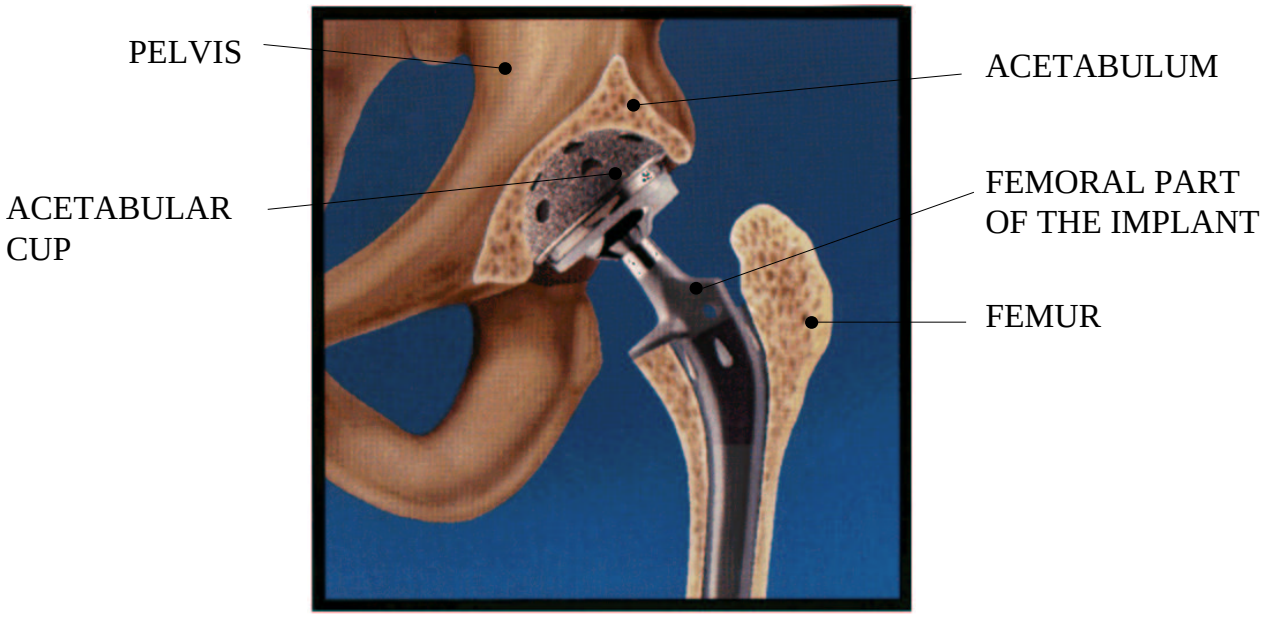
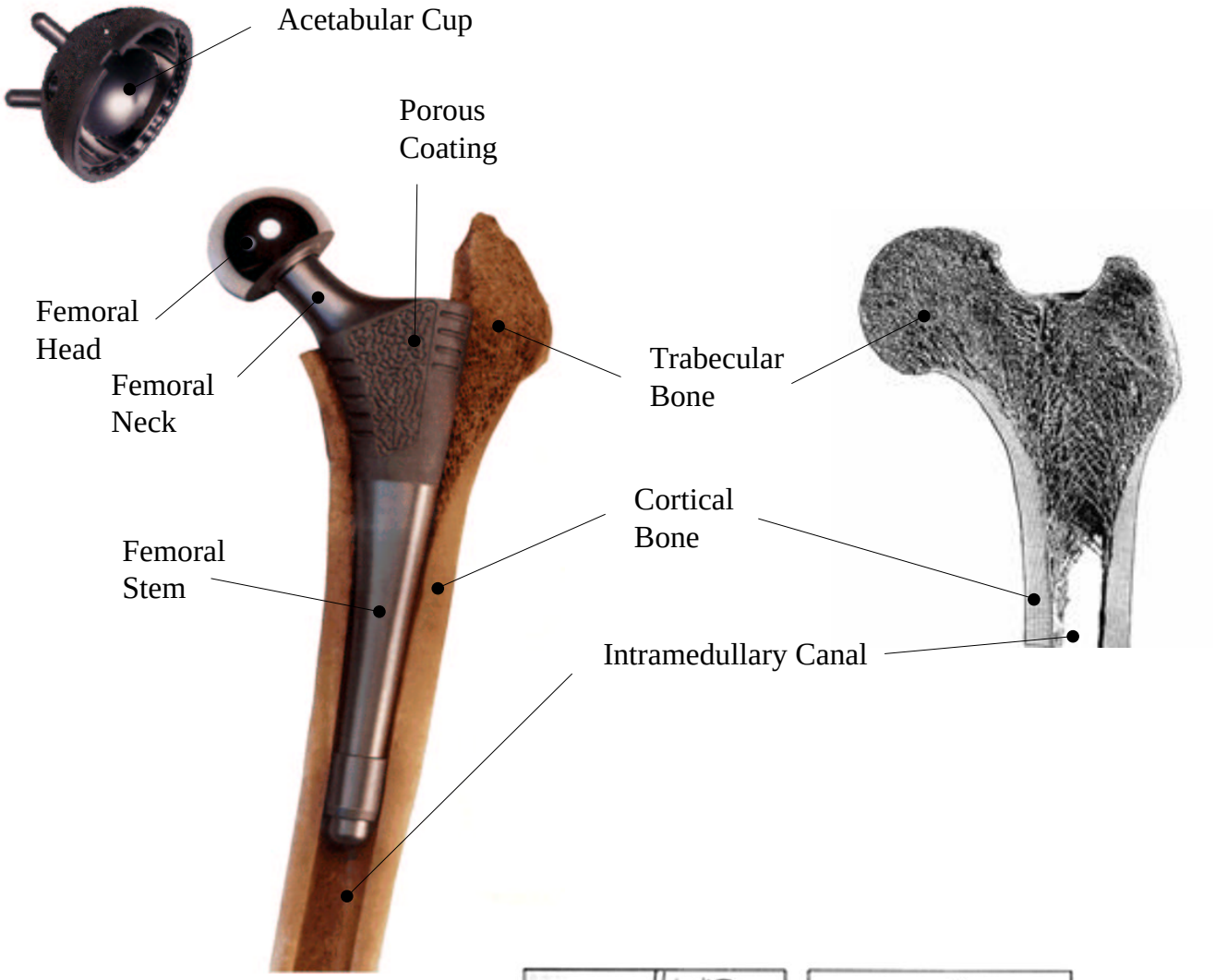


Figure 2: X-Ray of an implanted THR prosthesis. The femur head has been replaced by a metallic prosthesis. An acetabular component is also implanted in order to replicate the joint functionality.



Surgical procedure.

- Placement of acetabular cup.
- Drilling aand broaching of the intramedullary canal.
- Insertion of the femoral stem.

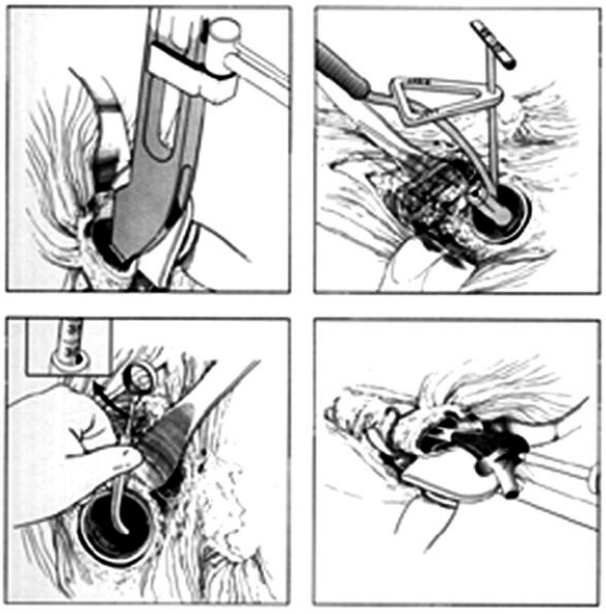


Figure 3: Main parts of a total hip replacement prosthesis. Acetabular cup and femoral stem.

2. Femur outer layer is cortical bone.

Geometry. Complex 3-D objects which are difficult to describe mathematically and even more difficult to mesh.

Interface and boundary conditions.

- Final contact area and displacements on this region have to be computed.
- Complicated-difficult to determine muscle forces.

1.3 Solid Modeling

In order to perform numerical analysis of an object, one must have its geometric description. The complexity of the human bones makes their mathematical description extremely difficult. Great amount of work has taken place in the previous years, mainly in the area of medical imaging.

The common procedure is the following: One takes MRI or CT-scan data representing cross sectional images of the human body. The extraction of the geometry of the structures under consideration takes place using image processing techniques. The output of these programs is a set of points describing the object in 3-D space as a set of planar contours.

After this step a further processing of data has to take place in order to reconstruct the boundary of the object. Various methods have been developed, each one having its drawbacks. The main concerns are the required mathematical smoothness of the surface, whether or not further manipulation of the object will take place but the most important issue is whether the reconstruction is done for imaging or numerical analysis purposes.

1.4 Analysis of The Hip Replacement

We investigate press fit Total Hip Arthoplasty (THA). The ultimate focus is 3-D patient specific finite element analysis of the femoral part insertion into the intramedullary canal. We begin with CT-scan data of the femur and we have to reconstruct the femur as a solid. Then we have to perform Boolean operations in order to create the hole where the femoral implant will be inserted, and proceed with the numerical investigation of the insertion. A similar analysis will be done for the acetabular cup modeling, on the pelvis.

2 Surface Reconstruction

2.1 General Considerations

There are three well-established paradigms for representing solids that are based on the boundary of the object, spatial subdivision and construction from primitives using Boolean operations [5].

The boundary of a solid consists of vertices, edges and faces. For each of these entities, the geometric part of the representation fixes the shape and/or location in space, and the topological and geometrical information is a boundary representation (B-rep) of a solid.

Instead of representing the boundary of an object, the object can be represented explicitly by its volume. The volume is represented as a collection of cells of a partition of space. The various data structures differ in how they organize the cells and what information about the object each cell contains. Some widely used data structures are hierarchical and subdivide space recursively. The simplest hierarchical data structure is the region octree based on regular decomposition. The method partitions a cuboidal region into eight equally sized octants. The region is represented by a node in the region octree, and the eight octants are its eight children. Region octrees are suitable for solids with faces that are parallel to the principal axes. Solids with inclined faces are approximated by region octrees. Thus they give only a rough description of the boundary of the object. For further details see [6].

Conceptually similar to cell representation techniques are the *voxel* methods. The voxel[5] method is a volume representation via the exhaustive enumeration of the occupancy of elementary cells that lie on a uniform 3-D grid. For each cell, a binary value indicates whether that cell is either inside or outside the object. This is called binary voxel model method. Other voxels techniques used in MRI and CT imaging allow more values for each cell in order to model different tissues. The accuracy of the approximation depends on the subdivision level. These methods provide unambiguous representations of objects and Boolean operations are straightforward to implement. However, other operations like geometric transformations and displaying can be tedious and computationally expensive. Moreover, to our knowledge, a lack of interface exists between commercial mesh generators and voxel models. There are works done on extracting boundary surfaces out of a voxel model. Additionally efficient algorithms have been described for the conversion from both polygonal surfaces and parametric surface patches, to voxel based models.

Both the boundary-based and the volume-based representations we have discussed herein are explicit representations. An alternative description is one in which a solid is described in terms of Boolean operations on simple volumetric primitives. This is an implicit method and is called *Constructive Solid Geometry*. A CSG representation is a tree structure in which the internal nodes represent Boolean operations and transformations[6], and the leaves represent primitives.The standard primitives include the half space, simple prismatic polyhedra, natural quadrics (sphere, cone, cylinder) and the torus.

Finally a different method to represent a solid is the *skeleton*. The interior skeleton [5] of a three-dimensional solid is the locus of the centers of all inscribed maximal spheres. A sphere is maximal if there is no other inscribed sphere that contains it completely. This kind of shape representation was proposed by computer vision research groups and is useful for generating 2-D finite element meshes. However, this idea is still under development.

In general, the main parameters to consider when analyzing geometric modeling problems are:

- The source of geometric data.
- The required precision.

[5] A *voxel* is a cube-like cell or a volume pixel.

[6] Transformations are geometric operations that position and orient the solid represented by the subtree rotations, translations and scaling.

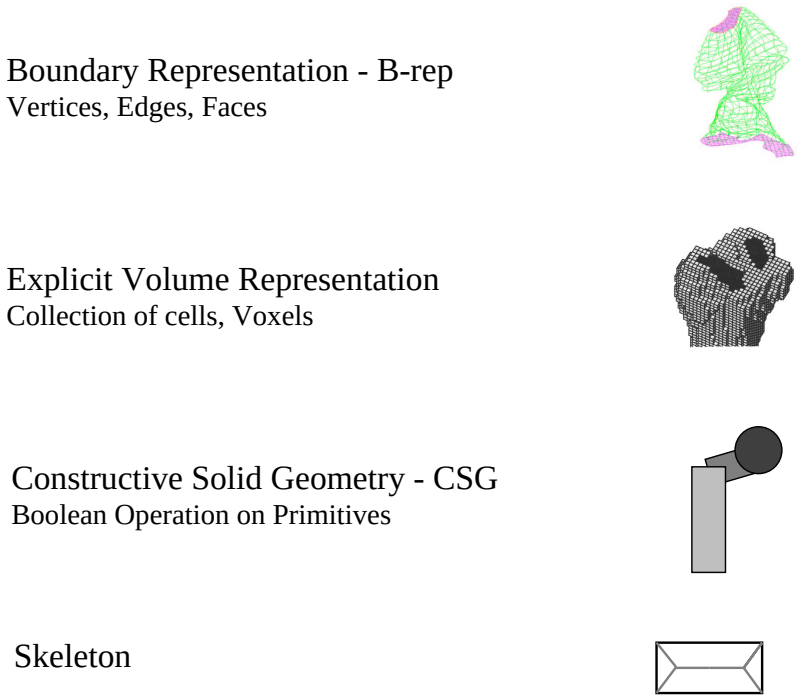


Figure 4: The main technologies for solid modeling.

- Algorithm complexity.
- Storage cost.
- Computational resources.
- Display complexity.
- Desired manipulations.
- Representation conversions.
- Links to FEM mesh generators.
- Links to other CAD programs.
- Manufacturing and dimensional control procedures.
- Future usefulness.

2.2 A Special Case: Anatomical Structures

The problem of reconstructing a three-dimensional surface from a set of planar sections[7] is an important problem. In clinical medicine, the data generated by various imaging technologies such as CT, ultrasound

[7] A *section* is the set of contours formed by one slice through an area of interest. The contours in a section do not necessarily come from the same object, and an object may be represented by more than one contour in a section. A *contour* is a simple polygon representing the intersection of the surface of an object and the plane of section.

or magnetic resonance imaging (MRI) provide a series of slices through the object of study. We review the basic technologies on this subject.

A CSG approach is very difficult. Anatomical structures have very complex geometry which is impossible to describe with simple primitives. For example, a femur could probably be approximated by a cylinder but we are dealing with a patient specific "exact" anatomical description. One could allow more complex primitives, but computationally, Boolean operations are either prohibitively expensive or insufficiently robust.

Therefore voxel and surface methods are the methods which are used. Volume based approaches assume that data are available as a three dimensional grid and a voxel representation is chosen. Surface based approaches assume the data define the intersection of a surface and a plane of sectioning. Which method is suitable depends on the nature of the data. When the available data is a dense three dimensional lattice of values, as in the case of the medical imaging techniques, a Voxel method can be used [8]. Going from voxel models to finite elements is straightforward. One can use voxels themselves as brick elements [9, 13]. However, in this case the element size is defined by the resolution of the model and when one wants good geometric approximation, one ends up with a high number of elements. When the problem includes nonlinearities and contact mechanics the computational requirements become very expensive.

Numerous workers have researched B-rep reconstruction of solids during the passed decade. The main idea is to perform image processing to the data in order to extract the contours of the object. Then apply an algorithm in order to join these contours together. The problem of generating a surface from a set of contours can be broken into several subproblems [10].

- The *Correspondence problem* is solved by determining the topological adjacency relationships between the contours of the data set. A solution to the correspondence problem determines the coarse topology of the final surface.
- The *tiling problem* is solved by generating the "best" topological adjacency relationships between the points of pairs of contours from adjacent sections by constructing a triangular mesh from their points. A commonly chosen metric for determining what is "best" is the minimization of the resulting area.
- The *branching problem* arises when an object is represented by a different number of contours in adjacent sections, in which case the standard method for solving the tiling problem cannot be used directly. A solution to the tiling and branching problems determines the topology of the surface and its coarse geometry.

These issues have been intensively studied by various investigators. The multiple branching problem has been the focus of research. However, the development of an efficient and automated algorithm for it remains an open problem.

Keppel's paper [16] seems to be the first publication on surface reconstruction from planar slices. These surfaces are constructed solely from elementary triangular tiles, each defined between two consecutive points on the same contour and a single point on an adjacent contour. Constructing this surface is shown to be equivalent to finding a path in a directed graph. However the method depends on the fact that there is a one-to-one relationship between contours. Christiansen and Sederberg [14] described a method that handles

some branching structures but requires user intervention in complex cases. Various researchers proposed methods based on a concatenation of contours. Boissonat [17] proposed a solution based on a Delaunay triangulation between each pair of slices. First, the volume of the object is created and then the boundary is extracted. This method is difficult to implement and does not take into account the topological similarity of the contours. Nevertheless, as a general method, it produces nice results and its code is available on the Internet [8]. Note though, that the Delaunay triangulation does not pose any constraints to the quality of the triangulation.

All these approaches result in a C^0 surface. From our experience these surfaces might create problems during the meshing stage. Usually triangulation algorithms are devised for visualization purposes and therefore they do not pose any constraints on the shape of the triangles. When this kind of triangulation is pipelined to a mesh generator it might lead to failure since the meshing algorithm respects the specified boundary. Again researchers have worked on this field to produce smooth surfaces. Based on the Delaunay triangulation and barycentric implicit[9] Bernstein-Bézier patches, Bajaj [11] has devised an algorithm for C^1 surfaces. Algorithms which take a C^0 piecewise surface and produce C^n piecewise surfaces also exist. Two different approaches include that of Jimenez [19], in which the surface is represented using trigonometric interpolation, and Meyling [18] who uses B-spline functions in spherical coordinates to represent starlike[10] objects. Bajaj [10] explores a technique that goes from the points to the mesh without attempting any surface reconstruction. The algorithm creates an unstructured 3D triangular (tetrahedral) mesh of the solid. However, if one wants to perform Boolean operations to the anatomical structure this approach is inadequate.

2.3 A Simple Approach

The problem is how to use CT-scan data in order to describe mathematically the femur and create a computational mesh based on this description. We decided to use B-rep methods. First we extract the contours of the femur and then we pass a surface through these contours. We are using ANSYS 5.2 and PATRAN P3/MSC as mesh generators and solvers. From the beginning, voxel methods were excluded since the aforementioned packages do not offer any interface to this representation. In the first place we have used a public domain triangulation program (nuages), which is based on the algorithm described in [17].

One issue not often mentioned in the literature was the quality of the CT-scan data. As we will see, the procedure of extracting the boundaries of the object under consideration is still under development, and the current practices induce error to the output data. Usually human intervention is necessary in order to process the CT-scan, with the aid of imaging software. This procedure produces 300-1000 points per contour. For the pelvis approximately 200 slices are used to describe its geometry. This makes a total of 200K points to interpolate. It is apparent that any algorithm whose complexity is greater than $O(n^2)$ gets prohibitively slow[11]. The most important problem is that this complexity will propagate to all the steps of the analysis; that is, displaying, meshing, and solving.

[8] Currently at: `http://www.inria.fr/prisme/personnel/geiger/nuages.html`

[9] *Implicit* surfaces are algebraic surfaces i.e. given in the form: $f(x, y, z) = c; c \equiv const.$ On the other hand *parametric* surfaces are given in the following form: $x := x(u, v), y := y(u, v), z := z(u, v)$.

[10] A *starlike* solid is one that has at least one internal point such that the lines connecting this point and all the points on the surface of the object are internal.

[11] For example the worst case complexity for the Delaunay triangulation in the R^3 is $O(n^2)$ and the worst case for incremental Delaunay triangulation is $O(n^3)$ [17]

A logical solution is to try to reduce the number of points describing each contour. Existing software in the lab uses piecewise linear interpolation and smoothing of the points, as follows:

$$If\ dist(\mathbf{P_i}, \overline{\mathbf{P_{i-1}P_{i+1}}}) \geq Tolerance\ Then$$
$$Discard\ \mathbf{P_i}.$$
$$Where\ \mathbf{P_i}, \mathbf{P_{i+1}} \in \Re^3.$$

The new set of contours was triangulated and used as input to ANSYS. We managed to mesh the surface of the femur. Additionally we managed to perform Boolean operations and to create the exact geometry of the femur before inserting the femoral part of the implant. However, the existence of nearly degenerated triangles led the volume meshing algorithm to failure. The shortcoming of these triangulated representations is that they limit the biggest element size the generator has to work with (Fig. 5). Moreover the whole procedure of Boolean operations was tedious and difficult to reproduce. Since our goal is not simply to perform one analysis, but to create a patient specific automated procedure, we decided to step back and change the reconstruction technique.

In order to allow mesh generators more flexibility a smooth surface must be used. The dominating mathematical tool for describing curves and surfaces are splines or more specifically Non Uniform Rational B-Splines (NURBS). Splines are the *lingua franca* in the CAD software, and they are included as an entity in IGES, STEP, DXF and other graphics exchange protocols.

Having the points of a contour, we interpolate these points using cubic splines. A knot removal algorithm is applied in order to reduce the necessary data and smooth the contour. Then, having a stack of contours we form a tensor product B-Spline surface. It has to be mentioned that currently we do not deal with branching. After the processing of the contours is completed ,the selection of the contours which will be stacked together is done by the user. In this way all the individual branches of the object are created and output as IGES 5.2 files. We use PATRAN 3 in order to join the individual branches.

Boolean operations remain a problem since at this moment no available algorithm exists to efficiently address this problem. Parametric patches are inherently difficult to work with. For example the intersection of an m-degree algebraic surface with a degree-n algebraic surface is a curve of degree mn. The intersection of two bicubic surfaces each with algebraic degree 18, could result in a space curve of degree 324.

Finally we inserted these IGES files to ANSYS and PATRAN for mesh generation and solving. We used PATRAN for meshing and ANSYS for solving. We performed this sequence for two femurs and one pelvis. The meshing of the pelvis is still problematic.

3 Processing CT-Scans Output

In collaboration with Shadyside Hospital we have obtained CT-scans from cadavers and patients. These CT-scans include both the pelvis and the femur. In order to extract the contours we used the commercial imag-

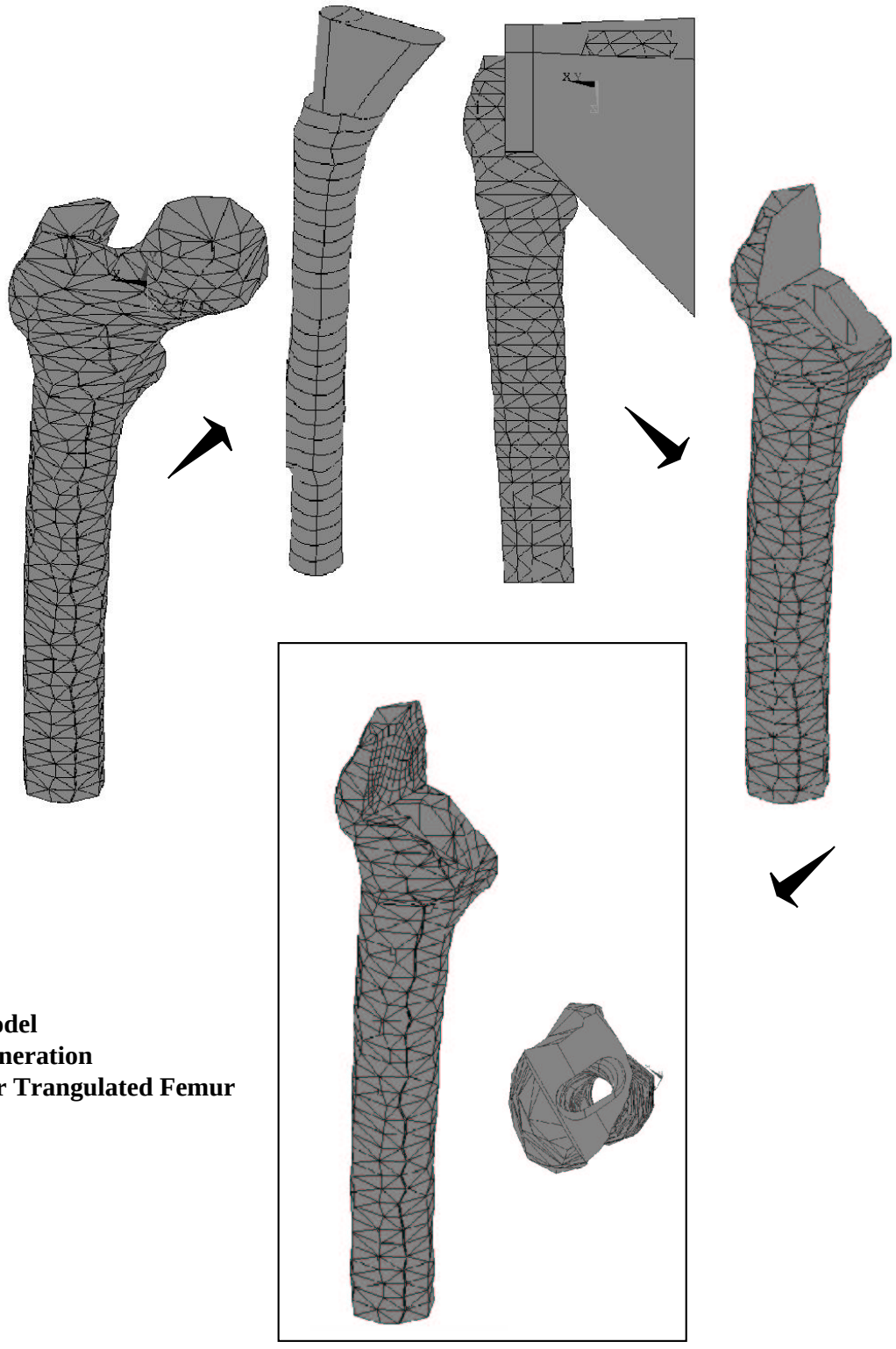


Figure 5: Surface reconstruction using triangulation and Boolean operations.

ing package AVS[12]. This package uses modules (functions) to perform operations and is programmable. The program to do contour extraction was developed by the MRCAS[13] group of the Robotics Institute at Carnegie Mellon University.

The thresholding technique is used to separate bone from soft tissue. In the grey scale lighter regions indicate dense material and the dark regions softer material. Using an upper and a lower value we can create sets of points whose grayscale color corresponds to bone. Then a seed is inserted to the regions of interest to distinguish between different regions of bone (pelvis and femur in our case). This seed is propagated from slice to slice in the vertical direction[14]. The next step is to create a closed region and extract the boundary of this region. Sometimes the bone quality is such that human intervention is necessary in order to complete this task. The output of the file is the one required by the nuages code and is described in the Appendix.

4 Triangulation

Our first approach was to use existing code for surface reconstruction. The intermediate step of linear approximation of the contours did not work that well. The resulting tessellation had almost degenerate triangles which led to the failure of the mesh generator. However we consider this approach as an alternative because it handles the problem of branching and is fully automated.

Our code allows the user to choose the approximation error and thus determine the number of points which define each (piecewise linear) contour and the step between the slices. This is an implicit way to control the size of the triangles which in turn defines an upper bound to the element size during the mesh generation.

Furthermore, if one wants to work with a smoother surface there are ways to obtain a C^1 or a C^2 triangulation starting from a C^0 one. Ihm [12] uses algebraic quintic surfaces and Hermite interpolation. A similar approach is followed by Menon [22] who uses *trunctets* to construct the surface, essentially using algebraic surfaces. He transforms a B-rep solid to union of truncated tetrahedra which have one facet in the form of an algebraic surface. Fong [21] and García [20] use barycentric coordinates and B-Spline patches. The basic idea is to average the normals of a set of triangles and fit polynomial surfaces. Algebraic patches have the important advantage that they are of lower algebraic degrees so all the computations become easier to perform. The price is reduced flexibility, but in triangulation smoothing this is not a problem.

[12] Advanced Visual Systems Inc., 300 Fifth Ave., Aeltham MA 02154

[13] Medical Robotics and Computer Assisted Surgery

[14] The coordinate system used has the xy-plane aligned with the slices.

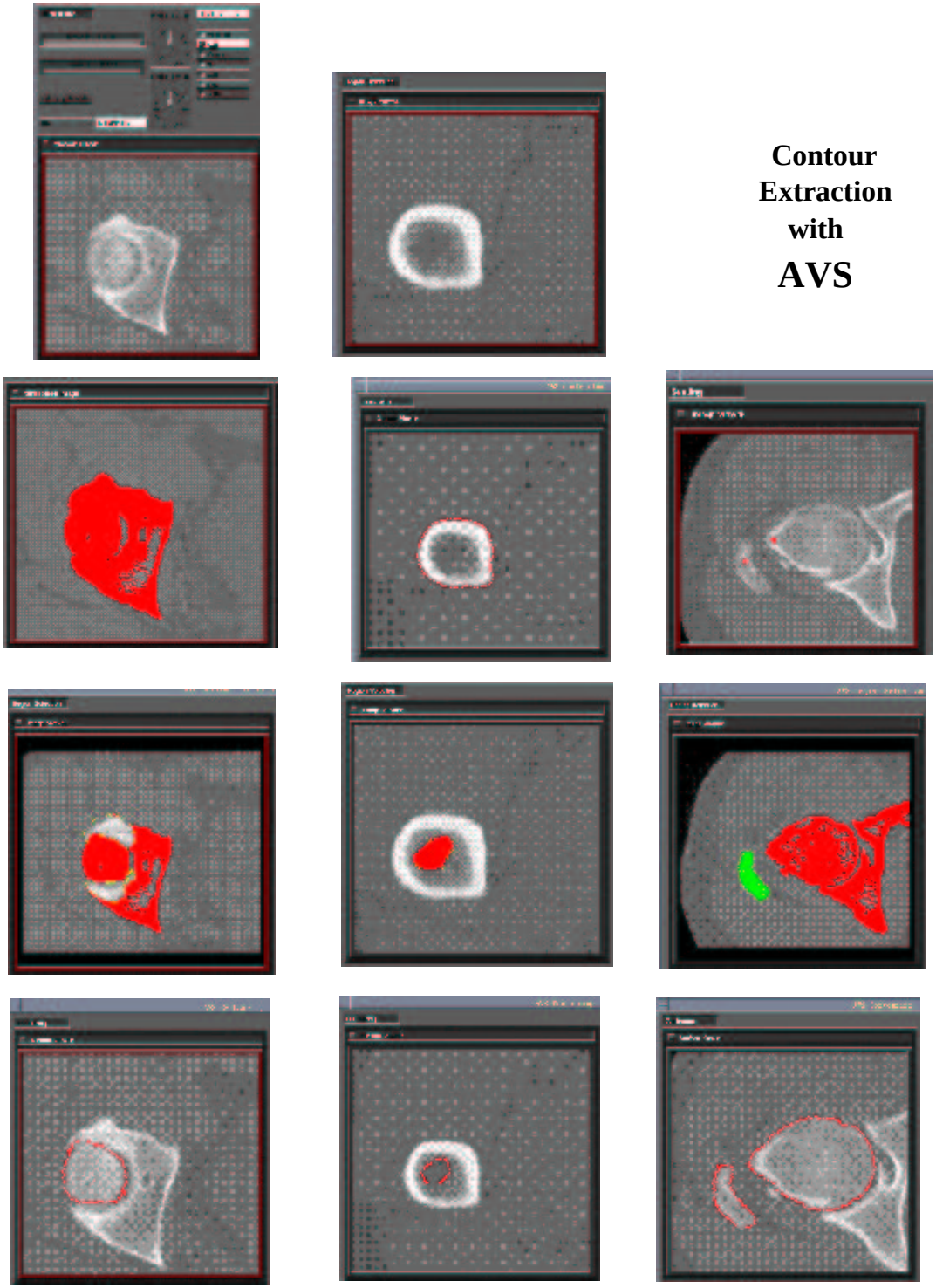


Figure 6: Sequence of the thresholding procedure.

5 Curve Smoothing

5.1 Mathematical Background

Spline functions (splines) are currently used in diverse domains of numerical analysis (interpolation, data smoothing, geometric modeling, numerical solutions of partial differential and integral equations, etc.). Based on the splines, NURBS have become a standard for describing complex geometries and is implemented in every kind of CAD software.

The advantages of splines and NURBS with respect their use in geometric modeling are:

- NURBS are genuine generalizations of non rational B-Splines.
- They offer a common mathematical form for representing and designing both standard analytic shapes[15] and free-form curves and surfaces.
- By manipulating the control points as well as the weights NURBS provide the flexibility required to design complex geometries.
- Evaluation is reasonably fast and computationally stable.
- B-splines and NURBS have a geometric tool kit (knot insertion/refinement/removal, degree elevation, splitting, smoothing,etc.).
- B-splines and NURBS are invariant under scaling, rotation, translation and shear as well as parallel and perspective projection.

However they have several drawbacks such as:

- Improper application of the weights can result in very bad parameterization, which can destroy subsequent surface constructions.
- Boolean operations. An example is surface/surface intersection where it is particularly difficult to handle the "just touch" or "overlap" cases.
- Fundamental algorithms as the inverse point mapping are subject to numerical instability.
- Displaying algorithms like rendering require C^0 representation of objects. Thus there is an inherent incompatibility between higher order splines and these algorithms.

So what are a spline function, B-splines and NURBS? A spline is a piecewise polynomial. It consists of polynomial pieces on subintervals joined together with certain continuity conditions. B-spline theory is a way of constructing a group of functions which form a basis for all possible splines up to a degree to a given interval and a given knot vector. NURBS is a generalization of B-splines in order to provide more flexibility and the ability to represent conics exactly.

[15] conics, quadrics, surfaces of revolution etc.

5.1.1 Splines

Definition. A function $s(x)$ defined on a finite interval $[a, b]$, is called a spline function of $degree\ k > 0$ and $order(k+1)$, having as $knots$ the strictly increasing sequence $x_j,\ j = 0, 1, \ldots, n-1$ $(x_0 = a, x_{n-1} = b)$, if the following two conditions are satisfied:

1. On each knot interval $[x_j, x_{j+1}],\ s(x)$ is given by a polynomial of degree k at most.

$$x \in [x_j, x_{j+1}],\ s(x) \in P_k,\ j = 0, \ldots, n-2. \tag{1}$$

2. The function $s(x)$ and its derivatives up to order $k-1$ are all continuous on $[a, b]$.

$$s(x) \in C^{k-1}[a, b]. \tag{2}$$

Definition. A spline function $s(x)$ is called *periodic* if in addition to (Eq. 1) - (Eq. 2) it satisfies

$$s^{(l)}(a) = s^{(l)}(b),\ l = 0, 1, \ldots, k-1.$$

Definition. A *natural* spline function is a spline of odd degree $k = 2l-1 (l \geq 2)$ which satisfies the additional constraints

$$s^{(l+j)}(a) = s^{(l+j)}(b) = 0,\ j = 0, 1, \ldots, l-2.$$

We now present a theorem to give an aspect of the geometric properties of the splines. The theorem is valid for cubic splines but it can be generalized for higher (odd) degree spline functions.

Theorem. Let f'' be continuous in $[a, b]$ and let $a = x_0 < x_1 < \ldots < x_{n-1} = b$. If s is the natural cubic spline interpolating f at the knots u_i for $0 \leq i \leq n$ then

$$\int_a^b [s''(x)]^2 dx \leq \int_a^b [f''(x)]^2 dx.$$

Recall that the curvature of a curve described by the equation $y = f(x)$ is the quantity

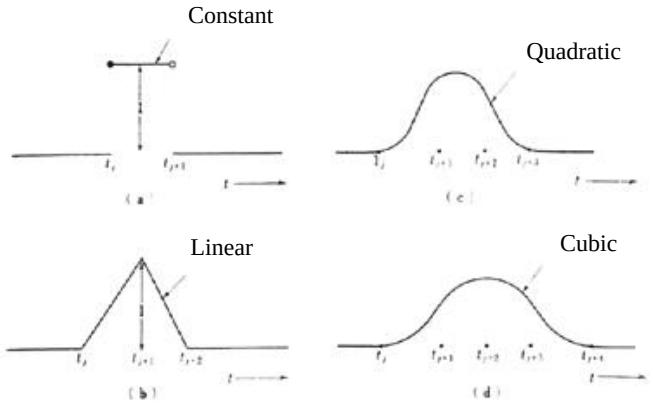
$$\mid f''(x) \mid [1 + \{f'(x)\}^2]^{-3/2}.$$

It is apparent that the natural spline produces the *smoothest* possible interpolating function.

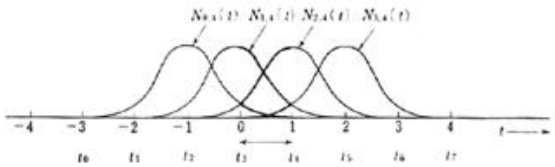
The vector space of functions satisfying (1)-(2) will be denoted by $S_{k,\mathbf{t}}$ and its dimension is $n + k - 1$ [25].

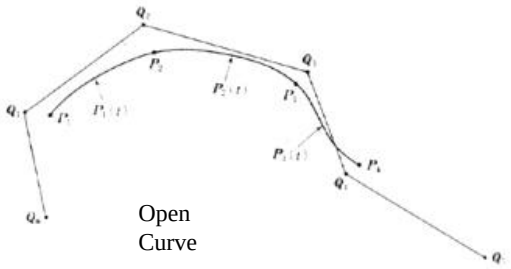
5.1.2 B-Splines

B-splines are a system of spline functions from which all other spline functions can be obtained by forming linear combinations. That is, these splines provide bases for certain spline spaces. The B-splines are distinguished by their elegant theory and their model behavior in numerical calculations.

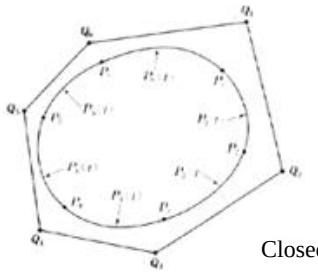


B-Splines

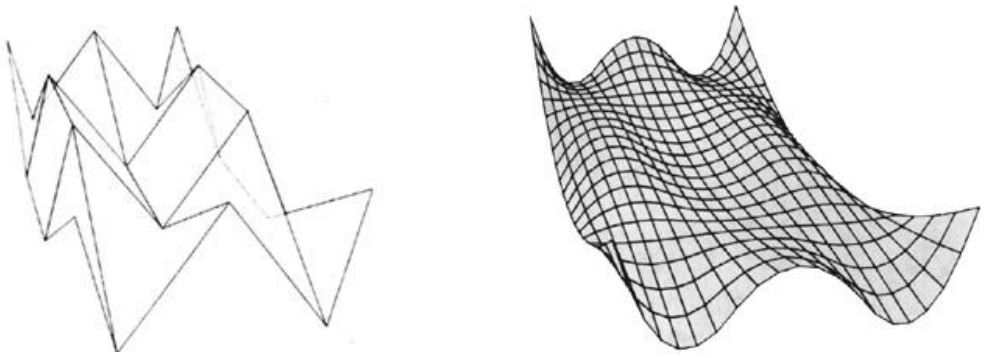




Open Curve



Closed Curve



Tensor Product Surface

Figure 7: Splines and B-splines. Curve and surface representation.

We begin with a knot vector of the form

$$\mathbf{t} := \{t_0, t_1, \ldots, t_{N+k}\}. \tag{3}$$

where k is the spline degree.

Definition. Let k denote the degree of the spline. Then a *normalized* (see Eq.8) B-spline $B_{i,k,\mathbf{t}}$ with knots $t_i, \ldots, t_{i+k+1}$ is defined as

$$B_{i,k,\mathbf{t}}(t) := (t_{i+k+1} - t_i) \sum_{j=0}^{k+1} \frac{(t_{i+j} - t)_+^k}{\prod_{\substack{l=0 \\ l \neq j}}^{k+1} (t_{i+j} - t_{i+l})} \tag{4}$$

where

$$(t_{i+j} - t)_+^k := \begin{cases} 0, & \text{if } t_{i+j} \leq t \\ (t_{i+j} - t)^k, & \text{if } t_{i+j} > t \end{cases}$$

For example a cubic B-spline on the knots $t_i, \ldots, t_{i+4}$ is given by :

$$B_{i,3,\mathbf{t}}(t) = \begin{cases} \frac{t-t_i}{t_{i+3}-t_i}\frac{t-t_i}{t_{i+2}-t_i}\frac{t-t_i}{t_{i+1}-t_i} & t_i \leq t \leq t_{i+1} \\ \frac{t-t_i}{t_{i+3}-t_i}\left(\frac{t-t_i}{t_{i+2}-t_i}\frac{t_{i+2}-t}{t_{i+2}-t_{i+1}} + \frac{t_{i+3}-t}{t_{i+3}-t_{i+1}}\frac{t-t_{i+1}}{t_{i+2}-t_{i+1}}\right) + & \\ \frac{t_{i+4}-t}{t_{i+4}-t_{i+1}}\frac{t-t_{i+1}}{t_{i+3}-t_{i+1}}\frac{t-t_{i+1}}{t_{i+2}-t_{i+1}} & t_{i+1} \leq t \leq t_{i+2} \\ \frac{t-t_i}{t_{i+3}-t_i}\frac{t_{i+3}-t}{t_{i+3}-t_{i+1}}\frac{t_{i+3}-t}{t_{i+3}-t_{i+2}} + & \\ \frac{t_{i+4}-t}{t_{i+4}-t_{i+2}}\left(\frac{t-t_{i+1}}{t_{i+3}-t_{i+1}}\frac{t_{i+3}-t}{t_{i+3}-t_{i+2}} + \frac{t_{i+4}-t}{t_{i+4}-t_{i+2}}\frac{t-t_{i+2}}{t_{i+3}-t_{i+2}}\right) & t_{i+2} \leq t \leq t_{i+3} \\ \frac{t_{i+4}-t}{t_{i+4}-t_{i+1}}\frac{t_{i+4}-t}{t_{i+4}-t_{i+2}}\frac{t_{i+4}-t}{t_{i+4}-t_{i+3}} & t_{i+3} \leq t \leq t_{i+4} \\ 0 & \text{otherwise} \end{cases}$$

Basic properties.

1. Recursion:

$$B_{i,k,\mathbf{t}}(t) = \frac{t - t_i}{t_{i+k} - t_i} B_{i,k-1,\mathbf{t}}(t) + \frac{t_{i+k+1} - t}{t_{i+k+1} - t_{i+1}} B_{i+1,k-1,\mathbf{t}}(t),$$

$$B_{i,0,\mathbf{t}}(t) = \begin{cases} 1, & \text{if } t \in [t_i, t_i + 1] \\ 0, & \text{if } t \notin [t_i, t_i + 1]. \end{cases} \tag{5}$$

2. Local support:

$$B_{i,k,\mathbf{t}}(t) = 0 \text{ if } t \notin [t_i, t_{i+k+1}].$$

3. Positivity:

$$B_{i,k,\mathbf{t}}(t) \geq 0 \ \forall t.$$

4. Boundary values:

$$B_{i,k,\mathbf{t}}^{(l)}(t_i) = B_{i,k,\mathbf{t}}^{(l)}(t_{i+k+1}) = 0, \ l = 0 \ldots, k-1.$$

5.1.3 Splines as Linear Combination of B-Splines

Given a knot vector $\mathbf{u}, (u_0 = a, u_{n-1} = b)$ with n elements we can construct $n - (k+1)$[16] linearly independent B-splines of degree k. However, the dimension of $S_{k,\mathbf{u}}$ is n+k-1 which means that in order to produce all possible splines in $[a, b]$ we have to expand the original knot vector. Hence, we need another set of $2k$ B-splines. So we create an *arbitrary* new knot vector $\mathbf{t}$ as follows:

$$\begin{array}{ccc} & t_0 \le t_1 \le \cdots \le t_k = a, & \\ t_i = u_j; & i = k, \ldots, n+k-1; & j = 0, \ldots, n-1. \\ & b = t_{n+k-1} \le t_{n+k} \le \ldots \le t_{n+2k-1}. & \end{array} \tag{6}$$

Thus we can construct n+k-1 B-splines $B_i, i = 0, \ldots, n+k-1$. Let $N = n+k-1$ then every spline $s(t) \in S_{k,\mathbf{u}}$ has a unique representation:

$$s(t) = \sum_{i=0}^{N-1} c_i B_{i,k,\mathbf{t}}(t), \tag{7}$$

in which the c_i are called the *B-spline coefficients* of $s(t)$. An important property of B-splines is that:

$$\sum_{i=0}^{n+k-1} B_{i,k}(t) \equiv 1 \; \forall t \in (a, b). \tag{8}$$

B-splines have many other properties like convexity, affinity, rules for integration and differentiation, etc. For more details see [23, 25, 26, 27].

Another important feature of B-spline functions is that one can use a family of discrete norms which approximate the usual L^p-norms well, at least for moderate values of k. Recall that the L^p-norm of a function s is defined by

$$\|s\|_{L^p} = \begin{cases} (\int |s|^p)^{1/p}, & \text{for } 1 \le p \le \infty \\ \sup|s|, & \text{for } p = \infty \end{cases}$$

and similarly the ℓ^p-norm of a vector $\mathbf{a} \in \Re^n$ is defined by

$$\|\mathbf{a}\|_{\ell^p} = \begin{cases} (\sum_i |a_i|^p)^{1/p}, & \text{for } 1 \le p \le \infty \\ \max_i|a_i|, & \text{for } p = \infty \end{cases}$$

If $s = \mathbf{b}_\mathbf{t}^T \mathbf{c}$ where $\mathbf{b}^T = (B_{0,k,\mathbf{t}}, \ldots, B_{N-1,k,\mathbf{t}})^T$ the ℓ^p, $\mathbf{t}$-norm of s is defined by

$$\|s\|_{\ell^p} = \begin{cases} (\sum_i |c_i|^p (t_{i+k+1} - t_i)/(k+1))^{1/p}, & \text{for } 1 \le p \le \infty \\ \max_i|c_i|, & \text{for } p = \infty \end{cases} \tag{9}$$

which can be written as $\|s\|_{\ell^p,\mathbf{t}} = \left\| \boldsymbol{E}_\mathbf{t}^{1/p} \mathbf{c} \right\|^{\ell^p}$, where $\boldsymbol{E}_\mathbf{t}^{1/p}$ is a diagonal matrix of dimension N with diagonal elements

$$e_{i,i} = \begin{cases} ((t_{i+k+1} - t_i)/(k+1))^{1/p}, & \text{for } 1 \le p \le \infty \\ 1, & \text{for } p = \infty \end{cases} \tag{10}$$

[16] Each B-spline $B_{i,k,\mathbf{u}}$ starts to the ith knot and extends to the $(i+k+1)$th knot.

The use of these norms is justified by the following inequality [23] , which denotes the equivalence of the $\ell^p, \mathbf{t}$ norms and the usual L^p norm.

$$D_k^{-1}\|s\|_{\ell^p,\mathbf{t}} \leq \|s\|_{L^p} \leq \|s\|_{\ell^p,\mathbf{t}} \tag{11}$$

Here D_k is a constant which depends only on k and for $k = 3$ $D_k = 10.03$ [23, 25]. These norms will be essential in the knot removal procedure.

5.1.4 Non Uniform Rational B-splines

When the theory of B-splines was under development most algorithms used a uniform distribution of the knots meaning that they were equally spaced. This caused high oscillations between control points and in general poor results when fitting points which were unevenly spaced. The definition of B-splines given in the previous section, is general and valid for non-uniform knot vectors.

A further step is to introduce the idea of weights associated with each spline B-spline coefficient.

Definition A NURBS spline is given from the following formula:

$$s(t) = \frac{\sum_{i=0}^{N-1} w_i c_i B_{i,k}(t)}{\sum_{i=0}^{N-1} w_i B_{i,k}(t)}. \tag{12}$$

The knot vector $\mathbf{t}$ on which the spline is defined is again the same as in (Eq.6). As we can see when all $w_i = 1$ we obtain the B-splines representation. The NURBS form was introduced to deal with the problem of exact representation of analytic shapes, and to introduce higher flexibility in the user interactive design. It is possible also to have NURBS and B-splines with identical knots in order to represent discontinuities. However, for our purpose B-spline representation with non-repeating knots is adequate. Hence, to avoid unnecessary software complexity from now on we will refer only to non-rational non-uniform B-splines.

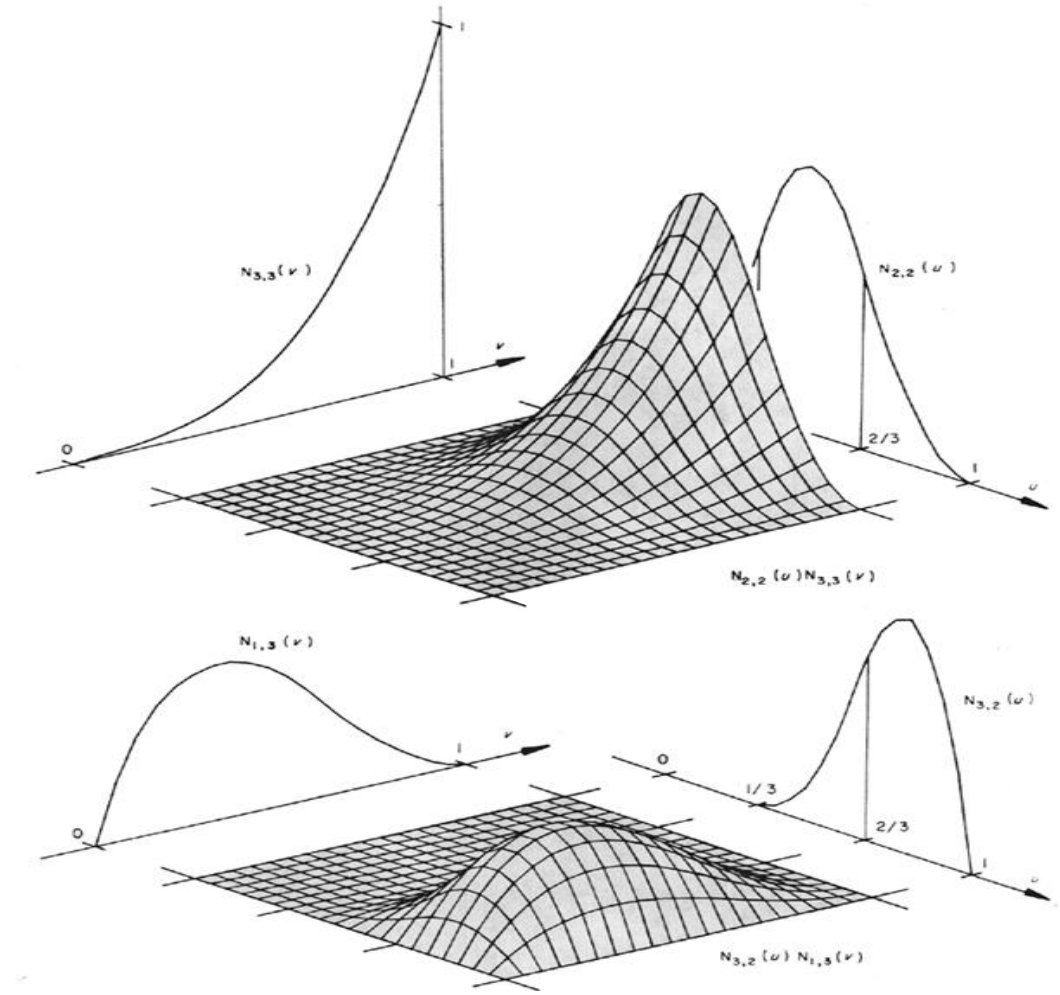
5.1.5 Bivariate Splines

The theory of one-variable splines can be extended in different ways to functions of more than one variable. The most common are the *tensor product splines* [23]. Most of the algorithms applied to univariate splines can be applied also to tensor product splines, but their restriction to rectangular domains is a serious disadvantage. As we mentioned before other types of splines (over triangular domains) are more flexible. However they are far more complex and therefore computationally less attractive.

Definition. Consider the strictly increasing sequences

$$a = \lambda_0 < \lambda_1 < \cdots < \lambda_{n-1} = b,$$
$$c = \mu_0 < \mu_1 < \cdots < \mu_{m-1} = d.$$

The function $s(x, y)$ is called a bivariate (tensor product) spline on $D = [a, b] \times [c, d]$, of degree $k > 0$ in x and $l > 0$ in y, with knots λ and μ in the x- and y-direction, if the following conditions are satisfied:



Source for all spline figures: *Curves and Surfaces in CAGD,* Fujio Yamaguchi.

Figure 8: Basis functions for bivariate surfaces.

1. On each sub-rectangle $D_{i,j} = [\lambda_i, \lambda_{i+1}] \times [\mu_j, \mu_{j+1}]$, $s(x, y)$ is given by a polynomial of degree k in x and l in y.

2. The function $s(x, y)$ and all its partial derivatives up to $k - 1$ and $l - 1$, are continuous on D .

$$\frac{\partial^{i+j} s(x, y)}{\partial x^i y^j} \in C(D), i = 0, \ldots, n - 1; j = 0, \ldots, m - 1.$$

Extending the ideas of vector spaces and basis functions, we can express every spline in D as a linear combination of B-splines. If we extend the λ and μ vectors as we did for the univariate spline we will obtain two new knots vectors $\mathbf{u}, \mathbf{v}$. Let $N = n + k - 1$, $M = m + l - 1$ then:

$$s(u, v) = \sum_{0}^{N-1} \sum_{0}^{M-1} c_{i,j} B_{i,k,\mathbf{u}}(u) G_{j,l,\mathbf{v}}(v), \tag{13}$$

where the $B_{i,k,\mathbf{u}}(x)$ and $G_{j,l,\mathbf{v}}(y)$ are the B-splines defined on $\mathbf{u}$ and $\mathbf{v}$ knot sequences respectively (Eq. 4).

5.1.6 B-Spline Representation of Curves and Surfaces

A parametric B-spline curve **f** in $\Re^d$ of degree k defined on the knot vector **t** where each component is a B-spline function with the same knot vector **t**, i.e.,

$$\begin{aligned} \mathbf{f}(t) &:= (f^1, \ldots, f^d)^T \\ f^1(t) &= \sum_i c_i^1 B_{i,k,\mathbf{t}}(t) \\ &\vdots \quad\quad \vdots \\ f^d(t) &= \sum_i c_i^d B_{i,k,\mathbf{t}}(t) \\ or & \\ \mathbf{f}(t) &= \sum_i \mathbf{c}_i B_{i,k,\mathbf{t}}(t). \end{aligned} \tag{14}$$

The coefficients $\mathbf{c}_i$ of the spline are vectors $\Re^d$. They are called also the control points of the spline. One property of splines is that they belong to the convex hull of their control points. This is important because it denotes the non-oscillatory nature of splines.

For the surface we use exactly the same pattern. Let k and l be the order of the splines in the u and v direction and let **u** and **v** be two knot vectors. Then the bivariate function to represent the surface is given by:

$$\mathbf{F}(u,v) = \sum_i \sum_j \mathbf{c}_{i,j} B_{i,k,\mathbf{u}}(u) B_{j,l,\mathbf{v}}(v) = \mathbf{b}_{k,\mathbf{u}}^T(u) \mathbf{C} \mathbf{b}_{l,\mathbf{v}}(v). \tag{15}$$

Where again $\mathbf{F} = (F^1, \ldots, F^d)$, and $\mathbf{c}_{i,j}$ are the control points of the surface. Note that Bézier curves and surfaces are a special case of B-splines and surfaces.

5.1.7 Discrete B-Splines

The theory of discrete B-splines and the associated algorithms were developed to provide a framework for understanding and implementing subdivision techniques for B-splines. Such technologies are useful for rendering and interactive design but they turned out to be also useful for numerical approximation and smoothing.

Consider a spline written in terms of B-splines on a knot vector τ and of degree k:

$$\begin{aligned} \boldsymbol{\tau} &:= (\tau_0, \ldots, \tau_{N+k}), \\ s(\tau) &= \sum_{i=0}^{N-1} c_i B_{i,k,\boldsymbol{\tau}}(\tau). \end{aligned}$$

There are certain situations where it is useful to increase the degrees of freedom of s by adding l additional knots to the already existing ones. Suppose we let $M = N + l$ and $\mathbf{t} := (t_0, \ldots, t_{M+k})$ be the new knot sequence so that $\tau \subset \mathbf{t}$. With $B_{j,k,\mathbf{t}}$ the B-splines on **t**, s can also be written as a linear combination of this

extended basis with unique coefficients d_j:

$$s(t) = \sum_{j=0}^{M-1} d_j B_{j,k,\mathbf{t}}(t).$$

The problem is given $k, N, M, \tau, \mathbf{t}$, as above compute d_j. There are several ways to do that. We can choose M points $\rho_0, \ldots, \rho_{M-1}$ and solve the interpolation problem,

$$\sum_{j=0}^{M-1} d_j B_{j,k,\mathbf{t}}(\rho_i) = s(\rho_i),\ i = 0, 1, \ldots, M-1.$$

If $t_j < \rho_j < t_{j+k+1}, j = 0, \ldots, M-1$, then this set of linear equations has a unique solution $d_0, \ldots, d_{M-1}$ [3]. Lyche *et al.* in [3] propose a method were d_j can computed as follows:

$$d_j = \sum_{0}^{N-1} b_{i,j} c_i,\ or\ \mathbf{d} = \mathbf{B}\mathbf{c},$$
$$b_{i,j} = a_{i,k+1,\tau,\mathbf{t}}(j).$$

The numbers $b_{i,j}$ are called *discrete splines* and the matrix $\mathbf{B}$ is called the *insertion matrix of order* $k+1$ from the knot vector $\boldsymbol{\tau}$ to knot vector $\mathbf{t}$.The numbers $a_i(j)$ are defined as follows:

Definition. Suppose $\tau_{i+k_1} > \tau_i$. Then

$$a_{i,1} = \begin{cases} 1 & \tau_i \le \tau_{i+j} \\ 0 & otherwise \end{cases}$$

Moreover for $k \ge 1$ and all i, j,

$$a_{i,k+1} = (t_{j+k} - \tau_i)\beta_{i,k}(j) + (\tau_{i+k+1} - t_{j+k})\beta_{i+1,k}(j),$$

where

$$\beta_{i,k+1} = \begin{cases} \frac{a_{i,k+1}}{\tau_{i+k+1}-\tau_i} & \tau_i < \tau_{i+k+1} \\ 0 & otherwise \end{cases}$$

Based on this definition Lyche *et al.* devised the *Oslo algorithm* to create the insertion matrix $\mathbf{B}$ given τ,$\mathbf{t}$ and k provided that $\tau \subset \mathbf{t}$. Based on their work we have implemented the Oslo algorithm in C. For further details about derivation and proofs see [3].

5.2 B-spline Curve Smoothing

5.2.1 Statement of the Problem

The contours extracted with the help of AVS consist, on average, of 400-1000 points. Two problems are associated with this procedure. The first one occurs when user intervention is necessary to modify the boundary of the bone resulting from the thresholding procedure. As a result, an error is introduced and the geometry becomes abnormal. The second problem is that the same geometry can be reproduced using less

points, even in the case when the noise is minimal. This happens because the module of AVS that creates the contour points does not recognize topological features, for example linear segments, which can described with less points.

Since we deal with planar contours we restrict the following formulation in the 2-D problem. The extension to any higher dimensions is straightforward. Given a set of N points $\mathbf{P}_i = (x_i, y_i), i = 0, \ldots, N-1$ we have to:

1. Interpolate these points with a parametric curve $\mathbf{f}(t)$ such that:

$$\mathbf{f}(t) := (x(t), y(t))$$
$$\mathbf{f}(t_i) = \mathbf{P}_i$$

   It is common practice in computational geometry to have $t \in [0, 1]$.

2. Find $\mathbf{g}(t)$ such that:

$$\| \mathbf{f} - \mathbf{g} \| \leq \epsilon$$

   in some suitable norm, where ϵ is a given acceptable error.

Many times the first step is skipped if it is known *a priori* that the data contain errors or due to the large number of sampled points. In order to proceed the form of the function $\mathbf{f}$ must be chosen. One may use polynomials, splines, trigonometric functions or some other suitable basis. In the case that splines are the choice these additional parameters must be determined:

- The degree k of the spline,
- The number and the position of the knots t_i,
- The coefficients $\mathbf{c}_i$, (see Eq. 14)

In this work we are using cubic splines $(k = 3)$. We do not need any higher continuity of derivatives thus we follow the common practice of the CAD community. As for the other parameters to be determined things are not nearly as simple. In the literature several spline fitting algorithms have been described which differ from each other in the choice of the knots and the approximation criterion. Various methods for knot creation have been proposed: least squares, weighted least squares, minimization of functionals, knot insertion, and knot removal algorithms [30, 31].

Most fitting algorithms deal with the problem of knot selection either arbitrarily or by solving computationally demanding, non-linear [27, 24] problems. For example one might choose to use a uniform partition on the $[0, 1]$ interval depending on the number of points. This method is considered obsolete. Another way is to use arc length approximation. A suitable one for our case is the chord length parameterization. We start from 0 and then each subsequent knot is given by the Euclidean distance between the points. When the distance between the points is too big this can lead to incorrect parameterization. Generally the problem with fitting algorithms is that there might exist another basis (i.e. another knot sequence) which leads to better approximation.

Because of the amount of data for each contour we believe that the method proposed by Lyche [1, 2] is a suitable one. First we interpolate the points of the contour using a cubic B-spline. Then we apply the knot removal algorithm given in [1]. The idea is that from the original set of points only a few are necessary to describe satisfactorily the underlying geometry. Taking into account that even the initial set is an approximation of the object boundary, we start with this initial approximation which is easy to compute, but it might contain excessive points or noise, and we perform data reduction and smoothing.

Using the chord length parameterization we obtain the starting knot sequence. Because the points are very close to each other essentially this is equivalent to the arc length. Also we are guaranteed that we have a dense enough sequence of knots that the "optimal" solution is included to this knot vector.

5.2.2 Knot Vector Creation and Interpolation

We start with the set of points $\mathbf{P}_0, \ldots, \mathbf{P}_{n-1}$. Because the contour we want to approximate is a closed line we can extend this set as follows:

$$\mathbf{P}_n := \mathbf{P}_0.$$

Then we can define a knot vector $\mathbf{a}$ as:

$$\begin{aligned} a_0 &= 0, \\ a_i &= \| \mathbf{P}_i - \mathbf{P}_{i-1} \|_{L_2},\ i = 1, \ldots, n. \end{aligned}$$

We normalize $\mathbf{a}$ in $[0, 1]$,

$$a_i \leftarrow \frac{a_i}{a_n}$$

In order to produce a B-spline basis that spans all the splines on the knot vector $\mathbf{a}$ we have to create a new vector $\mathbf{t}$. Since we are dealing with periodic splines the knot vector $\mathbf{t}$ is given by the following relations [26](for cubic splines $k = 3$):

$$\begin{aligned} \mathbf{t} &:= (t_0, \ldots, t_{n+6}) \\ t_i &= a_0 - (a_n - a_{i+n-3}),\ i = 0, 1, 2, \\ t_{i+3} &= a_i,\ i = 0, \ldots, N, \\ t_{i+n+4} &= a_n + (a_{i+1} - a_0),\ i = 0, 1, 2. \end{aligned}$$

We can write $\mathbf{f}$ as:

$$\mathbf{f}(t) = \sum_{i=0}^{n+2} \mathbf{d}_i B_{i,\mathbf{t}}(t).$$

where we have suppressed the subscript denoting the degree of the B-spline. To interpolate the given points we form the following n equations:

$$\sum_{i=0}^{n+2} \mathbf{d}_i B_{i,\mathbf{t}}(t_j) = \mathbf{P}_j, j = 0, \ldots, n-1. \tag{16}$$

An inconsistency appears to exist between the number of equations and the number of unknowns since we have n equations and $n + 3$ unknowns. However, because the curve is closed the restrictions in continuity

and of the fact that $\mathbf{P}_n = \mathbf{P}_0$ lead to:

$$\begin{aligned} \mathbf{d}_n &= \mathbf{d}_0, \\ \mathbf{d}_{n+1} &= \mathbf{d}_1, \\ \mathbf{d}_{n+2} &= \mathbf{d}_2. \end{aligned}$$

Because of the locality property of the B-splines the resulting matrix is tridiagonal and can be solved with $O(n)$ operations. Of course we have to perform this operation twice, for the x- and y-direction. We have used the code described in [32].

5.2.3 Knot Removal

Given n points representing a closed contour we have computed the interpolating spline $\mathbf{f}$. The data necessary to define this spline are its degree k, its knot vector $\mathbf{t}$ and the coefficients matrix $\mathbf{d}_i$. We denote by $S_{k,\mathbf{t}}$ the set of all splines of degree k on the knot vector $\mathbf{t}$. So if $\mathbf{t} = (t_0, \ldots, t_{n+2k})$ then $S_{k,\mathbf{t}}$ is an $(n+k)$[17] dimensional space of functions given by

$$S_{k,\mathbf{t}} = span\{B_{0,k,\mathbf{t}}, \ldots, B_{n+k-1,k,\mathbf{t}}\}.$$

Note that if $\boldsymbol{\tau} \subseteq \mathbf{t}$ then $S_{k,\boldsymbol{\tau}} \subseteq S_{k,\mathbf{t}}$. As we show before we are interested in the restriction of the elements of $S_{k,\mathbf{t}}$ to the interval $[t_k, t_{n+k}] \equiv [0,1]$. The knots $t_{k+1}, \ldots, t_{n+k-1}$ are called *interior knots*, and these are the only knots we try to remove. In [1, 2] they deal with open curves. We had to slightly modify their algorithm for closed curves. For closed curves each time we remove knots we have to modify the extended set, i.e. $t_0, \ldots, t_{k-1}$ and $t_{n+k+1}, \ldots, t_{n+2k}$ in order to assure that the new curve is C^2 at the end points.

Suppose a norm $\|\cdot\|$ on $S_{k,\mathbf{t}}$ has been chosen. We want to compute a subspace $S_{k,\boldsymbol{\tau}}$ of lower dimension and an element $\mathbf{g}$ of $S_{k,\boldsymbol{\tau}}$ such that $\|\mathbf{f}-\mathbf{g}\| < \epsilon$. In general, it is not necessary to produce the lowest possible dimension. The aim is to reduce the given vector up to an error within reasonable computational time.

The algorithm we are using involves four main components. We refer to these as *Weight*, *Rank*, *Remove*, and *Approximate*. In the first step we are computing the weights of the knots. These numbers indicate the significance of a knot in specifying the spline shape. The second step is to rank these knots. The third step is, given a specified number of knots, to remove the least important. In the final step, for the given reduced knot vector we compute a spline $\mathbf{g} = G(\mathbf{t};\boldsymbol{\tau})\mathbf{f}$ on this vector which minimizes the "distance" (since each spline can be considered as a vector) from the original spline. With G we denote the approximation, in our case this is a least squares approximation.

When a vector $\boldsymbol{\tau}$ and a spline $\mathbf{g}$ is computed we may be able to continue the reduction. If $\boldsymbol{\tau} = \mathbf{t}$ it means that all the knots are important within the specified error. If $\boldsymbol{\tau}$ contains no interior knots this means that all the knots were removed. In practice we will have some interior knots. We can try to improve our result by removing more knots. Here is a schematic algorithm:

1. $\boldsymbol{\tau}^0 = \mathbf{t}; \mathbf{g}^0 = \mathbf{f}$; (Initialize)

[17] I might appear that there is an inconsistency between the dimension given in (Sec. 5.1.1) and this one. Because we work with a closed curve of n points is like having $n+1$ points since the one more point (the end of the curve) is implicitly given. Thus the dimension is $(n+1)+k-1 = n+k$.

2. **for** $l = 0, 1, 2, \ldots$

    1. **if** $\boldsymbol{\tau}$ has no interior knots **then stop**;
    2. **Compute weights**; (Weight)
    3. **Rank knots**; (Rank)
    4. **Binary Search on** r
        1. **Initialize** r; (Number of knots to remove)
        2. $\boldsymbol{\tau} = \boldsymbol{\tau}^l / - r\ knots$; (Remove)
        3. $\mathbf{g} = G(\mathbf{t}; \boldsymbol{\tau})\mathbf{f}$; (Approximate)
        4. $error = \| \mathbf{f} - \mathbf{g} \|$; (Compute Error)
        5. **if** $error < \epsilon$ **increase** r;
        6. **else decrease** r;
    5. **if** $r = 0$ **stop**;
    6. $\boldsymbol{\tau}^{l+1} = \boldsymbol{\tau}$;
    7. $\mathbf{g}^{l+1} = \mathbf{g}$;

The algorithm generates a sequence $(S_{k,\boldsymbol{\tau}^j})$ of shrinking subspaces of $S_{k,\mathbf{t}}$ and a sequence of approximations $(\mathbf{g}^j)$ such that:

$$\boldsymbol{\tau}^{j+1} \subset \boldsymbol{\tau}^j.$$
$$\left\| \mathbf{f} - \mathbf{g}^j \right\| \leq \epsilon$$

In practice the majority of the knots are removed in the first few iterations. We now present some important details of the algorithm.

Weight. Let w_j denote the weight corresponding to knot $\boldsymbol{\tau}_j^l$ and $S_{j,\boldsymbol{\tau}}$ denote the subspace of spline functions on the knot vector $\boldsymbol{\tau} = \boldsymbol{\tau}^l - \tau_j^l = (\tau_0^l, \ldots, \tau_{j_1}^l, \tau_{j+1}^l, \ldots)$ The basic idea is to let w_j be an approximation to dist$(\mathbf{f}, S_{j,\boldsymbol{\tau}})$ in some norm. In this implementation we are using the *maximum* norm. For simplicity in the notation we will show the method for $l = 0$ so that $\boldsymbol{\tau}^l \equiv \mathbf{t}$ and $\mathbf{g}^l \equiv \mathbf{f}$. It is understood that the same procedure is applied for $(l = 1, 2, \ldots)$. We also suppress subscript k. We compute the weight of knot t_j and we set

$$\begin{aligned} w_j &= \|\mathbf{f} - \mathbf{g}\|_{\ell^\infty, \mathbf{t}} \text{ where} \\ \mathbf{g} &= \min_{\mathbf{g} \in S_{j,\boldsymbol{\tau}}} \|\mathbf{f} - \mathbf{g}\| \end{aligned} \tag{17}$$

Because $\mathbf{g} \in S_{\boldsymbol{\tau}}$ and $\mathbf{f} \in S_{\mathbf{t}}$ we extend $\mathbf{g}$ in $S_{\mathbf{t}}$

$$\mathbf{g} = \sum_i^m \mathbf{c}_i B_{i,\boldsymbol{\tau}} \in S_{\boldsymbol{\tau}}$$

where $m = n + k - 2$ for $l = 0$. Since (See Sec. 5.1.7) $S_{\boldsymbol{\tau}} \subset S_{\mathbf{t}}$, we have

$$\mathbf{g} = \sum_i^{m+1} \hat{\mathbf{c}}_i B_{i,\mathbf{t}}$$

where (See [1] and Sec. 5.1.7)

$$\hat{\mathbf{c}} \equiv (\hat{\mathbf{c}^0}, \hat{\mathbf{c}^1}, \hat{\mathbf{c}^2}) = \begin{cases} \mathbf{c}_i, & \text{for } i = 0, 1, \ldots, j-k-1 \\ \mu_1 \mathbf{c}_i + \lambda_i \mathbf{c}_{i-1}, & \text{for } i = j-k, \ldots, j-1 \\ \mathbf{c}_{i-1}, & \text{for } i = j, \ldots, n+1 \end{cases}$$

and

$$\lambda_i = \frac{\tau_{i+k+1} - t_j}{\tau_{i+k+1} - \tau_i},\ \mu_i = \frac{t_j - \tau_i}{\tau_{i+k-1} - \tau_i}$$

So if $\mathbf{c} \equiv (\mathbf{c}^0, \mathbf{c}^1, \mathbf{c}^2)$, $\hat{\mathbf{c}} \equiv (\hat{\mathbf{c}}^0, \hat{\mathbf{c}}^1, \hat{\mathbf{c}}^2)$ and $\mathbf{d} \equiv (\mathbf{d}^0, \mathbf{d}^1, \mathbf{d}^2)$ (Eq. 16) with

$$\begin{aligned} \mathbf{c}^0 &= (\mathbf{c}_0, \ldots, \mathbf{c}_{j-k-2})^T \\ \mathbf{c}^1 &= (\mathbf{c}_{j-k-1}, \ldots, \mathbf{c}_{j-1})^T \\ \mathbf{c}^2 &= (\mathbf{c}_j, \ldots, \mathbf{c}_m)^T \\ \hat{\mathbf{c}}^0 &= (\hat{\mathbf{c}}_0, \ldots, \hat{\mathbf{c}}_{j-k-1})^T \\ \hat{\mathbf{c}}^1 &= (\hat{\mathbf{c}}_{j-k-1}, \ldots, \hat{\mathbf{c}}_j)^T \\ \hat{\mathbf{c}}^2 &= (\hat{\mathbf{c}}_{j+1}, \ldots, \hat{\mathbf{c}}_{m+1})^T \\ \mathbf{d}^0 &= (\mathbf{d}_0, \ldots, \mathbf{d}_{j-k-2})^T \\ \mathbf{d}^1 &= (\mathbf{d}_{j-k-1}, \ldots, \mathbf{d}_j)^T \\ \mathbf{d}^2 &= (\mathbf{d}_{j+1}, \ldots, \mathbf{d}_{m+1})^T \end{aligned}$$

The solution for $\hat{ci}$ is not unique since we are solving an $\ell^i nfty$ problem. We can obtain a solution for (Eq. 17) by setting

$$\begin{aligned} \mathbf{c}^0 &= \hat{\mathbf{c}}^0 = \mathbf{d}^0 \\ \mathbf{c}^2 &= \hat{\mathbf{c}}^2 = \mathbf{d}^2 \\ \text{For } \mathbf{c}^1 \text{ solve } &\min_{\mathbf{c}^1} \|\mathbf{d}^1 - \mathbf{B}\mathbf{c}^1\|_{\ell^\infty} \end{aligned} \tag{18}$$

where $\hat{\mathbf{c}}^1 = \mathbf{B}\mathbf{c}^1$ and

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ \lambda_{j-k} & \mu_{j-k} & \cdots & 0 & 0 \\ 0 & \lambda_{j-k-1} & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \mu_{j-2} & 0 \\ 0 & 0 & \cdots & \lambda_{j-1} & \mu_{j-1} \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

is a matrix with $k+2$ rows and $k+1$. A specific method for solving (Eq. 18) is presented in [1]. For a short description see Appendix 9.1. It can be solved by inverting a 5×5 matrix for each knot. Finally we get

$$\mathbf{w} = \|\mathbf{d} - \mathbf{B}\mathbf{c}\|_{\ell^\infty}$$

Rank. After we have computed the weights we have to rank the knots. For each knot t_j we compute each ranking number v_j which is an integer as follows:

$$\begin{aligned} v^1 &= \lfloor \frac{\ln \mathbf{w}_j^1 - \ln \epsilon}{\ln 2 + 1} \rfloor \\ v^2 &= \lfloor \frac{\ln \mathbf{w}_j^2 - \ln \epsilon}{\ln 2 + 1} \rfloor \\ v_j &= \max(v^1, v^2) \end{aligned}$$

and then we sort v_j in a nondecreasing order.

Remove. If we are to remove r knots from $\mathbf{t}$, then we remove all the knots whose rank is *strictly less* than v_r. Because many knots might have equal ranks it is possible that the number of knots we removed is less than r. To choose the remaining, say p knots to be removed, let $u_0 \leq \cdots \leq u_q$ be the knots with ranking numbers equal to v_r listed in the order in which the occur in $\mathbf{t}$. We remove $(u_{d_0}, \ldots, u_{d_{p-1}})$, where

$$d_i = \lfloor \frac{(r+1)(i+0.5)}{p} + 1 \rfloor - 1$$

that is we remove the knots uniformly on subscripts.

Approximate This problem is similar to the one which occurred in the "weight" step but in more general form since we are removing more than one knot. However, here we use best approximation in the ℓ^2, $\mathbf{t}$-norm. We want to compute $\mathbf{g}$ given $\mathbf{f}$, $\boldsymbol{\tau}$, $\mathbf{t}$ and k such that:

$$\begin{array}{c} \boldsymbol{\tau} \subset \mathbf{t} \\ \mathbf{g} = \sum_{i=0}^{N} \mathbf{c}_j B_{i,k,\boldsymbol{\tau}} = \sum_{i=0}^{M} \mathbf{q}_i B_{i,k,\mathbf{t}} \\ \mathbf{f} = \sum_{i=0}^{M} \mathbf{d}_j B_{i,k,\mathbf{t}} \\ \min_{\mathbf{g}} \|\mathbf{f} - \mathbf{g}\|_{\ell^2} \end{array}$$

were the norm is explained in (Sec. 5.1.3, Eq. 9, 10). The coefficients $\mathbf{q_i}$ which define $\mathbf{g}$ on $S_{\mathbf{t}}$ are computed with the discrete splines insertion matrix $\mathbf{B}$ which we compute with the Oslo algorithm. So we end up with a usual algebraic least squares problem of the form:

$$\min_{\mathbf{c}} \left\| \boldsymbol{E}_{\mathbf{t}}^{1/2} (\mathbf{B}\mathbf{c} - \mathbf{d}) \right\|_{\ell^2}$$

It is shown [1] that it is advantageous, with respect the condition number of the normal equations of this least squares problem, to solve the following for $\mathbf{x}$,

$$\min_{\mathbf{x}} \left\| \boldsymbol{E}_{\mathbf{t}}^{1/2} (\mathbf{B}\boldsymbol{E}_{\boldsymbol{\tau}}^{-1/2}\mathbf{x} - \mathbf{d}) \right\|_{2}$$

and then $\mathbf{c} = \boldsymbol{E}_{\boldsymbol{\tau}}^{-1/2}\mathbf{x}$. The condition number of the above equation depends only on k and is small for low values of k [23, 1, 25]. A discrete norm is used because it is simple and fast to work with, and the ℓ^2, $\mathbf{t}$-norm because it gives rise to linear system of equations, and it approximates L^2 well. However it does not give any pointwise information about the error. A solution to this problem is the folowing: when measuring the error in order to decide how many knots to remove, ℓ^∞ is used. We used both norms and did not observe any noticeable difference. In any case the user can choose the norm in which error is measured.

6 Surface Skinning

Surface skinning is a process of passing a smooth surface through a set of curves [28]. Usually these curves represent cross-sections of an object so they often mentioned as cross-sectional curves. Using the B-splines representation, it is possible to "force" a surface to assume almost any curve as an parametric curve be it a

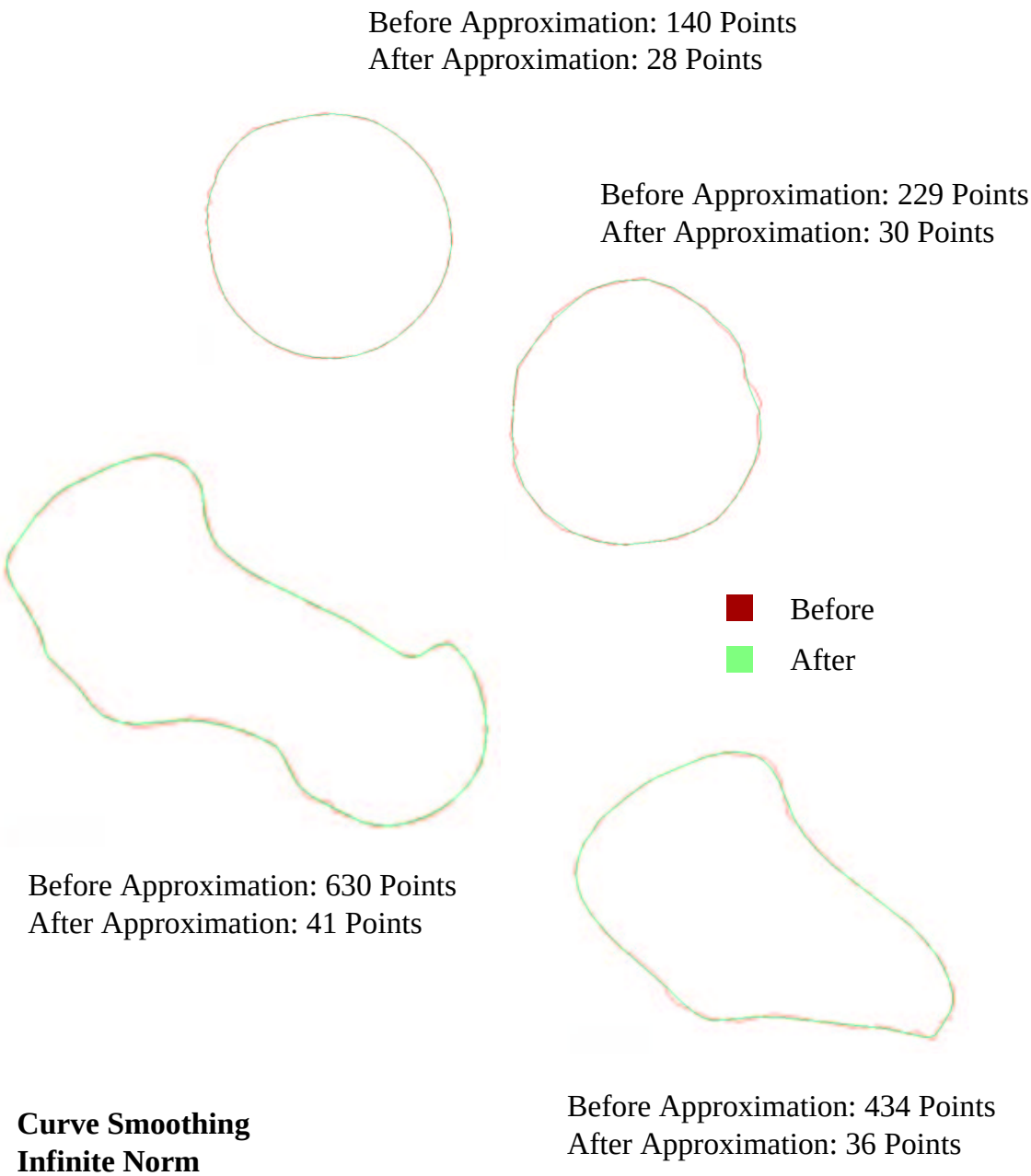


Figure 9: Results of the curve smoothing procedure

piecewise linear curve, conic section or a high order spline. However, in order to skin across the curves of various types, they all have to be made compatible, i.e. the have to have the same degree and to be defined over the same knot vector. The equation of a tensor product b-spline surface is given by

$$\mathbf{F}(u,v) = \sum_i \sum_j \mathbf{c}_{i,j} B_{i,k,\mathbf{u}}(u) B_{j,l,\mathbf{v}}(v) = \mathbf{b}_{k,\mathbf{u}}^T(u) \mathbf{C} \mathbf{b}_{l,\mathbf{v}}(v). \tag{19}$$

In our problem u corresponds to t-variable of each contour and v represent the z-axis of the CT-scan. Therefore if we hold v fixed, we recover the contour corresponding to the specific z. Recall that $v \in [0,1]$. The compatibility requirement is now apparent. Each curve has to use the same $B_{k,\mathbf{u}}$ functions, thus k and $\mathbf{u}$ must be the same for all curves. We can create a knot vector that is the union of all curves' knot vectors and then, by using discrete B-splines, we can compute the coefficients of each curve in the new extended spline space. However, this procedure is inefficient. If we have 40 curves with 20 knots each, the latter technique would create $40 \times 20 \times 40$ control points. It would essentially cancel the previous knot removal operation, and thus we follow a different approach. We divide the $[0,1]$ into $n-1$ intervals and we take the geometric average of the knots that lie within each interval. The result of this process is a set of n knots where n is specified by the user. So comparing with the aforementioned example this average leads to a *approximate* skinning of the surface by using only $n \times 40$ control points. In practice $30-40$ knots where enough to reproduce the curves from the curve approximation.

The next step is to interpolate the curves on this new vector. Let

$$\mathbf{u},\ (u_0, \ldots, u_k = 0, \ldots, u_{n+k} = 1, \ldots, u_{n+2k})$$

be the union knot vector. The vertical direction v-vector is created in th usual way of chord length parameterization [28, 27]. For each curve j we compute n points as follows:

$$\mathbf{P}_{ij} = \sum_l \mathbf{d}_{lj} B_{l,k,\mathbf{u}}(u_i)\ , i = k, \ldots, n+k.$$

where $\mathbf{d}_j$ are the coefficients of the jth curve on the $S_{k,\mathbf{u}}$. Then we solve the following linear system [26, 28] for $\mathbf{C}$:

$$\mathbf{P}_{i,j} = \mathbf{b}_{k,\mathbf{u}}^T(u_i) \mathbf{C} \mathbf{b}_{l,\mathbf{v}}(v_j).$$

The number of operations to solve this system if $O(n \times q)$ where q is the number of curves. The created B-spline surface is output in the IGES 5.2 formatted file.

Our implementation provides the user with the ability to choose which curves he/she wants to skin. This represents a crude way of data reduction over the v direction. Moreover one can create multiple files representing different branches of a complicated structure. However, these surfaces will be disjoint and further manipulation must take place to put them together. We used PATRAN's solid modeling features to complete this task.

7 Femur Pre-Processing

We applied the described methodologies on CT-scans from patients and cadavers in order to create geometric models for the femur and the acetabulum. For the femur we obtain a representation of the surface by using

1200 control points, 40 knots for each curve and 30 curves out of 104. The original data file from the AVS extraction procedure contains 60,000-100,000 points while the final file has around 1000-2000 points.

During the THA operation the surgeon has to cut the femoral head and drill a hole into the femur in order to insert the femoral part of the implant. Therefore, further processing of the femur surface is needed in order to create the final geometry. Our first approach was to use solid boolean operations between the (triangulated) femur volume and auxiliary volumes representing the volume of the material to be removed. This technique worked but the resulting surfaces were very badly shaped and it was impossible to create a mesh. The problem can be simplified by splitting this procedure into two parts. The first is the cutting of the femoral head and the second is the drilling of the intramedullary canal in order to create the cavity in which the implant sits. We believe that it is possible to resolve these problems with of-the-shelf algorithms.

The femoral head removal can be simulated as follows. Using 3 points we can describe mathematically the plane of the cutting. Then using a surface/plane intersection algorithm we can create a curve which represents the boundary of the final top surface. It is trivial then, to skin this surface with the remaining set of curves and create the final outer surface. Another approach which currently explore by the MRCAS group is to rotate the CT-scan data and create the contours in planes parallel to the cutting plane and then just discard the upper part of the femur.

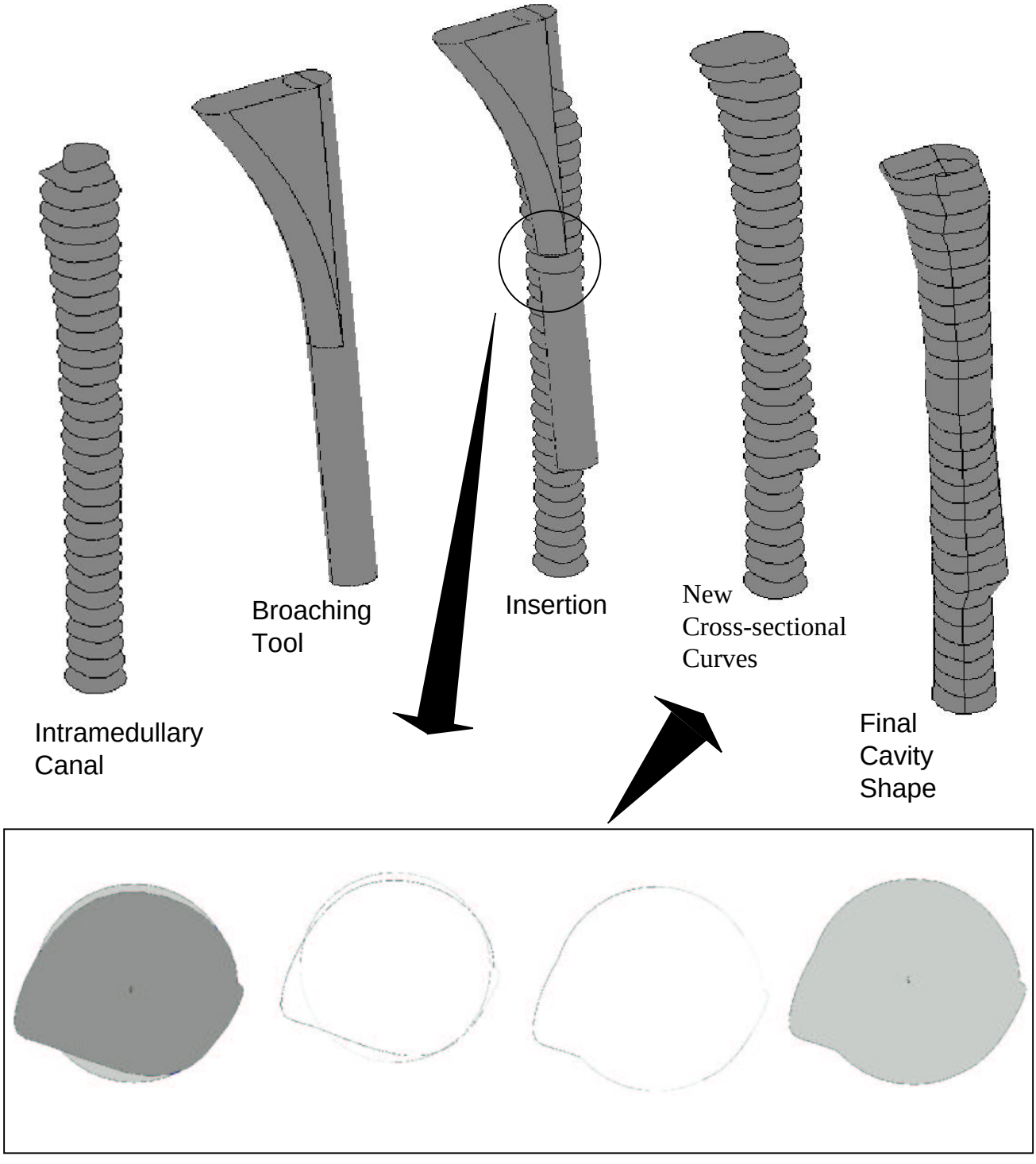
The creation of the femoral cavity is more tedious to achieve. Computationally, is very difficult to perform surface/surface Boolean operation since the two surfaces are overlapping (Sec. 5.1). The solution is to use the plane/surface intersection algorithm we mentioned before and create cross-sectional curves on the same z-planes as of the CT-scan and then perform boolean operations on the two curves set.

We have written a C program which runs in real time within an ANSYS session in order create the the blended femoral cavity (Fig. 10). We also applied these ideas using PATRAN solid modeling modules and the results are very satisfactory. Furthermore because the resulting surfaces are the conformal mapping of a rectangular domain they can be meshed using quadrilateral elements, which is important for the contact mechanics analysis. We also managed to mesh the resulting volumes in $O(hour)$ time.

8 Future Work

There are many issues that have to be addressed in order to complete the patient specific pre-operative analysis. The surface reconstruction of the femur still needs considerable user intervention. The algorithms of surface/plane and curve/curve intersection must be integrated into our program.

Another important issue is the variation of the material properties. Cortical bone can with reasonable accuracy, be modeled as linearly homogeneous isotropic elastic material. In the contrary trabecular bone is in-homogeneously anisotropic, admits large strains and plastifies. Several ideas have been explored in order to distinguish between cortical and trabecular bone. One is to represent these two regions as different volumes. The other one is to allow material properties vary among and within elements. The latter requires the development of an interface between CT-Scan data and the mesh generator. The former requires extra work for the solid modeling.



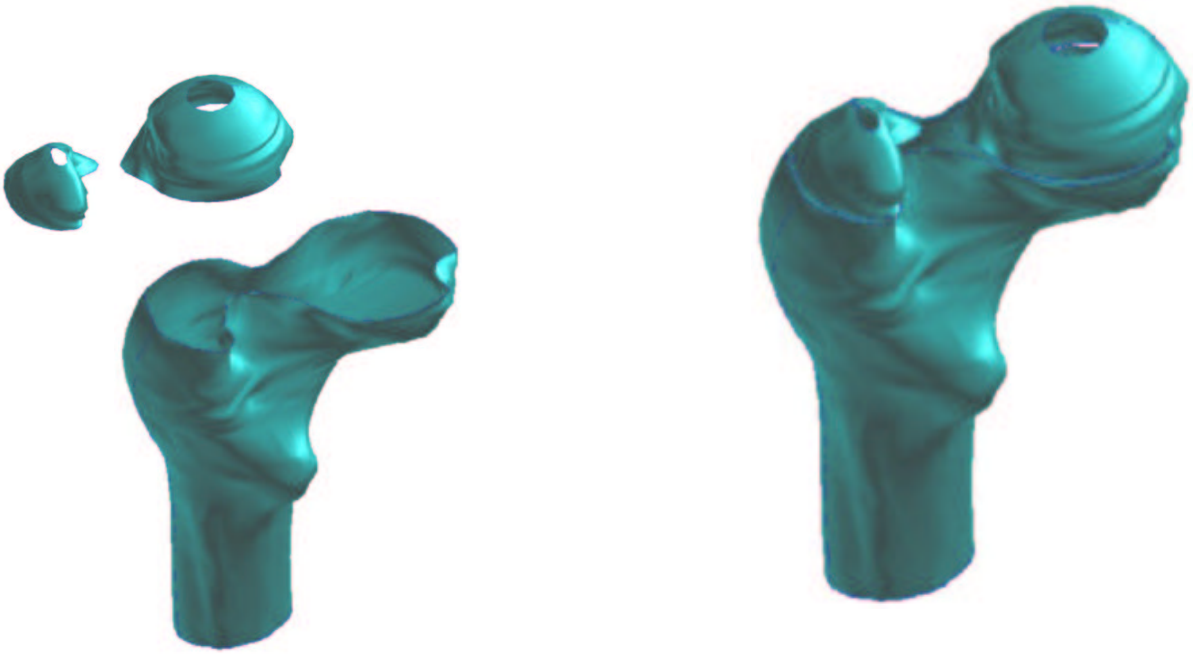
Intramedullary canal preparation for implant insertion.

Figure 10: Preparation of femur intramedullary canal using lower level (area instead of volume) Boolean operations

CT-scan Data: 104 Slices, 40000 Points.

Approximated Data: 104 Slices, 4000 Points

Femur Modeling: Surface Reconstruction



Skinned Surface: 1800 Control Points.

Figure 11: Femur skinning.

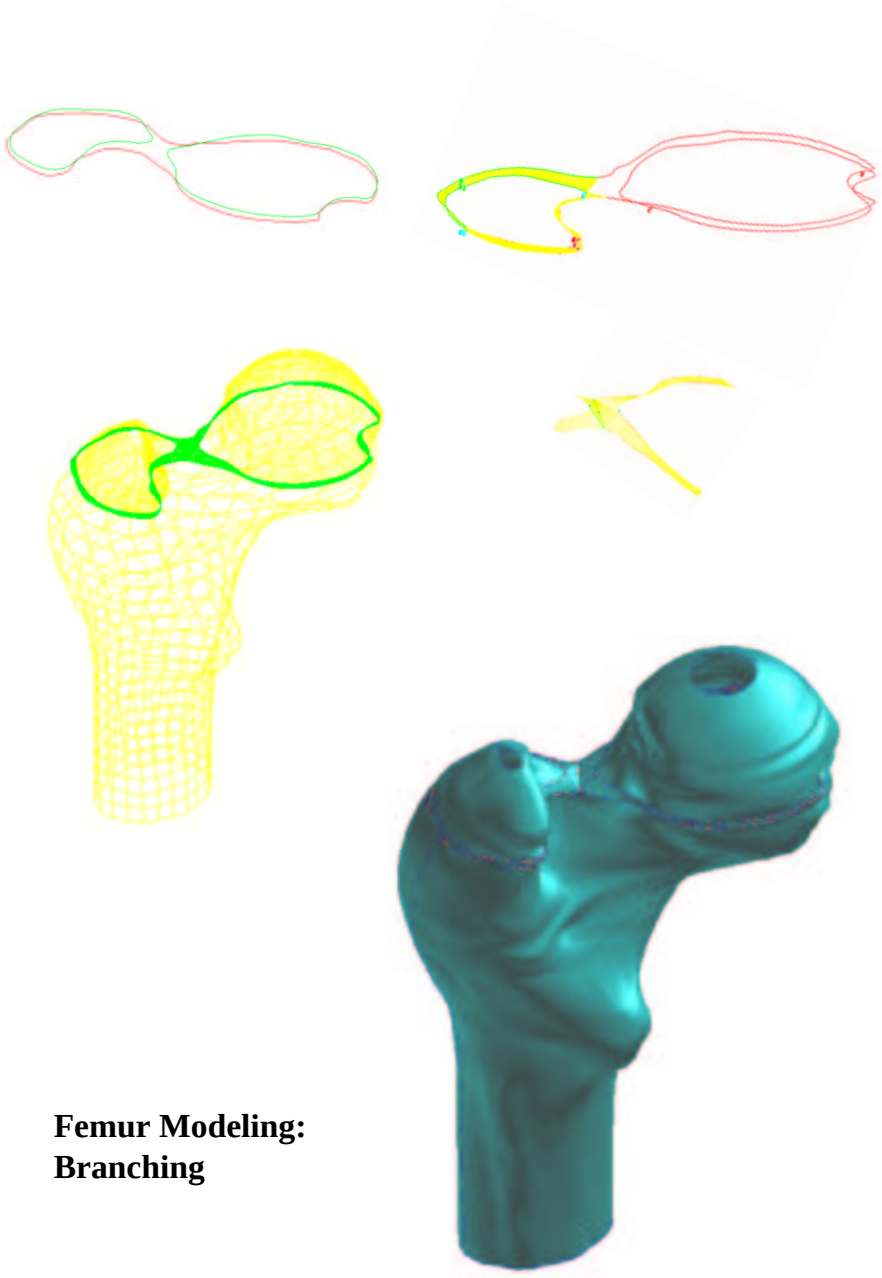


Figure 12: Femur branching

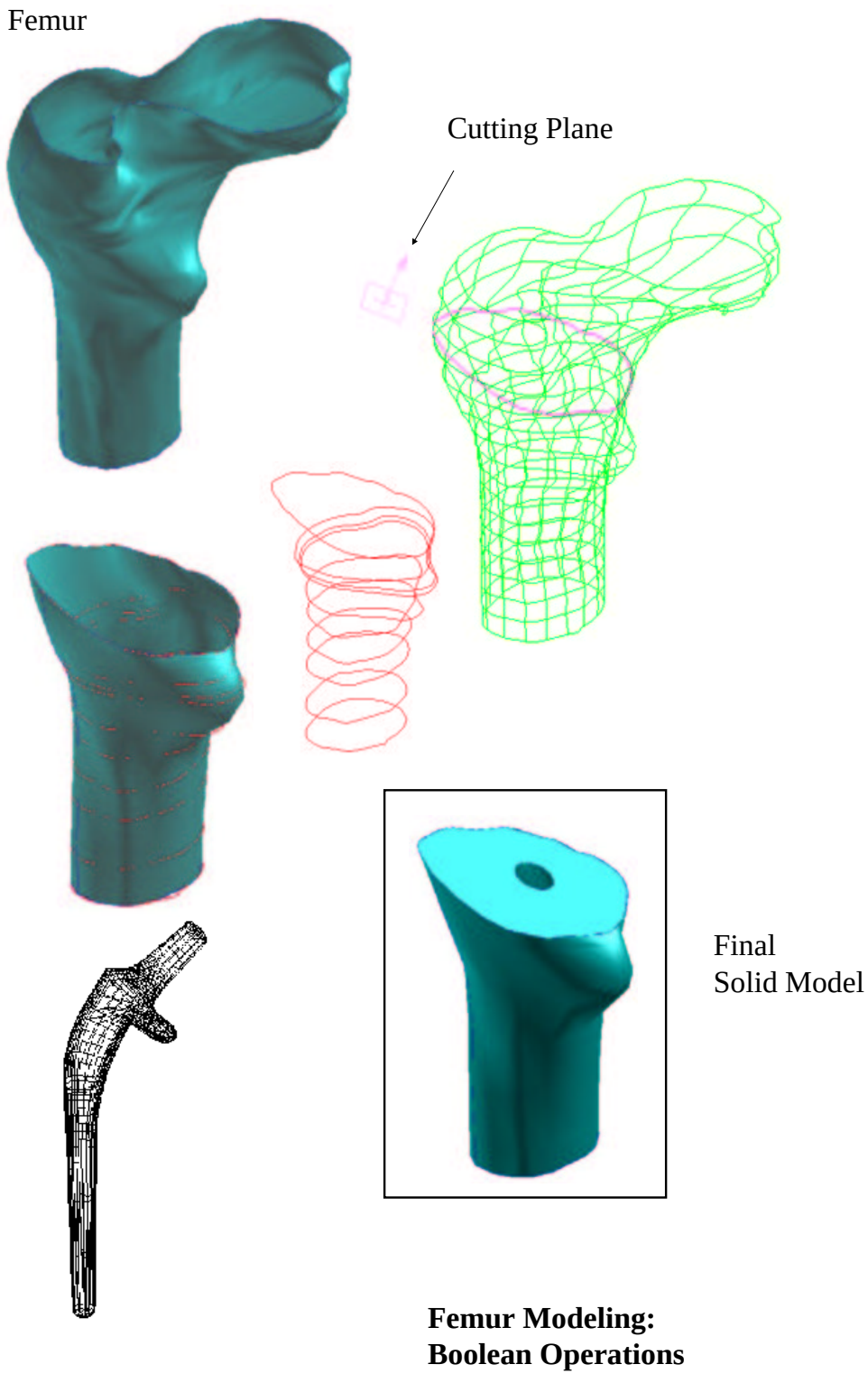
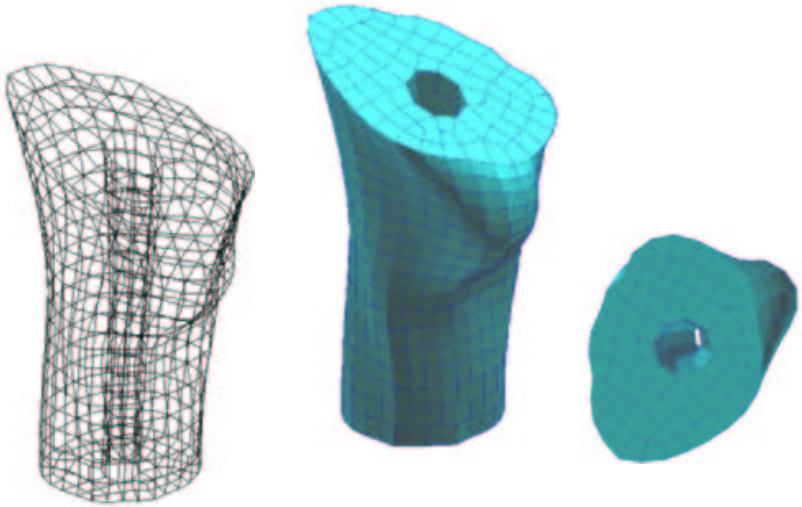
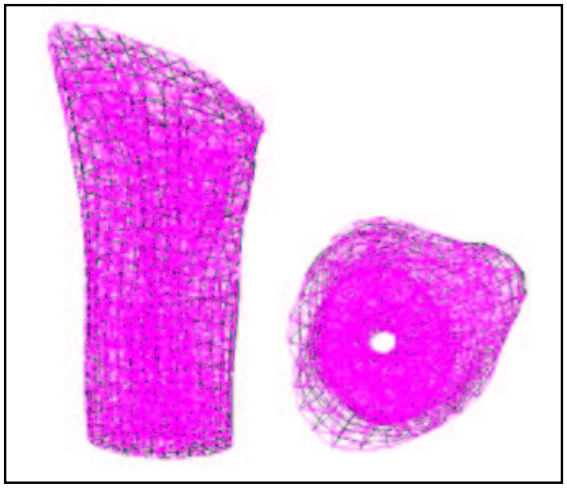


Figure 13: Boolean operations on femur in order to produce the final model

Surface Mesh



Femur Modeling: Meshing

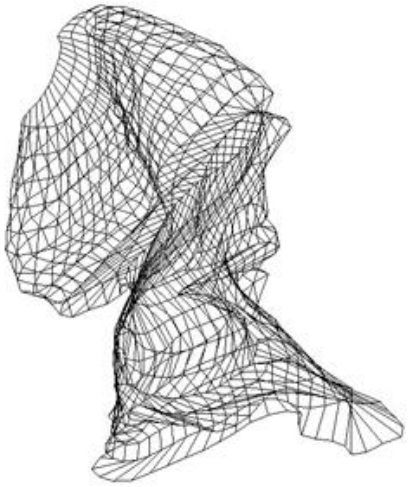


Final Volume Mesh: 4649 Elements

Figure 14: Computational mesh of the femur.

Smoothing

Skinning



Meshing

Pelvis Modeling

Figure 15: Application of the method on pelvis.

Meshing of the resulting surface is performed by commercial software as we mentioned before. In the case of femur, one can take advantage of the cylindrical topology. We are developing a module that uses conformal mapping to create a mesh on each curve plane using quadrilaterals and then stack the elements together.

The goal of the research undertaken by our team is to improve post-operative quality of life of the patient via a pre-operative analysis. We hope that with this work we have make a small step toward this goal.

9 Appendix

9.1 Computing the Weights

In this section we consider the problem

$$\min_{\mathbf{c}} \|\mathbf{Bc} - \mathbf{d}\|_{\ell^\infty}$$

where $\mathbf{d} \in \Re^n$, $\mathbf{c} \in \Re^{n-1}$ and the matrix $\mathbf{B} \in \Re^{n\times(n-1)}$ has the special form

$$\mathbf{B} = \begin{bmatrix} \mu_0 & 0 & \cdots & 0 & 0 \\ \lambda_1 & \mu_1 & \cdots & 0 & 0 \\ 0 & \lambda_2 & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \mu_{n-3} & 0 \\ 0 & 0 & \cdots & \lambda_{n-2} & \mu_{n-2} \\ 0 & 0 & \cdots & 0 & \lambda_{n-1} \end{bmatrix}$$

The solution of this problem can be found [1] by solving a linear system of n equations in n unknowns[18]

$$\mathbf{Ax} = \mathbf{d}$$

where for the matrix $\mathbf{A}$ takes the form

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & (-1)^{n-1} \\ \lambda_{j-k} & \mu_{j-k} & \cdots & 0 & 0 & (-1)^{n-2} \\ 0 & \lambda_{j-k-1} & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \mu_{j-2} & 0 & 1 \\ 0 & 0 & \cdots & \lambda_{j-1} & \mu_{j-1} & -1 \\ 0 & 0 & \cdots & 0 & 1 & 1 \end{bmatrix}$$

If $\mathbf{x} = (x_0, \ldots, x_{n-1})^T$ solves $\mathbf{Ax} = \mathbf{d}$ then the solution $\mathbf{c}$ is given by $c_j = x_j$ for $j = 0, 1, \ldots, n-2$.

9.2 Nuages Input File

Assuming that the CT-scan runs on the z-axis and the image produced on the xy-plane The structure of the input file for nuages is the following: `s` *integer1*

[18] In our case $n = 5$

v *integer2* z *floatz*
{
floatx floaty
⋮
}
v *integer2* z *floatz*
{
float x float y
⋮
}
where *integer1* denotes number of slices and *integer2* denotes number of points to each slice. Different contours are included in curly brackets within each slice.

References

[1] T. Lyche and K. Mørken. A data reduction strategy for splines. *Research Report no. 107,* Institute of Informatics, University of Oslo, February 1987.

[2] T. Lyche and K. Mørken. Knot Removal for Parametric B-spline Curves and Surfaces. *Research Report no. 109,* Institute of Informatics, University of Oslo, March 1987.

[3] T. Lyche and Richard Riesenfeld. Discrete B-Splines and Subdivision Techniques in Computer-Aided Geometric Design and Computer Graphics. *Computer Graphics and Image Processing,* Vol. 14, 87-111, 1980.

[4] Biegler et al. Effect of Porous Coating and Loading Conditions on total hip femoral stem stability. *The Journal of Arthoplasty.* Vol. 10 No.6 1995, 839-847.

[5] Christoph M. Hoffman and George Veněček, Jr. Fundamental Techniques for Geometric and Solid Modeling. *CSD-TR-91-044, Technical Report*, Purdue University, Department of Computer Science, 1991.

[6] H. J. Samet. Design and Analysis of Spatial Data Structures: Quadtrees, Octrees, and other Hierarchical Methods. *Addison-Wesley, Redding, MA, 1989.*

[7] G. J. Jense. Voxel-based methods for CAD *Computer Graphics Forum*, Vol. 13, 238-243, 1992.

[8] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface reconstruction algorithm. *Computer Graphics*, Vol. 21(4),163-169, July 1987.

[9] J. H. Keyak et al. Three-Dimensional Finite Element Modeling of the Distal Radius: A new method to study fracture. *Transactions ,42nd Meeting of The Orthopedic Research Society,* Vol. 21, Sec 2, February 1996.

[10] Chandrajit L. Bajaj et al. Surface and 3D Triangular Meshes from Planar Cross Sections. *CSD-TR-96-002, Technical Report,* Purdue University, Department of Computer Science, 1996.

[11] Chandrajit L. Bajaj et al. Automatic Reconstruction of Surfaces and Scalar Fields from 3D Scans. *CSD-TR-94-0001, Technical Report*, Purdue University, Department of Computer Science, 1994.

[12] Insung Ihm and Chandrajit L. Bajaj. C^1 Smoothing of Polyhedra with Implicit Algebraic Surface Patches. *CSD-TR-91-060, Technical Report*, Purdue University, Department of Computer Science, 1991.

[13] Frey et all. Fully Automatic Mesh Generation for 3-D Domains Based upon Voxel sets. *International Journal for Numerical Methods in Engineering.* Vol 37, 2735-2753, 1994.

[14] H.N. Christainsen and T. W. Sederberg. Conversion of complex contour line definition into polygonal element mosaics. *Computer Graphics*, Vol.12, Iss.3, 187-192, 1978.

[15] A. B. Ecoule et all. A triangulation algorithm from arbitrary shaped multiple planar contours. *ACM Transactions on Graphics*, Vol. 10, Iss. 2, 182-199, 1991.

[16] E. Keppel. Approximating complex surfaces by triangulation of contour lines. *IBM Journal of Research Developments*, Vol. 19, 2-11, 1975

[17] J. D. Boissonat. Shape reconstruction from planar cross sections. *Computer Vision, Graphics and Image Processing*, Vol. 44, 1-29, 1988.

[18] R. H. J. Gmelig Meyling and P. R. Pfluger. B-Spline Approximation of a Closed Surface. *IMA Journal of Numerical Analysis*, Vol. 7, 73-96,1987.

[19] J. C. Jimenez, R. Biscay and E. Aubert Parametric Representation of Anatomical Structures of the Human Body by Means of Trigonometric Interpolating Sums. *Journal of Computational Physics*, Vol. 126, 243-250, 1996.

[20] Miguel Angel García. Efficient Surface Reconstruction From Scattered Points through Geometric Data Fusion. *IEEE MFI 1994 Proceedings, Las Vegas*, 559-566, Oct. 1994.

[21] Philip Fong and Hans-Peter Seidel. Control Points for Multivariate B-Spline Surfaces over Arbitrary Triangulations. *Computer Graphics Forum*, Vol. 10, 309-317, 1991.

[22] Jai P. Menon. Constructive Shell Representations For Free-Form Surfaces and Solids. *RC-18450, IBM research Report*, IBM T.J.Watson Research Center, October 1992.

[23] C. de Boor. A Practical Guide to Splines. *Springer-Verlag*, New York 1978.

[24] M. G. Cox. Strategies for knot placement in least squares data fitting by splines. *NPL report, DITC 101/87*,National Physical Laboratory (UK),1987.

[25] Larry L.Schumaker. Spline Functions: Basic Theory. *Wiley and Sons*, New York, 1981.

[26] Fujio Yamaguchi. Curves and Surfaces in Computer Aided Geometric Design. *Springer-Verlag*, Berlin, 1988.

[27] Paul Dierckx. Curve and Surface Fitting with Splines. *Clarendon Press*, Oxford, 1993.

[28] I. Faux and M. Pratt. Computational Geometry for Design and Manufacture. *Ellis Horwood*, Chichester, 1979.

[29] National Bureau of Standards. IGES,Initial Graphics Exchange Specification, version 5.2 *US PRO/IPO-100*,American National Standard, Gaithersburg, MD, 1986.

[30] W. Boehm, W. Farin and J. Kahmann. A survey of curve and surface methods in CAGD. *Computer Aided Geometric Design*, Vol. 1, 1-60, 1984.

[31] Les Piegl. On NURBS:A Survey. *IEEE Computer Graphics & Applications*, Vol. 1, 55-71,1991.

[32] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery. Numerical Recipes in C, Second Edition. *Cambridge University Press*.