Scan and Solve: Acquiring the Physics of Artifacts

Michael Freytag[*] Vadim Shapiro[†] Igor Tsukanov[‡]

Spatial Automation Laboratory
University of Wisconsin-Madison
1513 University Avenue
Madison, WI, 53706-1572, USA

June 27, 2007

Abstract

Existing physical artifacts including sculpture, mechanical parts, and anatomical structures are commonly acquired by modern surface and volumetric scanning technologies for archival, visualization, and diagnostic purposes. While the native representations for such data are largely sufficient for visualization purposes, more advanced field simulation currently requires extensive manual conversions into simplified surface and volume meshes compatible with the traditional finite element analysis pipeline. These conversions are tedious, error-prone, and require expertise in the mesh construction process. We demonstrate automated field simulation on acquired artifacts, bypassing the difficult geometric and topological meshing problems through a meshfree paradigm based on approximate distance fields computed from the native acquired data through sampling.

INTRODUCTION

Physical artifacts are frequently captured by acquisition technology for representation in the computer for visualization and measurement applications. For example, magnetic resonance imaging (MRI) and computed tomography (CT) scanning technologies enable physicians to "see" inside the body to perform diagnoses or plan surgeries [1]. Sculptural objects are acquired by laser and photo scanning technologies for archival documentation, and long-distance study [2]. In industry, 3D scanners are used to reverse engineer shapes for which the manual generation of computer models would otherwise be both time-consuming and inaccurate. For inspection, diagnosis, and measurement, the resulting volumetric and surface point data sets are largely adequate.

Frequently, physical analysis must be performed on the structures represented by this acquired geometry. In medicine, we may bring analysis closer to clinical application for the patient-centric design of orthopedic implants with variability to account for aging and anatomical difficulties, to investigate the risk of femoral neck fracture, to account for variability of bio-mechanical factors in animal studies, and to support interpretation of clinical results in follow-up studies [3]. Other applications include patient-specific identification of appropriate load regimes after skeletal surgery or in support of pre-operative planning. Past efforts have have converted the voxels of medical CT data directly to finite elements [4]. Similarly, engineering components have been captured by high energy CT and converted to hexahedral meshes for rapid stress analysis, reducing the total analysis time from 3 months to 3 days [5]. With automated analysis, non-destructive testing can be ex-

[*]E-mail: freytag@sal-cnc.me.wisc.edu
[†]E-mail: vshapiro@engr.wisc.edu
[‡]E-mail: tsukanov@engr.wisc.edu

tended beyond inspection, measurement and examination applications to computer simulation of the performance of components, essentially acquiring the physics of the artifacts themselves.

Computational stress analysis is largely based on the finite element method (FEM). To perform finite element stress analysis in acquired artifacts such as bones, a sequence of conversions is required. First, medical images must be converted into Computer Aided Design (CAD) geometry, and then, to finite element meshes that are suitable for input into analysis software. This process is shown in Figure 1. Often, geometric models obtained from medical images such as CT scans are very complex. In order to produce a domain mesh that is suitable for finite element analysis (FEA), the models require simplification and consequent manifold verification. In addition, meshing of the simplified CAD models is further complicated by the need to take into account spatial variation of the bones' physical properties. The regions where material properties change rapidly require increasing the mesh density.

Usually several separate CAD systems are required to prepare a bone model for analysis: segmentation of CT scans and reconstruction of the bones surface are performed in Mimics [6]; Solidworks is used for combining the geometric models of the bone and chosen implant; meshing and subsequent finite element analysis is performed in ABAQUS [7]. Often, the data conversions among these computer systems are difficult, non-robust, and time consuming. Furthermore, computer simulations of bone growth/resorption processes and shape optimization of an orthopedic implant are time consuming since they are iterative and require re-meshing of the model when the shape or mechanical properties of the bone or implant change. For these reasons, commonly used stress analysis tools like FEA are too cumbersome to support automation for case-by-case assessment.

Related Work

To do analysis using traditional techniques, the raw geometric data must be converted into data types compatible with the finite element method, usually a boundary representation. Typically, the resulting

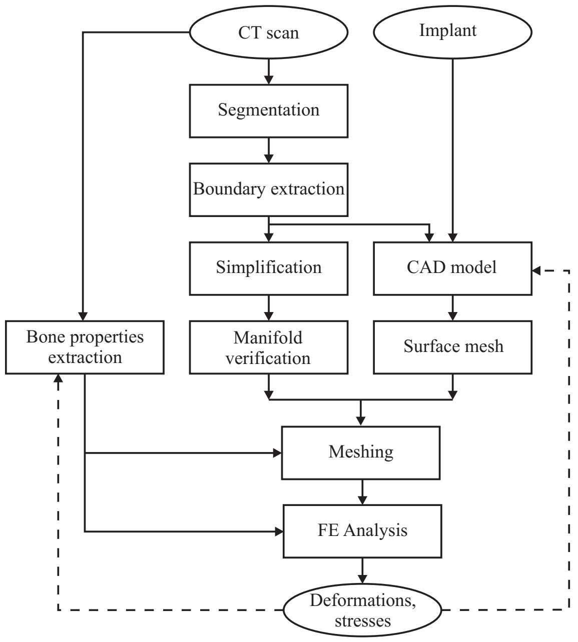


Figure 1: TYPICAL PROCESS FOR FINITE ELEMENT STRUCTURAL ANALYSIS OF A SCANNED BONE.

boundary representation is a dense surface mesh computed through application of an isosurface extraction algorithm to voxel data [8], reconstruction from contours extracted from slices of voxel data [9, 10], or reconstruction from surface points with an ordering induced to form a closed manifold [11, 12, 13]. Surface points may also be used to construct an implicit function whose zero set is extracted by some version of the marching cubes algorithm [14, 15]. Since the resulting surface meshes are quite dense, simplification steps must be performed to reduce the surface mesh density and ensure that a manageable number of well-shaped volumetric mesh elements can be generated [16, 17, 18, 19, 20].

The analysis pipeline is governed by the ability to manually generate compatible file formats as the interchange medium among the disparate tools. Unfortunately, limited compatibility of interchange formats can introduce increasing levels of geometric approximation that make validation of analysis results problematic. Also, when modifications must be made to the geometry, for example to simulate surgical operations, an additional side-path must be followed to completely reverse-engineer a shape and produce a CAD solid model. Finally, analysis environments may limit the types of analysis that can be performed depending on the format of the geometric input [10].

Analysis methods that convert CT voxel data to directly to hexahedral finite elements do not faithfully represent the smoothness of the outer surface, which is considered most significant in terms of stress and failure [21]. As a result the solution may suffer from singularities. Attempts have been made to smooth the analysis results and reduce the magnitude of the singularities [22]. However, the analysis results after such *ad hoc* massaging must be treated with skepticism. Smoothing the geometric representation, on the other hand, yields accurate analyses through better approximations of the geometry [23, 24, 25]. We too base our analyses on good approximations of the geometry, but use implicit representations computed from the segmented CT data using samples of signed Euclidean distance.

Geometric and topological problems of meshing and reconstruction notwithstanding, throughout the length of the current analysis pipeline different tools must be coupled by generating compatible geometric representations. Additional conversions are necessary to combine different geometric data sources through CAD editing or other surgical operations on the geometry. Specific choices must be made in these conversions that may limit the types of analysis that can be performed. Further, expertise in the analysis process itself is required of the user to ensure accurate analysis results. To advance automated analysis, the chain of unreliable geometric data conversions must be eliminated.

ANALYSIS ON ACQUIRED ARTIFACTS USING DISTANCES

The desired level of automation is possible through application of the distance sampling approach proposed by Freytag et al. in [26]. In that work, the authors extended the meshfree method of Rvachev et al. [27, 28] to use samples of distance to construct implicit representations for the geometry. These implicit representations form the backbone of the meshfree method, providing the means to satisfy essential boundary conditions and serving as a point membership classification function for domain integration within the solution procedure. Requiring only differentiable approximations of Euclidean distance, the meshfree method completely bypasses the difficult geometric and topological problems of meshing. The troublesome chain of geometric data conversions and topological constructions is thus avoided, instead abstracting the geometry to samples of distance – a property intrinsic to the geometry.

Meshfree Analysis

The original credit for the theory underlying our meshfree method goes to Kantorovich who proposed to represent a field that satisfies Dirichlet boundary conditions as the product of two functions: $u = \omega\Phi$, where ω is an implicit representation of geometry with zero set corresponding to the geometry and non-vanishing gradient at all points of the zero set [29]. Φ is a function with degrees of freedom to satisfy a governing differential equation, typically represented by a linear combination of basis functions: $\Phi = \Sigma C_i \chi_i$. A key feature of this formulation is the ability to satisfy the boundary conditions on the zero set of ω. Rvachev generalized Kantorovich's approach to include virtually all types of boundary value problems [27, 30]. Using distance-like functions, ω, Shapiro and Tsukanov showed that the method may be completely automated for a wide class of geometric and physical problems in a common meshfree environment[31, 32, 33]. More recently, Höllig improved the numerical stability of this ap-

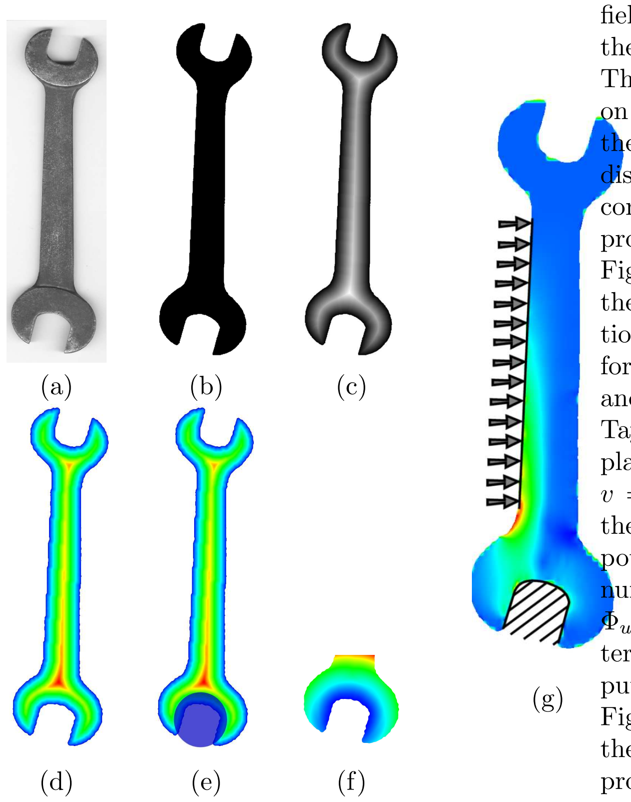


Figure 2: THE SEQUENCE OF STEPS (a-g) FOR ANALYSIS FROM ACQUIRED GEOMETRY.

proach by approximating the function Φ by linear combinations of weighted extended B-splines (WEB-splines) [34, 35].

The steps in the "Scan and Solve" approach are depicted graphically in Figure 2. The geometry is scanned to produce a 2 or 3 dimensional image, Figure 2(a). The image is segmented using image processing techniques to produce a binary image with foreground and background only, Figure 2(b). To facilitate distance sampling, a Euclidean distance transformation [36] is applied to compute a distance value for each image element. The resulting distance map, (c), is sampled at randomly distributed points and a set of basis functions are fit to these samples to produce a smooth approximation of the distance field, Figure 2(d). This approximate distance field is then used to support meshfree structural simulation. The displacement and loading boundary conditions on the wrench are shown in Figure 2(g). To apply the kinematic boundary condition; the approximate distance field, ω_f, to the fixed portion of boundary is constructed by normalized trimming [32] of the approximate distance field of the wrench in Figure 2(d). Figure 2(f) shows a portion of the distance field to the fixed boundary. The loading boundary condition is applied by integration along the line where the force is applied. The solution procedure combines ω_f and another system of B-splines into a generalized Taylor series that represents components of the displacement vector field, $\{u, v\}$, where $u = \omega_f \Phi_u$ and $v = \omega_f \Phi_v$ (one scalar series for each component of the displacement vector field). By minimizing the potential energy of elastic deformation, we compute numerical values of the undetermined coefficients in Φ_u and Φ_v. Once the displacement field has been determined, stresses inside the wrench body are computed using standard automatic differentiation tools. Figure 2(g) presents a plot of principal stress σ_1 in the wrench. Applicability of the meshfree method to problems in elasticity has been demonstrated previously in [37]. Here we extend the idea to incorporate differentiable distance fields computed from image data. The solution procedure requires point membership classification and integration over a segment of the boundary, tasks that do not require full boundary representation, domain meshing, or other topological reconstructions that require user intervention or expertise.

Scan and Solve System

Figure 3 shows a block diagram of the meshfree analysis system for acquired geometry. In the center is the meshfree structural solver. The solver takes the acquired geometry and specified boundary conditions, builds a *solution structure*, and computes the solution. The solution is determined through automatic differentiation of the basis functions, adaptive integration over the interior and boundary of the domain, and solution of the resulting linear system. We will discuss the internal details of the solver in the next

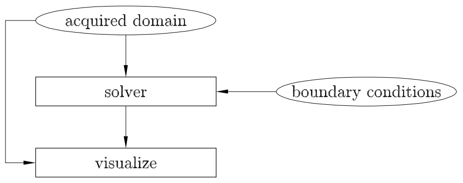


Figure 3: BLOCK DIAGRAM FOR THE SCAN AND SOLVE ANALYSIS SYSTEM FOR ACQUIRED GEOMETRY.

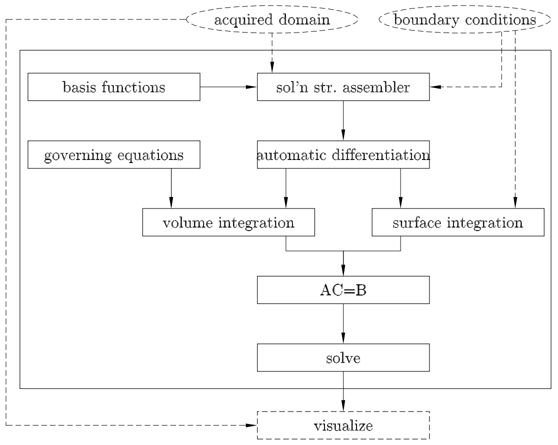


Figure 4: INTERNAL STRUCTURE OF THE MESHFREE SOLVER.

section. Kinematic boundary conditions are incorporated in the solution through the solution structure, while force boundary conditions are handled naturally through surface integration. The visualization component samples the solution field at points on the boundary and interior of the domain, maps the field values to hue, and provides a visual display of the solution for user interpretation.

Anatomy of the Meshfree Solver

Figure 4 outlines the components of the solver from Figure 3. The solution structure assembler takes all known information about the structural problem to form a single function with degrees of freedom to satisfy the governing differential equation. The problem geometry and boundary conditions are incorporated into the solution structure by approximate distances ω and ω_i, to the domain boundary and to regions of the boundary where kinematic boundary conditions are applied, respectively. For example, the solution structure for structural analysis problems represents the displacement vector field as three scalar components u, v, w:

$$\begin{aligned} u &= \omega_u \Phi_u \\ v &= \omega_v \Phi_v \\ w &= \omega_w \Phi_w, \end{aligned} \tag{1}$$

where $\omega_u, \omega_v, \omega_w$ are the approximate distances to the portions of boundary where the u, v, and w displacements are zero, respectively. The $\Phi_{u,v,w}$ in Expression (1) are linear combinations of basis functions:

$$\Phi_{u,v,w} = \sum C_i^{u,v,w} \chi_i^{u,v,w} \tag{2}$$

whose coefficients C_i are to be determined.

The concept of the solution structure is due to Rvachev who observed that the term $u = \omega\Phi$ can be considered a remainder term in a special case of a generalized Taylor series expansion of u in the neighborhood of the boundary by the powers of the distance ω to the boundary. A straightforward generalization of this observation allows systematic construction of solution structures for any and all boundary conditions. In each case, the solution structure will exactly satisfy all values and derivatives prescribed on the boundary and will contain necessary degrees of freedom (approximating the remainder term in the Taylor series expansion) to approximate the governing equations of the problem. It can be shown that such a solution structure forms a complete space of functions that satisfy the given boundary conditions exactly and approximate the governing equations of the problem [28].

A typical meshfree solution procedure requires computing partial derivatives of the solution structure, for example with respect to unknown coefficients and/or other parameters. Many differentiation methods have been developed, but all of them can be divided into three categories: numerical, symbolic, and automatic differentiation methods. Numerical differentiation techniques are not sufficiently accurate for most applications especially when high order derivatives have to be computed. Symbolic differentiation methods aim at derivation of symbolic expressions for the derivatives. Symbolic differentiation methods can compute the numerical values of the derivatives with high accuracy, but they face insurmountable computational difficulties when functions to be differentiated grow in size or are constructed by algorithms.

Automatic differentiation methods have been developed to address these computational difficulties. Automatic differentiation employs the same exact differentiation rules, but propagates the numerical values of the derivatives rather than their symbolic expressions. This allows computing derivatives of any order with high accuracy up to round-off error, as well as differentiation of functions constructed by algorithms or represented by a computer program [33, 38].

The degrees of freedom in the solution structure are typically represented by a linear combination of basis functions whose coefficients are determined by minimizing the potential energy of elastic deformation within the domain [39]. This minimization process requires numerical integration over the interior and portions of the boundary of the domain to form a linear system. Solution of this linear system then yields the numerical values of the basis function coefficients. We will see that integration over the interior and boundary of the domain requires only queries of ω.

Computational Components

In this section, we discuss critical computational components required to support physical simulation from acquired artifacts. From signed Euclidean distance we compute a smooth approximation, ω, that is used to satisfy boundary conditions and support both volumetric and surface integration. We also briefly describe visualization of the analysis results.

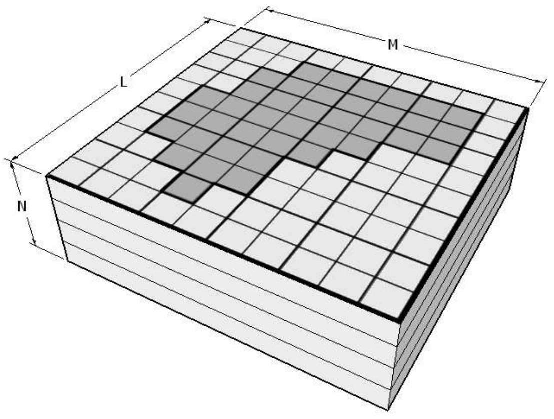


Figure 5: NOMENCLATURE FOR CT DATA SET SHOWING STACK OF N IMAGE SLICES, L PIXELS HIGH BY M PIXELS WIDE.

Distance Computation

Euclidean distance fields are spatially varying scalar fields whose evaluation at any point returns the exact distance to the geometry. Distance fields can be computed from most geometric representations, including clouds of points, meshes, triangulations, boundaries, volumetric data, and other unambiguous representations.

The preferred method for computing distance on voxel volumes is to use the Euclidean distance transform to propagate distance values among voxels via adjacency [40]. The Euclidean distance transformation takes advantage of the regular voxel structure to compute for each voxel of a binary image, the distance to the nearest boundary voxel. Saito's algorithm [41] is used, as it is relatively easy to implement, is accurate for the non-cubical voxels typically encountered with CT data, and is reasonably fast.

Our implementation of Saito's algorithm for computing the distance map from a binary voxel image consists of three one-dimensional stages operating on two 3D arrays of floating point values, A and B. The array dimensions correspond to height (L), width (M), and depth (N) of the CT data set, so we refer to the array entries as distance voxels, refer to Figure

5. Those voxels corresponding to the domain interior are referred to as foreground, while background refers to voxels exterior to the domain. A boundary voxel is a foreground voxel that is 6-connected to a background voxel. The individual steps in the algorithm are as follows:

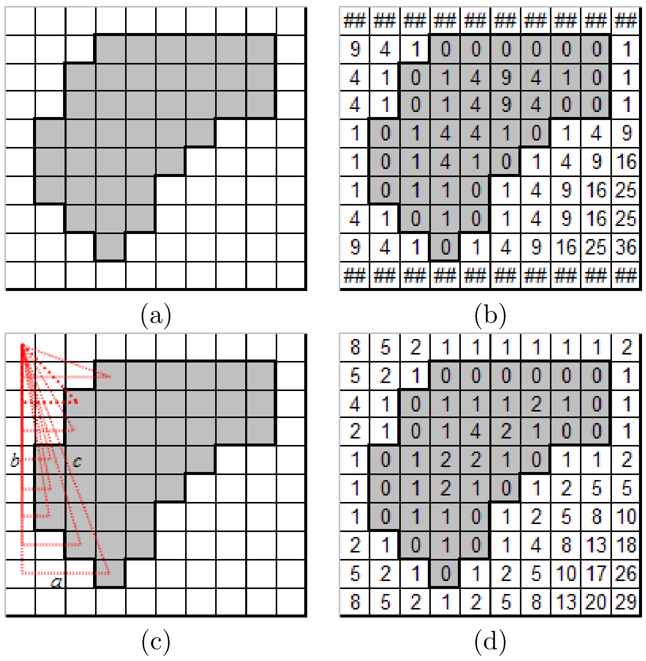
1. For each voxel, a_{ijk}, store the squared distance to the nearest boundary voxel in its row.

2. For each voxel, a_{ijk}, find a_{imk} that minimizes $c^2 = a_{imk}^2 + (m-i)^2$ for $m \in \{0, M\}$. Store c^2 at b_{ijk}.

3. For each voxel, b_{ijk}, find b_{ijn} that minimizes $d^2 = b_{ijn}^2 + (n-k)^2$ for $n \in \{0, N\}$. Store d at a_{ijk}.

Step 1 is usually carried out in two scans. One scan, left to right, propagates squared distance values after each boundary voxel is encountered in a row. A second scan right to left, stores the minimum of the squared distance value being propagated and the value stored in the previous scan. An example for a single slice is shown in Figure 6(b). Step 2, depicted in Figure 6(c), determines for each voxel, a_{ijk}, the stored value from its column in A that minimizes the length of the hypotenuse in 2D. The squared hypotenuse length is then stored at b_{ijk}. As depicted in Figure 7, in step 3, a search is conducted through the depth to find the shortest hypotenuse in 3D. The length of one leg of the triangle was previously stored during step 2. The length of the other leg is determined by the distance between the current plane and the other plane under consideration.

There are computational subtleties that improve the efficiency of the algorithm that, for clarity of presentation, are not described here. For details on efficiency improvements, matters of computational complexity, and issues with non-cubical voxels, the reader is referred to the original article, [41]. Additional discussion of exact and approximate distance transformation algorithms, as well as their applications can be found in [40].



(b)

##	##	##	##	##	##	##	##	##	##
9	4	1	0	0	0	0	0	0	1
4	1	0	1	4	9	4	1	0	1
4	1	0	1	4	9	4	0	0	1
1	0	1	4	4	1	0	1	4	9
1	0	1	4	1	0	1	4	9	16
1	0	1	1	0	1	4	9	16	25
4	1	0	1	0	1	4	9	16	25
9	4	1	0	1	4	9	16	25	36
##	##	##	##	##	##	##	##	##	##

(d)

8	5	2	1	1	1	1	1	1	2
5	2	1	0	0	0	0	0	0	1
4	1	0	1	1	1	2	1	0	1
2	1	0	1	4	2	1	0	0	1
1	0	1	2	2	1	0	1	1	2
1	0	1	2	1	0	1	2	5	5
1	0	1	1	0	1	2	5	8	10
2	1	0	1	0	1	4	8	13	18
5	2	1	0	1	2	5	10	17	26
8	5	2	1	2	5	8	13	20	29

Figure 6: FIRST TWO STAGES IN SAITO'S EUCLIDEAN DISTANCE TRANSFORMATION. (a) INPUT BINARY SLICE (b) DISTANCE MAP UPDATED WITH SQUARED HORIZONTAL DISTANCE TO NEAREST BOUNDARY VOXEL IN EACH ROW. (c) GRAPHICAL DEPICTION OF MINIMIZATION PROCESS WITHIN EACH COLUMN. (d) DISTANCE MAP UPDATED WITH SQUARED DISTANCE TO NEAREST BOUNDARY VOXEL.

Distance Smoothing

The distance map is piecewise-constant at the resolution of the voxel input and is therefore not differentiable. This makes it an unsuitable representation for a distance field, ω, in the meshfree method or other applications requiring differentiability. However, approximate distance fields can be constructed from a distance map by fitting. To do this, the approximate distance field is represented by a linear combination

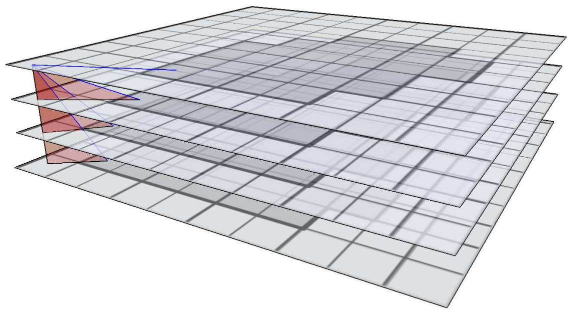


Figure 7: GRAPHICAL DEPICTION OF MINIMIZATION PROCESS THROUGH THE DEPTH FOR THE TOPMOST VOXEL ON THE LEFT.

of basis functions:

$$\omega(x,y,z) = \sum_{i=1}^{N} C_i^{\omega} \chi_i^{\omega}(x,y,z) \tag{3}$$

chosen from any sufficiently complete system of basis functions: polynomials, radial basis functions, B-splines, etc. Though we are free to choose any type of basis function, our choice determines the accuracy, convergence, differential, and computational properties of the resulting approximate distance field. Due to their good computational, approximation, and local properties [42] we use tensor product B-splines to approximate the distance maps. For example, tensor product B-splines of nth degree guarantee C^{n-1} continuity of the approximating function.

As we showed in [26] the numerical values of the coefficients $\{C_i^{\omega}\}_{i=1}^{N}$ can be computed by the least square method that minimizes the square of the difference between the signed Euclidean distance map and its approximation by a set of basis functions. The least square fit to the distance samples requires the minimization of the following functional:

$$F = \sum_{j=1}^{k} \left(\sum_{i=1}^{N} C_i^{\omega} \chi_i^{\omega} - d_j \right)^2, \tag{4}$$

where k is the number of distance samples, and d_j is the j^{th} signed distance sample taken at the point (x_j, y_j, z_j). This leads to the following linear system:

$$\mathbf{AC} = \mathbf{B}, \tag{5}$$

with matrix elements assigned as follows:

$$a_{i,j} = \sum_{l=1}^{k} \chi_i^{\omega}(x_l,y_l,z_l)\chi_j^{\omega}(x_l,y_l,z_l), \tag{6}$$

and

$$b_i = \sum_{l=1}^{k} d_l \chi_i^{\omega}(x_l,y_l,z_l). \tag{7}$$

For the tensor product B-splines, the matrix $\mathbf{A}$ has a block-diagonal structure that allows efficient storage using a row-indexed scheme [43]. Further, algebraic systems with such matrices can be efficiently solved by a bi-conjugate gradient method [43]. Once numerical values of the coefficients are determined, they are substituted into expression (3), establishing a signed approximate distance field to the geometric boundary. This approximation can then be evaluated and differentiated anywhere within the domain.

Representing Boundary Conditions

Individual kinematic and force boundary condition are represented using a pair consisting of a bounding volume and a vector field. The intersection of the set defined by $\omega = 0$ with the interior of the bounding volume defines the segment of the domain's boundary where the condition is applied, while the vector field defines the applied condition.

Associated with the bounding volume for kinematic boundary conditions is an approximate distance field, τ_i, that is positive inside the volume, zero at its boundary, and negative outside. The bounding volumes may be represented implicitly by τ_i alone, or τ_i may be derived from explicit geometric representations by sampling signed Euclidean distance [26]. R-trimming operations, see [32], are used to combine the τ_i with ω to construct the ω_u, ω_v, and ω_w in the solution structure (1). In the case of non-zero kinematic boundary conditions, a slight modification of the solution structure is made to include displacement vector fields φ_u, φ_v, and φ_w that satisfy the specified displacements:

$$\begin{aligned} u &= \omega_u \Phi_u + \varphi_u \\ v &= \omega_v \Phi_v + \varphi_v \\ w &= \omega_w \Phi_w + \varphi_w. \end{aligned} \tag{8}$$

These φ_* are constructed by transfinite interpolation using ω_i derived by R-trimming operations of ω with τ_i to smoothly interpolate the specified displacement conditions [44].

Volumetric Integration

Another key component in the meshfree solution of structural problems is computation of the numerical values of the undetermined coefficients in the basis functions by minimization of the potential energy of deformation within the domain. This requires differentiation of the solution structure (1) and volumetric and surface integration over the geometric domain. Differentiation of the solution structure with respect to the coefficients of the basis functions (and some algebra) yields an integral expression for the elements of the matrix in the linear system, A in Figure 4. Since the domain can be arbitrarily shaped, we must use numerical integration that is adaptive to the geometry at run-time.

Adaptive volumetric integration tends to be easier than finite element domain meshing as there is no need to consider element compatibility. Instead, the domain is subdivided into cells within which volumetric numerical quadrature can be applied. This is easier than dealing with the challenges of guaranteeing topological compatibility among adjacent elements. Octree subdivision of space ensures covering the domain with cells whose interiors are mutually disjoint. Further, the only query required to support volumetric integration is evaluation of ω for point membership classification against the domain.

Numerical integration boils down to summing weighted samples of the integrand, distributed spatially according to rules that guarantee a certain level of accuracy. For example, one dimensional Gaussian quadrature rules apply weights with a spatial distribution that will integrate exactly, polynomials up to the degree of the rule. Higher dimensional (rectangular) domains generally have tensor product Gaussian rules applied. Specialized rules have also been developed for triangular domains, so integrals over meshed surfaces are simply the sum of the integrals over the individual triangles.

Our implementation of volumetric integration extends the 2D integration algorithms described in [33]. We subdivide the domain into a uniform lattice of octrees with mutually-disjoint interiors and boundaries aligned with the basis functions used to represent the solution. The octree leaves are classified using the sign of ω to determine the membership of the leaf vertices. Within the leaves, lattice rules of required accuracy are allocated along rays radiating from faces, edges, or vertices depending on leaf classification. The "fully-exterior" status of leaf cells is also verified by queries of ω at points randomly distributed within leaves whose vertices classify as outside the domain.

Specific integration rules have been implemented for a subset of the possible leaf classifications. For implemented classifications, the integral is computed numerically. For example the simplest classification is fully-interior leaves, integrated using tensor product allocation of quadrature points in each dimension within the leaf cell. Cylindrical coordinates are used to allocate quadrature points in octree leaves having only two adjacent vertices interior to the domain. In cylindrical coordinates, quadrature points are allocated along rays radiating from a local z-axis joining the vertices classified as interior. The length of the quadrature intervals along the rays is determined by binary search for $\omega = 0$. Spherical coordinates are used to parameterize leaves having only one vertex in the domain. Quadrature points are allocated along rays radiating from the origin of the spherical system with the lengths determined by binary search against ω. For leaves with classifications that have not been implemented, recursive subdivision and integration is carried out. Recursion terminates when all leaves have been integrated, or a minimum leaf size is reached. Allocation schemes for more complex classifications have been implemented, but their description is beyond the scope of this article. However, all require only membership classification using the sign of ω.

Surface Integration

Numerical integration is required over those portions of the domain's boundary where force boundary conditions are applied. Surface integration yields the elements of the B vector in the linear system, see Figure 4. The easiest method for performing this integration is to form a polygonal mesh approximation of the domain's boundary and perform numerical integration using standard triangle quadrature rules. Fortunately, the constraints on surface meshes for surface integration are not particularly demanding. Therefore, marching cubes or another polygonization algorithm may be used to extract the zero set of ω. The shape of the triangles is not a concern so long as they form a good approximation of the boundary where the forces are applied. Further, with the meshfree approach, the computational effort required is determined by the background grid of the basis functions independently of the surface mesh density. Even the connectivity of the surface meshes is not a concern so long as the triangle interiors are disjoint and they form a covering of the domain boundary.

In contrast, the geometric constraints on surface meshes for finite element meshing are particularly stringent. Surface meshes to support finite element analysis are required to have triangles with favorable aspect ratios, a minimum size, and connectivity to interior elements. For best stress modeling accuracy, the surface triangles must be nearly equilateral. Their minimum size is bounded by the available computational resources since the domain mesh must be compatible with the surface discretization. Notwithstanding the geometric and topological problems of domain meshing from surface meshes, geometrically-accurate, dense surface meshes lead to dense domain meshes and the need for more computational resources.

Once the segments of our boundary have been discretized using triangles, there are a variety of rules for allocating quadrature points within these triangles. These rules can be grouped into two categories: *symmetric* and *transformed*. Symmetric rules assign points in a distribution that has no favored vertex. The simplest symmetric rule is the *midpoint rule* which samples the integrand at the centroid of each triangle. Next is the *trapezoid rule* which samples the integrand at the vertices of each triangle. Higher-order rules exist for triangles, but their derivation is difficult and they frequently require sampling outside the domain[45].

The relative scarcity of symmetric rules compared to the many choices available for integrating over rectangular regions leads us to rules that are constructed by transformation of coordinates. Transformed rules are constructed by repeated application of one-dimensional rules in a direction parallel to one side of the triangle. One technique, *Duffy transformation*, transforms the triangle into a unit square for which the repeated application of one-dimensional rules is standard practice[46]. Due to the wide variety of one-dimensional quadrature rules, and the accompanying wealth of development, there is virtually no limit to the rules we can derive for triangles. However, there are two drawbacks to rules generated in this way. First, we can construct three different rules for each triangle and we cannot know beforehand which will be the most accurate. Second, the rules require slightly more sample points for a given accuracy compared to symmetric rules. Despite these minor drawbacks, transformed rules are arguably the most valuable for numerical integration over triangles.

Numerical surface integration is performed using triangular quadrature rules applied over mesh approximations of the zero set of ω, localized to the regions where force boundary conditions are applied. The mesh approximations are computed using a modified version of Bloomenthal's marching cubes algorithm [47] that uses repeated evaluation of ω to isolate the intersection of its zero set with the edges of the polygonization lattice. Localization of the mesh is achieved in two steps. First, the initial bounding box for the marching cubes algorithm is determined from the bounding box for the geometric extent of each force boundary condition. Each extracted mesh thus approximates the zero set of ω where a force boundary condition is applied. Second, the extracted mesh is trimmed against the boundary condition volume by discarding all triangles outside the volume and trimming those that cross. We are left with triangular mesh approximations of the boundary of the domain localized to those boundary segments where force conditions are applied.

Finally, representing the solution using a set of basis functions having finite support requires that the surface integrals be computed over the intersection of the above localized mesh approximations with the supports of the basis functions. The meshes are thus subdivided along the planes defining the supports of the basis functions. Adding edges and faces to the mesh in this manner simplifies the allocation of quadrature points and weights to the appropriate basis functions. The surface integral is then computed over the resulting mesh. Membership of the quadrature points against the basis function supports determines their contribution to the linear system.

Field Visualization

Once a solution has been computed, the results must be presented in a human interpretable form. A natural method is to visualize the solution on the boundary and interior of the domain by mapping solution values to hue, and using these hue values to color a 3D rendering of the acquired geometry. Visualization thus has a distinct similarity to surface integration, but instead of sampling basis functions and differential equations we will be sampling the solution field. Obviously then, we can extract the zero set of ω as a set of triangles using a marching cubes algorithm, evaluate the solution field at the vertices of these triangles, map field values to hue, and feed the colored triangles to the graphics subsystem for display.

Alternately, point-based graphics methods may be brought to bear. Each boundary voxel is used to generate a surface element, a "surfel" whose hue is determined by sampling the solution field [48]. For greatest accuracy, the centroids of the surfels should lie on the zero set of ω; however, since the zero set of ω only approximately interpolates the centroids of the boundary voxels, we must shift the centroid, $\mathbf{x}$, of each surfel along the gradient of omega:

$$\mathbf{x}_{new} = \mathbf{x} + \omega(\mathbf{x})\nabla\omega(\mathbf{x}). \tag{9}$$

This moves the sample points for visualization to the boundary of the geometric domain. With this approach, the initial voxel resolution must provide sufficient density to support seamless visualization and accurate depiction of field values.

RESULTS

Geometric Accuracy

Geometric accuracy is central to the accurate modeling of the physical performance of acquired artifacts. We have seen that fitting a set of basis functions to samples of signed Euclidean distance is an effective means to produce geometrically accurate approximate distance fields from acquired data. We measure the convergence properties and accuracy of our constructed approximate distance fields by evaluating the deviation of the zero set of ω over a mesh representation of the Visible Human femur. This benchmark mesh representation is constructed using the proprietary surface extraction algorithms built into Mimics [6]— state of the art software in the medical research community. The approximate distance field is constructed from the binary voxel pre-cursor for the mesh representation. Simultaneous convergence of the mean and variance of ω, evaluated on the mesh representation, indicates convergence to the geometric boundary. Figure 10 shows the convergence of the mean and variance of the zero set of the approximate distance field from the benchmark geometric boundary with decreasing grid spacing.

We have observed through numerical experimentation that the mean deviation of the zero set of the approximate distance field approaches a non-zero value with increases in both sampling density and B-spline grid resolution. Since the distance map associates distance values with the centroids of the distance map cells, the zero set of the approximate distance field then approximates the centroids of the boundary cells (see Figure 9.) In actuality, the interior of all boundary cells is interior to the domain, and the zero set of the approximate distance field ends up offset toward the interior of the domain by, on average, half the voxel edge length. To compensate, we add half the average voxel edge length to all distance samples, cognizant that we are introducing some small error due to non-cubical voxels. Taking into account the offset required of the values from the distance map, we observe convergence of the results to a value closer to zero. The variance remains the same as the results without an offset, irrespective of sampling density or grid spacing.

To improve the computational performance of the fitting process, the values from the distance map are tri-linearly interpolated to ensure samples are taken from a continuous function. Interpolation of the values in the distance map during sampling and fitting greatly improves the conditioning of the matrix of the linear system, substantially reducing the number of iterations required to reach a solution.

Bone Stress Analysis

Coupling a geometrically accurate, approximate distance field constructed as above with volumetric and surface integration techniques, we are able to perform simple structural analysis on the Visible Human femur. Figure 11 shows the scaled deflection of the bone geometry with a uniform pressure of 1.11 MPa applied to the top of the femur ball in a direction parallel to the primary femur axis. This is equivalent to a load of 2540 N. The distal end of the femur is fixed. The bone geometry is represented by an approximate distance field consisting of a $40 \times 46 \times 217$ grid of tri-cubic B-splines fit to 20 random samples of the interpolated distance map. The deformation field is represented using linear B-splines on uniform $(13 \times 13 \times 51)$ cubical lattice with a spacing of 10 mm. The average of longitudinal and lateral bone modulus is used to assign an isotropic modulus of 17.2 GPa to the bone. A Poisson ratio of 0.3 is used, as is typical for bone. The model exhibits deflection in the expected direction, but the deflection magnitude is doubtlessly low since the true material distribution is not included in this simple model.

Conclusions

We have described a system and the fundamental computational components for performing analysis from acquired artifacts— essentially capturing their physics. The only query we require of the geometry is signed distance to its boundary. Distilling the solution process to queries of distance enables automated analysis of acquired artifacts without the burden of traditional reverse engineering processes. Being based on distance, the techniques can also be applied to analysis of engineering components irrespective of the existence of a traditional CAD model. While not stressed here, the presented meshfree approach also allows the resolutions for the geometry and the field to be independently controlled [26].

The ultimate goal is an automated process for performing rapid field analysis on physical artifacts *in situ*. We expect to extend the approach to include material properties for accurate modeling of bone physical performance, followed by validation of analysis results through physical experimentation and comparison with published reports. Other avenues of exploration include application to point sampled geometry from art and the complete reverse-engineering of physical artifacts acquired by industrial CT.

Acknowledgments

This research is supported in part by the National Science Foundation grants OCI-0636206, CMMI-0323514, CMMI-0621116, CMMI-0500380; and the Wisconsin I&EDR Program. The authors would also like to thank Jill Schmidt and Dr. Heidi Ploeg, for the segmented Visible Human Data. Thanks also to Brian Luft for his implementation of Saito's Euclidean distance transformation.

References

[1] Lattanzi, R., Viceconti, M., Petrone, M., Quadrani, P., and Zannoni, C., 2002. "Applications of 3d medical imaging in orthopaedic surgery:introducing the hip-op system". In Proceedings of the First International Symposium on 3D Data Processing Visualization and Transmission, pp. 808–811.

[2] Levoy, M., 1999. "The digital michelangelo project". In Proceedings of Second International

Conference on 3-D Digital Imaging and Modeling.

[3] Viceconti, M., Davinelli, M., Taddei, F., and Cappello, A., 2004. "Automatic generation of accurate subject-specific bone finite element models to be used in clinical studies". *Journal of Biomechanics*, pp. 1597–1605.

[4] Keyak, J. H., Meagher, J. M., Skinner, H. B., and Mote, C. D., 1990. "Automated three-dimensional finite element modelling of bone: a new method". *Journal of Biomedical Engineering*, pp. 389–397.

[5] Hollister, S. J., Guldberg, R. E., Kikuchi, N., and McGee, R. M., 1998. Reverse engineering of geometrically complex automotive structures using x-ray computed tomography and digital image based finite element methods. Tech. rep., SAE Technical Paper 981193.

[6] Materialise: Mimics. Software.

[7] Dassault systmes: Abaqus. Software.

[8] Lorensen, W. E., and Cline, H. E., 1987. "Marching cubes: A high resolution 3d surface construction algorithm". *Computer Graphics,* **21**(3), pp. 163–169.

[9] Armand, M., Beck, T. J., Boyle, M., Oden, M. Z., Voo, L., and Shapiro, J. R., 2003. "A semi-automatic technique for generating parametric finite element model of femur from imaging modalities". In Proceedings of 2003 Summer Bioengineering Conference.

[10] Coleman, J. C., Roberts, M. D., Maender, C. M., and Hart, R. T., 2000. Solid model creation from ct image data. Tech. rep.

[11] Bajaj, C. L., Bernardini, F., and Xu, G., 1995. "Automatic reconstruction of surfaces and scalar fields from 3d scans". In Proceedings of SIGGRAPH, pp. 109–118.

[12] Amenta, N., Bern, M., and Kamvysselis, M., 1998. "A new voronoi-based surface reconstruction algorithm". In Proceedings of SIGGRAPH, pp. 415–422.

[13] Dey, T. K., and Giesen, J., 2001. "Detecting undersampling in surface reconstruction". In Proceedings of 17th ACM Symposium on Computational Geometry, pp. 257–263.

[14] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W., 1992. "Surface reconstruction from unorganized points". In Proceedings of SIGGRAPH, pp. 71–78.

[15] Ohtake, Y., Belyaev, A., Alexa, M., Turk, G., and Seidel, H.-P., 2003. "Multi-level partition of unity implicits". In Proceedings of SIGGRAPH, pp. 463–470.

[16] Schroeder, W. J., Zarge, J. A., and Lorensen, W. E., 1992. "Decimation of triangle meshes". In Proceedings of SIGGRAPH.

[17] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W., 1993. "Mesh optimization". In Proceedings of SIGGRAPH.

[18] Hoppe, H., 1996. "Progressive meshes". In Proceedings of SIGGRAPH, pp. 99–108.

[19] Garland, M., and Heckbert, P. S., 1997. "Surface simplification using quadric error metrics". In Proceedings of SIGGRAPH.

[20] Frey, P. J., 2004. "Generation and adaptation of computational surface meshes from discrete anatomical data". *International Journal for Numerical Methods in Engineering*, pp. 1049–1074.

[21] Boyle, M. P., Magee, T. C., Baily, L. E., and Beck, T. J., 1999. "Automated three-dimensional finite element modelling of the proximal femur using dual-energy x-ray absorptiometry derived structural geometry". In Proceedings of the Seventh International Symposium on Computer Simulation in Biomechanics.

[22] Guillaume, T., and Guldberg, R. E., 2000. "Improving the local solution accuracy of large-scale digital image-based finite element analyses". *Journal of Biomechanics*, pp. 255–259.

[23] Grosland, N. M., and Brown, T. D., 2002. "A voxel-based formulation for contact finite element analysis". *Computer Methods in Biomechanics and Biomedical Engineering,* **5**(1), January, pp. 21–32.

[24] Wang, Z. L., Teo, J. C. M., Chui, C. K., Ong, S. H., Yan, C. H., Wang, S. C., Wong, H. K., and Teoh, S. H., 2005. "Computational biomechanical modeling of the lumbar spine using marching-cubes surface smoothened finite element voxel meshing". *Computer Methods and Programs in Biomedicine,* **80**(1), October, pp. 25–35.

[25] Boyd, S. K., and Müller, R., 2006. "Smooth surface meshing for automated finite element model generation from 3D image data". *Journal of Biomechanics,* **39**(7), pp. 1287–1295.

[26] Freytag, M., Tsukanov, I., and Shapiro, V., 2006. "Field modeling with sampled distances". *Computer Aided Design,* **38**(2), February, pp. 87–100.

[27] Rvachev, V. L., 1982. *Theory of R-functions and Some Applications.* Naukova Dumka. In Russian.

[28] Rvachev, V. L., Sheiko, T. I., Shapiro, V., and Tsukanov, I., 2000. "On completeness of RFM solution structures". *Computational Mechanics*, pp. 305–316.

[29] Kantorovich, L. V., and Krylov, V. I., 1958. *Approximate Methods of Higher Analysis.* Interscience Publishers.

[30] Rvachev, V. L., and Sheiko, T. I., 1996. "*R*-functions in boundary value problems in mechanics". *Applied Mechanics Reviews,* **48**(4), pp. 151–188.

[31] Shapiro, V., and Tsukanov, I., 1999. "Meshfree simulation of deforming domains". *Computer-Aided Design,* **31**(7), pp. 459–471.

[32] Shapiro, V., and Tsukanov, I., 1999. "Implicit functions with guaranteed differential properties". In Proceedings of Fifth ACM Symposium on Solid Modeling and Applications, pp. 258–269.

[33] Tsukanov, I., and Shapiro, V., 2002. "The architecture of sage a meshfree system based on RFM". *Engineering with Computers*, pp. 295–311.

[34] K., H., U., R., and J., W., 2001. "Weighted extended b-spline approximation of dirichlet problems". *SIAM Journal on Numerical Analysis,* **39**(2), pp. 442–462.

[35] Höllig, K., 2003. *Finite Element Methods with B-Splines.* No. 26 in Frontiers in Applied Mathematics. SIAM.

[36] Leymarie, F., and Levine, M. D. "Fast raster scan distance propagation on the discrete rectangular lattice". *Computer Vision, Graphics, and Image Processing,* **55**(1), January.

[37] Rvachev, V. L., and Sinekop, N. S., 1990. *R-functions method in problems of the elasticity and plasticity theory.* Nauk. dumka, Kiev. In Russian.

[38] Tsukanov, I., and Hall, M., 2003. "Data structure and algorithms for fast automatic differentiation". *International Journal for Numerical Methods in Engineering,* **56**(13), April, pp. 1949–1972.

[39] Washizu, K., 1982. *Variational methods in elasticity and plasticity*, 3rd ed. Oxford, Eng. ; New York : Pergamon Press.

[40] Cuisenaire, O., 1999. "Distance transformations: Fast algorithms and applications to image processing". *PhD Thesis.*

[41] Saito, T., and Toriwaki, J.-I., 1994. "New algorithms for euclidean distance transformation of an n-dimensional digitized picture with applications". *Pattern Recognition*, pp. 1551–1565.

[42] de Boor, C., 2001. *A Practical Guide to Splines.* Springer-Verlag.

[43] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., 1992. *Numerical Recipes in C*, second ed. Cambridge University Press.

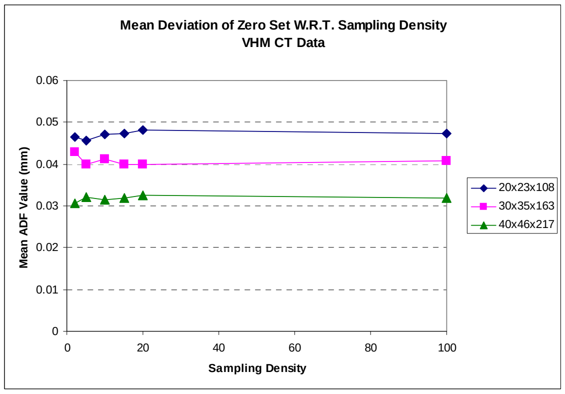
[44] Rvachev, V. L., Sheiko, T. I., Shapiro, V., and Tsukanov, I., 2001. "Transfinite interpolation over implicitly defined sets". *Computer-Aided Geometric Design,* **18**(4), pp. 195–220.

[45] On, R. T. Numerical integration.

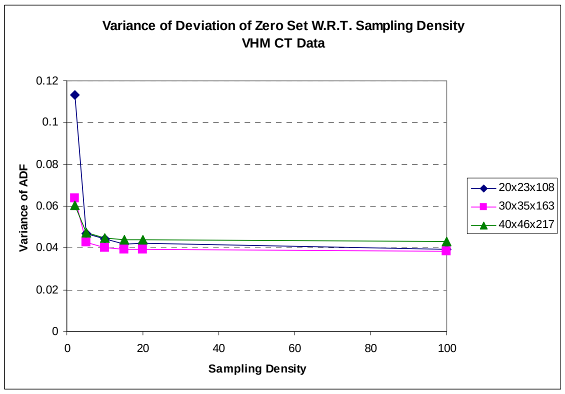
[46] Lyness, J. N., and Cools, R., 1994. "A survey of numerical cubature over triangles". *Technical Report: Mathematics and Computer Science Division, Argonne National Laboratory*, November.

[47] Bloomenthal, J., 1988. "Polygonization of implicit surfaces". *Computer Aided Geometric Design,* **5**(4), November, pp. 341–355.

[48] Pfister, H., Zwicker, M., van Baar, J., and Gross, M., 2000. "Surfels: Surface elements as rendering primitives". In Proceedings of SIGGRAPH, pp. 335–342.



(a) MEAN DEVIATION



(b) VARIANCE OF DEVIATION

Figure 8: FIGURES ILLUSTRATE THAT WITH INCREASING SAMPLING DENSITY THE ZERO SET OF THE APPROXIMATE DISTANCE FIELD CONVERGES TO THE GEOMETRIC BOUNDARY. DIFFERENT CURVES CORRESPOND TO THE NOTED GRID SIZES.

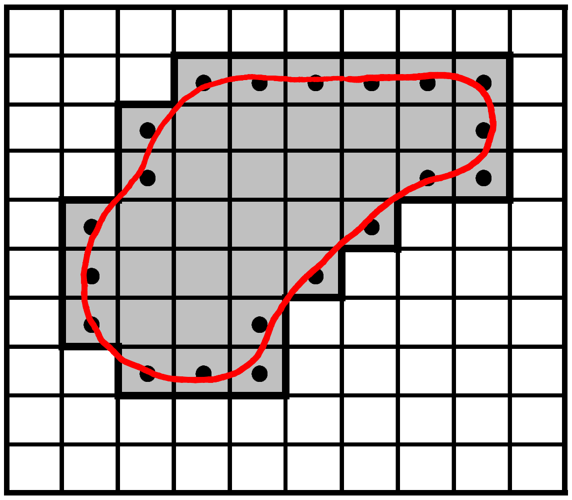
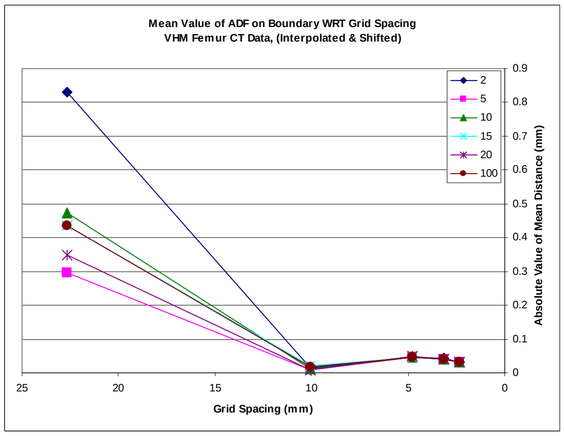
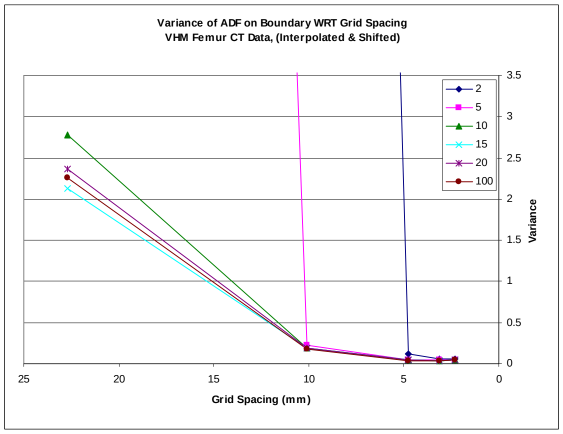


Figure 9: FIGURE ILLUSTRATES HOW ZERO SET OF DISTANCE FIELD IS INTERIOR TO BOUNDARY.



(a) MEAN DEVIATION



(b) VARIANCE OF DEVIATION

Figure 10: FIGURES DEMONSTRATE THAT WITH DECREASING GRID SPACING THE ZERO SET OF THE DIFFERENTIABLE DISTANCE FIELD CONVERGES TO THE GEOMETRIC BOUNDARY. DIFFERENT CURVES CORRESPOND TO THE NOTED SAMPLING DENSITIES.

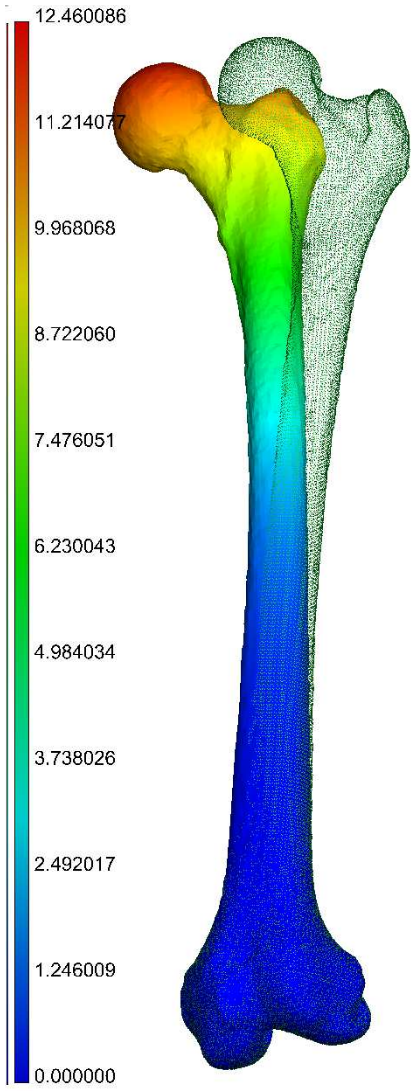


Figure 11: SCALED DEFLECTION OF SOLID BONE COMPUTED WITH THE SCAN AND SOLVE APPROACH.