B-Mode Ultrasound Image Simulation in Deformable 3-D Medium

Orcun Goksel*, *Student Member, IEEE*, and Septimiu E. Salcudean, *Fellow, IEEE*

***Abstract*—This paper presents an algorithm for fast image synthesis inside deformed volumes. Given the node displacements of a mesh and a reference 3-D image dataset of a predeformed volume, the method first maps the image pixels that need to be synthesized from the deformed configuration to the nominal predeformed configuration, where the pixel intensities are obtained easily through interpolation in the regular-grid structure of the reference voxel volume. This mapping requires the identification of the mesh element enclosing each pixel for every image frame. To accelerate this *point location* operation, a fast method of projecting the deformed mesh on image pixels is introduced in this paper. The method presented was implemented for ultrasound *B*-mode image simulation of a synthetic tissue phantom. The phantom deformation as a result of ultrasound probe motion was modeled using the finite element method. Experimental images of the phantom under deformation were then compared with the corresponding synthesized images using sum of squared differences and mutual information metrics. Both this quantitative comparison and a qualitative assessment show that realistic images can be synthesized using the proposed technique. An ultrasound examination system was also implemented to demonstrate that real-time image synthesis with the proposed technique can be successfully integrated into a haptic simulation.**

***Index Terms*—*B*-mode image synthesis, deformation slice rendering, medical image simulation, sonography training, ultrasound image simulation.**

I. Introduction

ULTRASOUND is a noninvasive and safe medical imaging modality and hence one of the most commonly used examination tools. However, image anisotropy and the existence of various significant artifacts cause the need for extensive echographer training. Current standard education is in the form of supervised examination of real pathologies during clinical practice. Despite its many advantages, this approach involves significant time expenditure of qualified personnel and can only be performed when a supervisor and a patient are available. Furthermore, training on rare pathologies poses a problem. Indeed, students have the chance to learn only 80% of the important pathologies during one year of standard education [1]. This need for ultrasound examination training has motivated several computer-based simulation environments [2]. In addition to examination, training of medical procedures that utilize ultrasound imaging, e.g., prostate brachytherapy and breast biopsy, can also significantly benefit from such simulation techniques. The ability to mentally register 2-D image slices within the 3-D anatomy is a nontrivial skill required by any sonographer. Real-time ultrasound simulators have the potential to accelerate and improve such training.

Manuscript received February 10, 2009; revised February 16, 2009. First published March 10, 2009; current version published October 28, 2009. *Asterisk indicates corresponding author.*

*O. Goksel is with the Department of Electrical and Computer Engineering, University of British Columbia, V6T 1Z4 Vancouver, BC, Canada (e-mail: orcung@ece.ubc.ca).

S. E. Salcudean is with the Department of Electrical and Computer Engineering, University of British Columbia, V6T 1Z4 Vancouver, BC, Canada (e-mail: tims@ece.ubc.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMI.2009.2016561

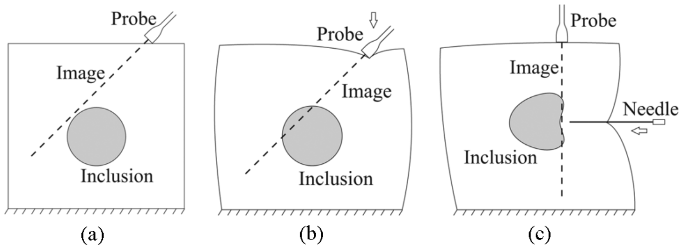


Fig. 1. An image slice (a) before and (b) after a deformation caused by probe pressure and (c) illustration of deformation during needle insertion.

In a typical ultrasound simulation scenario, the user must be presented with an image slicing the target anatomy. In an actual diagnostic or operative procedure, such target anatomy is often deformed under various forces, such as ultrasound probe contact, as illustrated in Fig. 1(a) and (b). Note that an ultrasound probe only compresses the surface of the tissue, whereas there exist other medical tools that further manipulate the tissue internally, e.g., percutaneous needles as in Fig. 1(c). A realistic image simulation should take such tissue deformations into account in order to deliver an immediate representation of the anatomy in its current deformed configuration. Modeling of tissue deformation has been studied extensively in the literature [3]–[6]. Common techniques such as mass-spring models and the finite element method (FEM) use a discretization (mesh) of the tissue volume and corresponding elasticity parameters to approximate its behaviour under load. In general, these model parameters are abstracted *a priori* and used with given forces in real-time to compute deformation, which is commonly expressed as a set of displacements of the given mesh nodes. To enable a real-time computation of deformation, this discretization often has a significantly coarser structure than the typical resolution of medical imaging modalities. This paper presents an image generation technique in deformed 3-D meshes and addresses the computational challenges for real-time performance. Realistic simulation of ultrasound, which is a real-time imaging modality, is the primary target application of our image generation technique. While the image slicing methodology we propose is described for *B*-mode ultrasound, it also applies to other modalities such as magnetic resonance (MR) and computed tomography (CT).

The paper is organized as follows. First, our choice of interpolation procedure, which consists of finding the image pixel intensities by referring their positions back to the nominal predeformed configuration, is introduced. Then, its application within meshes that are deformed based on the finite element method (FEM) is outlined in 2-D. Next, the pixel location problem that arises in such a scheme and our proposed numerical treatment for this are presented in 3-D. In the results section, the proposed technique is demonstrated for real-time ultrasound synthesis in a tissue-mimicking ultrasound phantom and *in vivo* thigh data deformed by the ultrasound probe itself. A discussion of the limitations and possible future extensions conclude this paper.

II. Previous Work

An ultrasound simulator necessitates rapid and realistic image rendering of deformed tissue in response to probe or tool manipulation by a trainee. There exist two major approaches for simulating *B*-mode ultrasound images, *the generative approach* and *the interpolative approach*. The former simulates the ultrasonic wave propagation by using accurate models of the probe, the tissue scatterers, and the wave interaction [7], [8]. Generating a single *B*-mode frame using this technique takes hours. Thus, this approach is not suitable for real-time applications. Furthermore, in practice it is not possible to extract an exact scatterer model of a complex medium such as the tissue and hence the images generated with this technique typically look artificial. The latter approach generates images by interpolating from preacquired images of the volume. While interpolation directly from arbitrarily-oriented B-scans was demonstrated in [9], the construction of a regular-grid reference volume, called *3-D ultrasound reconstruction* [10], [11], is commonly the preferred method because it enables data processing with off-the-shelf algorithms. UltraSim [12], which is one of the first commercial ultrasound image simulators, and several others [13]–[18] follow this latter approach. Refer to [2] for a review on ultrasound training simulators.

Note that anisotropic image artifacts, such as shadowing and reverberation in ultrasound, may not be reproduced correctly by an interpolation scheme due to their direction-dependent characteristics. One attempt to remedy this shortcoming is to acquire real ultrasound images of the volume at several positions/orientations. Subsequently, for a given probe location during simulation, the image that corresponds best with that orientation can be selected from that database and shown to the user. This is not feasible in practice due to the unlimited number of possible probe and/or medical tool configurations during a simulation [1]. Since a generative simulation approach with a full-blown wave interaction model is not feasible for real-time applications, some recent work has focused on developing heuristic models that can be computed in real-time. Some researchers looked at the problem in the context of computer graphics, such as first texture mapping different tissue regions by precomputed backgrounds, then imposing a Gaussian noise to generate an artificial speckle pattern, and finally applying a depth-dependent radial blurring to simulate a convex probe [19], [20]. Others proposed processing imaginary rays mimicking ultrasound using heuristic interaction functions defined for coarse (pixel level) tissue representations with abstracted parameters, namely attenuation, reflection, and scatterer power [1]. Unfortunately, the adjustment of such parameters was not addressed in this work. Deriving parameters from CT data was also proposed in [21] and in [22], separately, in order to generate ultrasound images by processing CT images. Although such pseudo-generative methods for echography simulation are appealing, due to the substantially complex nature of actual wave interactions, it is extremely difficult to generate even common ultrasound phenomena, such as speckle formation, using these methods, let alone realistic images. As a result, existing simulators that are studied for clinical training scenarios [2], [12], [13], [15]–[18] are interpolation-based.

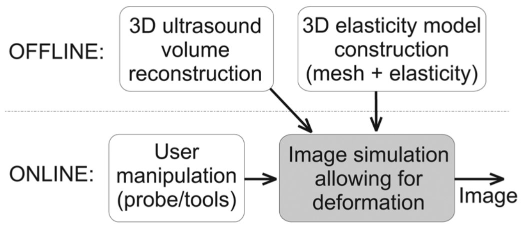

OFFLINE:
3D ultrasound volume reconstruction
3D elasticity model construction (mesh + elasticity)
ONLINE:
User manipulation (probe/tools)
Image simulation allowing for deformation
Image

Fig. 2. Online and offline steps of the proposed interpolation-based simulation.

In many medical procedures, such as prostate brachytherapy [5], brain surgery, or breast biopsy, significant deformation is caused by medical tools or by the ultrasound probe. In certain applications, such as in the diagnosis of deep-vein thrombosis (DVT), deformation observed in ultrasound images during deliberate probe indentation contains essential diagnosis information. Fast synthesis of ultrasound images in soft tissues under deformation will facilitate the development of training simulators. With this goal, a DVT diagnosis simulator was proposed in [18]. It simulates the probe pressure by first slicing an image from the 3-D ultrasound data set and then applying a 2-D elastic deformation to this image using quadtree-splines. This 2-D in-plane deformation is precomputed offline by registering the segmentations of predeformed and postdeformed anatomy of a test case [23].

Real-time ultrasound image slicing using physically-valid 3-D deformation models has not been addressed in the literature. Our work is motivated by this need. A recent work applies similar techniques to generate ray-traced volume rendering of a deformable liver model [24].

For a deformed-volume image slicing strategy, as illustrated in Fig. 2, a reference volume dataset is required. The reference image volume can either be obtained using a 3-D ultrasound probe or, alternatively, it can be constructed from individual 2-D *B*-mode slices. This 3-D ultrasound reconstruction has been studied extensively in the literature [10], [11], [25], [26]. Given this reference volume and a mesh-based deformation model, the image synthesis component (Fig. 2) of an ultrasound simulator is the subject of this paper, preliminary results of which were presented earlier in [27] and [28].

III. Methods

Let the spatial voxel locations of an $i \times j \times k$ 3-D regular grid be V^0, where the superscript zero refers to this being the initial (time-zero) configuration of these voxels. Assume that V^0 are the locations of a given reconstructed volume, in other words, the locations at which the intensities (also known as the *gray-values*) $I^0(V^0)$ are known *a priori* [see Fig. 3(a)]. Note that the illustrations in Fig. 3 are given in 2-D for the ease of presentation, although they represent 3-D concepts.

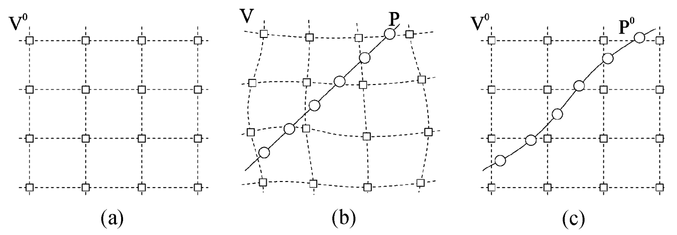


Fig. 3. Voxel data and image plane in (a) nominal, (b) postdeformation, and (c) undeformed configurations in 2-D (circles denote the image pixels and squares denote the volume data voxels).

Let V^0 be transformed to V by the deformation $f(\cdot)$ at a given simulation instance as follows:

$$V = f\left(V^0\right) \tag{1}$$

as shown in Fig. 3(b). Throughout this paper, these two states V^0 and V are referred as the predeformed and the deformed tissue configurations, respectively.

Consider an image, formed by a set of n planar equidistant pixels P, cutting this deformed volume V. Such an image is shown with circles in Fig. 3(b). Synthesizing this image involves finding the immediate intensity values $I(P)$ at these n pixel locations for every image frame to be displayed on the screen. Note that this operation has a lower bound of $\Omega(n)$. Indeed, any algorithm processing an entire image (even just simply displaying it on screen) needs to access all n pixels proving this lower bound.

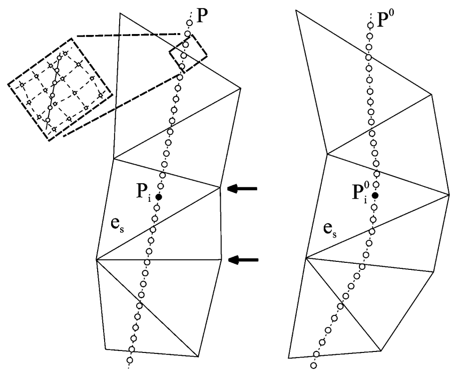
A. Accounting for the Deformation

One approach to the image synthesis above is to first compute the deformed voxel locations V and then to find (interpolate) the pixel intensities $I(P)$ within the known values of $I(V) = I(f(V^0))$. Note that similar deformation computations are employed by common elastic registration techniques [29], [30].

As seen in Fig. 3(b), the major disadvantage of the approach above is that the deformed voxels V no longer lie on a regular-grid structure. Consequently, computationally-expensive scattered-data interpolation techniques are needed. Another disadvantage is the need to transform the entire voxel volume from V^0 to $f(V^0)$ for each image frame. This is not practical. Indeed, the interpolation step does not demand the entire volume, since only the voxels *near* the image pixels have an effect on their intensity values $I(P)$. Therefore, it is theoretically possible to compute only the deformation of such nearby voxels—a small subset of V—as required by the particular interpolation technique used. Hence, if this approach is to be used, an effective way of identifying this subset is needed. Determining computational bounds for such a method is difficult, since this subset is not fixed and it changes with both the deformation and the image location.

Due to the above disadvantages, the following approach of first mapping the image pixel locations back to the predeformed configuration and then interpolating the regularly-spaced $I^0(V^0)$ at these *undeformed* pixel locations is proposed in this paper. For an invertible deformation f, the pixel locations P can be mapped to the reference volume as

$$P^0 = f^{-1}(P). \tag{2}$$



(a) (b)

Fig. 4. Mesh-based image *undeformation* illustrated in 2-D: (a) image slice within a mesh that is under force/displacement constraints and (b) corresponding image pixels mapped to the nominal mesh, where the reference volume is given on a regular-grid structure.

An illustration of such *undeformed* pixels with the reference volume voxels V^0 can be seen in Fig. 3(c). Subsequently, the pixel intensities $I^0(P^0)$ at these nominal pixel locations are interpolated from $I^0(V^0)$.

With this method, the required interpolation is on a regular-grid of known values, which enables the use of well-studied simple and fast interpolation techniques. Furthermore, as opposed to the former approach, the inverse deformation needs to be computed only for the image pixels, which yields a fixed number of computations for the synthesis of each frame regardless of the deformation and the image location.

B. Image Pixels in an FEM Tessellation

Many deformation models employ a mesh to simulate displacements during deformation. One of the most common methods for tissue deformation computation is the FEM, in which tessellations are typically much coarser than medical imaging resolutions due to computational constraints. Mesh-based models simulate deformation in the form of node displacements of the overlaid mesh, so the deformation f is only defined at the nodes. The displacement of other locations within the mesh can be approximated from these node displacements. For example, for tetrahedral meshes, barycentric coordinates can be used for this purpose. Consider the situation in 2-D, where an image P slices an object deformed under constraints shown with the arrows in Fig. 4(a). For each image pixel P_i ($i \in 1 \ldots n$), once its barycentric coordinates with respect to the deformed element e_s enclosing this pixel are found, they point to the corresponding P_i^0 in the predeformed configuration (details in Section III-D). Accordingly, for a computed mesh deformation and a given probe position/orientation, our image frame synthesis consists of the following basic steps:

```
For each $i \in \{1..n\}$
  I.   Find the element $e_s$ enclosing $P_i$
  II.  Find the location $P_i^0$ using its barycentric coordinates
       and the node displacements of $e_s$
  III. Interpolate the given data $I^0(V^0)$ at $P_i^0$
```

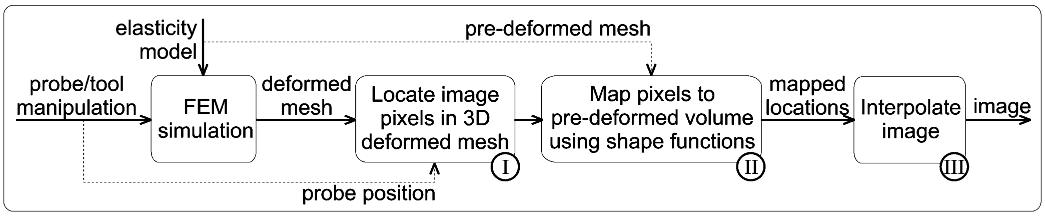


Fig. 5. Basic steps of the proposed algorithm.

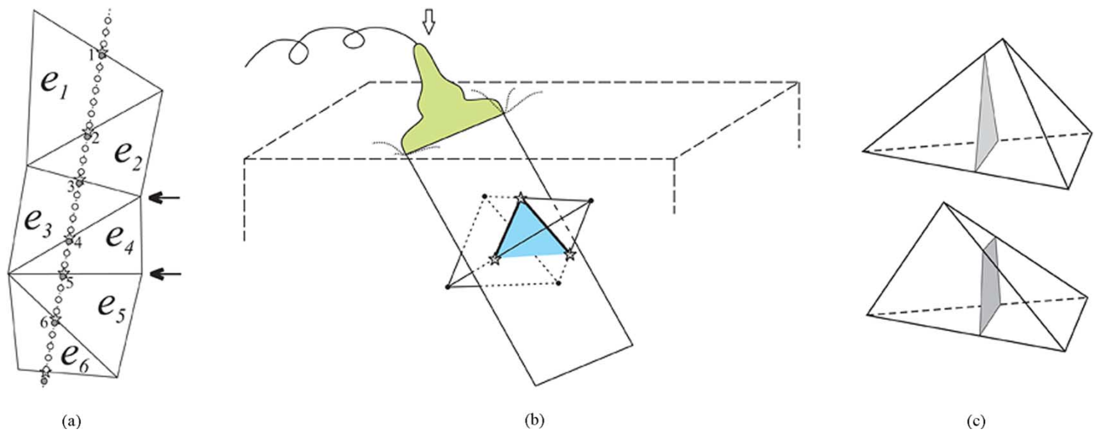


Fig. 6. (a) Marked image pixels close to projections of element borders (stars) on a 2-D mesh, (b) a tetrahedral element intersecting the image plane, and (c) two possible configurations for a planar cross section of a tetrahedron with either 3 or 4 edges intersected.

Fig. 5 presents a flowchart showing the data flow between these steps and the data interaction with the deformation model.

Steps II and III above are constant-time operations, for which there exist well-studied fast implementations. However, the *point location* problem of step I is computationally demanding and hence the bottleneck of this technique. Note that the enclosing element of an image pixel depends on both mesh deformation and the image position/orientation; hence, it needs to be computed at each time instant t and cannot be decided offline.

In computational geometry, the 2-D point location problem has been studied extensively [31], resulting in common techniques such as *slab decomposition* [32], *Kirkpatrick's algorithm* [33], and *trapezoidal maps*. However, only a few of these methods extend to higher dimensions, i.e., point location in 3-D spatial subdivisions. Besides, most techniques focus on either locating a single point or a set of points scattered over the given domain, whereas the points in our case—the image pixels—are regularly-spaced on a 2-D planar surface that is embedded in 3-D. Consequently, exploiting this property of our problem to build intermediate data structures for locating all image pixels cumulatively will accelerate the process substantially. Indeed, any conventional method of locating each point individually would not allow real-time processing of typical medical image resolutions. For instance, the point location routine in the QuickHull package [34] locates the 90 K pixels of a typical image presented in our Results section in over 30 s. Therefore, the rest of this paper focuses on exploiting this spatial relationship of image pixels in order to accelerate step I above.

C. Fast Equidistant-Point Location on Image Planes in 3-D

Due to the 3-D mesh elements being much larger than the image pixels, numerous neighboring pixels are enclosed by a single element that is cut by the image. This fact can be exploited to predict the enclosing element of a pixel from its pixel neighbors. Although this yields a significant speed gain, it is still not fully taking advantage of the known grid structure of the pixels. Even though most predictions will succeed, each prediction has to be verified by an operation such as *point-in-tetrahedron check*, requiring many additional operations. For failed predictions, finding the enclosing element is again the same nontrivial point location problem above. Therefore, a temporary data structure is proposed for each frame such that, following the construction of this data structure, each and every pixel is located accurately and immediately, i.e., in constant time.

Consider the faces of the mesh elements intersecting the image plane. Note that the mesh–image intersection is the only information needed to locate all image pixel points. Indeed, when moving from a pixel to its neighbor, if no intersection is crossed, the latter pixel still lies in the same element, otherwise, it lies in a different element that can be deduced from the intersection information. Thus, using a scan-line approach, we determine to which mesh element each of the pixels belongs by traversing the image. The traversal occurs along a line parallel to an arbitrary axis, e.g., the axial direction of the ultrasound imaging plane.

The conventional slab decomposition technique for point location, also called the partitioning scan-line algorithm, locates individual (possibly scattered) points in a domain using edge comparisons along a scan-line. In contrast, in our case of an image domain, pixels are positioned at discrete locations, and therefore scan-lines only sweep discrete columns. In this paper, this discrete pixel structure is utilized to efficiently store the mesh intersection information such that the image intersections with mesh faces are discretized at the pixel locations and are stored for use during line scans. Such a data structure that sub-

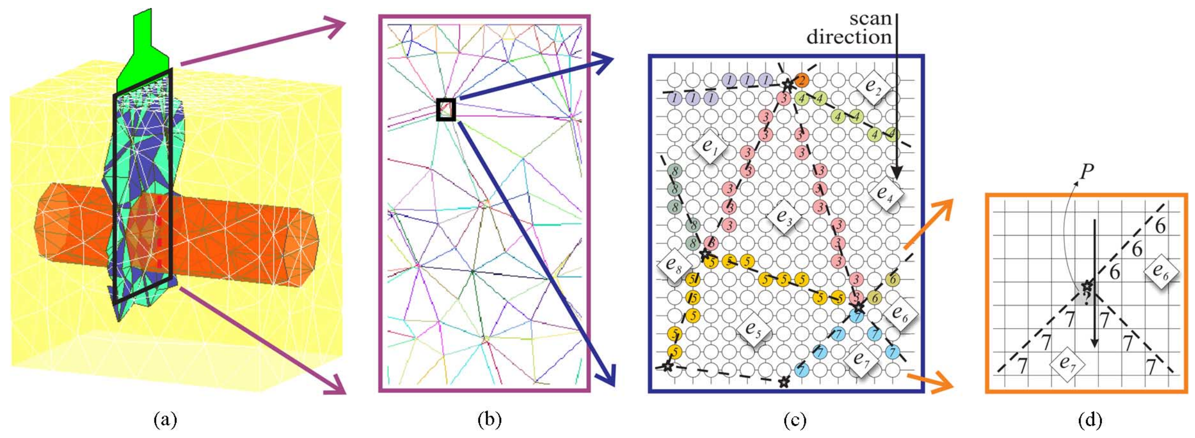


Fig. 7. (a) Mesh and cross sections of the elements sliced by the image simulated for an ultrasound probe, (b) element boundaries discretized and marked on the image prior to the interpolation, (c) a close-up to this marked image, and (d) a pixel of conflict, that is to be marked according to the scan direction.

stantially accelerates the overall image generation procedure is built temporarily prior to every frame being synthesized.

Let us demonstrate a simple 2-D case in Fig. 6(a). Assume that the pixels just below the mesh intersections, shown with stars, are identified and marked with the corresponding element numbers as depicted. For a downwards traversal (scan) of the image, it is inherent that any pixel encountered on and after a marked pixel is guaranteed to be in that marking element until another marked pixel is encountered. Implementing such a data structure to store the discretized mesh intersections (shaded pixels) requires at most $O(n)$ storage. Although in practice these marked pixels will be a small fraction of the image pixels, allocating an array the size of the image is computationally more efficient and is not demanding on today's computers. So, the discretized mesh intersection information can also be seen as an added property to each pixel, specifying whether it is being intersected by a face and, if so, by which element's face.

Finding the pixels to mark in a 2-D case, e.g., the darker circles in Fig. 6(a), involves solving for line–line intersections of element edges with the image. Similarly, the intersections of 3-D element faces with a planar image can be found in 3-D using the deformed-mesh node positions and the image plane equation. However, further processing is required to discretize and mark them on image pixels.

Tetrahedral elements are chosen for presentation in this paper due to their common use in tessellations. A tetrahedral element intersecting the image is shown in Fig. 6(b). Note that a tetrahedron may intersect a plane in one of two possible configurations yielding a triangular or a quadrilateral cross section as illustrated in Fig. 6(c). Using the deformed node positions and mesh connectivity, the element edge intersections shown with stars in Fig. 6(b) are found easily solving line-plane intersections. Subsequently, the line segments connecting these stars need to be discretized and marked on the corresponding pixels. Recalling our downwards scan-line direction, only the *upper* line segments [shown with darker lines for the instance in Fig. 6(b)] need to be marked. This is due to the fact that the top-half of any cross section has to be first crossed by the scan-line in order for it to reach the interior of the cross section. Similarly, once leaving this cross section, the top portion of the next element below will be crossed indicating that the upcoming pixels do not belong to the previous element anymore. Therefore, only marking the edge of elements in the direction of an incoming scan-line is sufficient.

Let us demonstrate this pixel marking for a tetrahedral mesh sliced by an ultrasound image plane in Fig. 7. The 3-D view in Fig. 7(a) shows the element cross sections. The discretization of these face intersections on the image pixels, which is the above-mentioned intermediate data structure, is seen in Fig. 7(b). Part of this structure is illustrated enlarged in Fig. 7(c), where the actual face intersections are depicted with dashed-lines and their pixel discretizations with colored circles. The element numbers marking these pixels are also labeled in the figure.

Bresenham's line drawing algorithm is used to discretize these segments on the grid of image pixels [35]. Note that it is possible for more than one intersection segment to be discretized on the same pixel. Although this is more likely to occur at the corners of the polygons, it can extend to more pixels of a line depending on the relative slopes of neighboring line segments. For instance, consider the central pixel P shaded in Fig. 7(d), which is involved in the discretizations of both the elements e_6 and e_7. If this pixel were to be marked as belonging to e_6, then our line scan would mislocate any pixel in the column below P. Such discretization conflicts, where the same pixel is involved in the discretization of more than one edge intersection, may occur at a substantial number of pixels and each may involve many edges. Therefore, a mechanism is required to ensure that such pixels are marked correctly. In this paper, we resort to a method of topologically sorting all the tetrahedra cross sections prior to marking them. The details are explained in Appendix A.

D. Finding the Pixel Intensity Value

Each 3-D pixel location is mapped to the reconstructed image voxel volume, where its intensity can subsequently be interpolated. The deformation part of this mapping is addressed using the barycentric coordinates with respect to the enclosing element. The overall mapping from the discrete image plane coordinates to the reconstructed volume coordinates can be expressed as a combination of linear transformations as shown in Fig. 8. In this figure, T_{VP} is the image-to-mesh transformation, which is determined by the probe position/orientation and

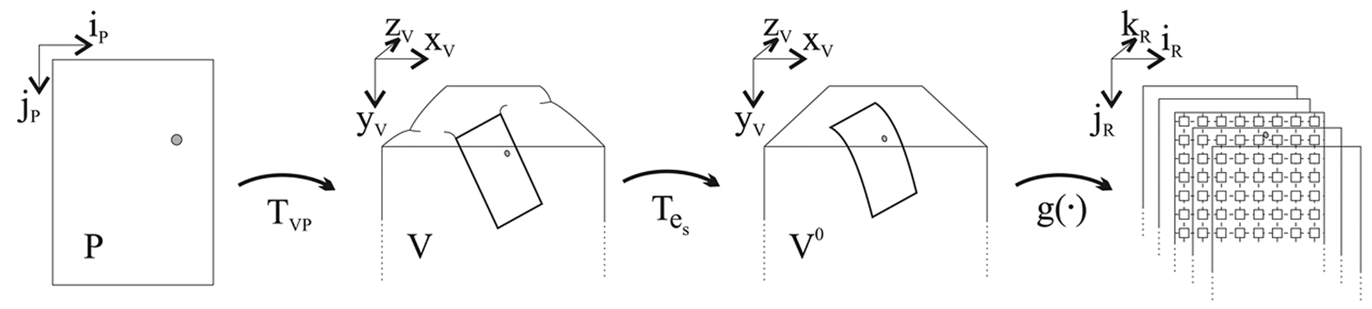


Fig. 8. Transformations that map a pixel to the reference voxel volume, where an interpolation $g(\cdot)$ finds its intensity value.

hence is constant for all pixels of the same image frame. T_{e_s} is the deformed-to-nominal mesh transformation defined by the node displacements of the enclosing element e_s due to the deformation. This transformation is constant within an element for the mesh-node displacements of that given instant and it is calculated using barycentric coordinates [36], as described in Appendix B.

Note that $T_{\rm VP}$ changes with probe motion, whereas T_{e_s} with deformation. Consequently, a cumulative transformation

$$T_s = T_{e_s} T_{\rm VP} \tag{3}$$

can be computed for any given element e_s during the image synthesis of a given probe position/orientation within a known/simulated deformation. Since any pixel within that element is subject to this same transformation, it is calculated only once per intersected element per image frame.

E. Summary of the Proposed Algorithm

For the synthesis of every frame, first the set of elements that are intersected by the imaging plane is compiled. This set L is composed by traversing the elements in the plane using a 3-D mesh element neighborhood list, which is precompiled offline as soon as a 3-D mesh is available. Note that the neighbors of an element do not change with deformation. Therefore, given an intersected element, such a list enables us to deduce its neighbors that are also intersected by observing which face of the current element intersects the image plane. The set L is then sorted topologically and the top-halves of the element projections are discretized and marked on the image in that sorted order using Bresenham's algorithm. Fig. 7(b) demonstrates an instance of marked image pixels. The intensity of a pixel P_i is found by interpolating $T_s P_i$ in the reference voxel data. This interpolation is denoted by the operator $g(\cdot)$ in Fig. 8. For an example of using the nearest-neighbor interpolation (NNI), if we assume that $\mathrm{round}(\cdot)$ function gives the nearest discrete reference-grid location to a point, then $I^0(\mathrm{round}(T_s P_i))$ is the intensity of the pixel sought.

The transformations for all the intersected elements are computed while the set L is compiled. Subsequently, for every edge crossing of the scan-line, a *current transformation* pointer can be switched to point to the transformation matrix of the current enclosing element. Our proposed algorithm for the synthesis of one image frame can be summarized as follows:

```
1. Compile the set $L$ of intersected elements and their partial ordering relations.
2. Sort $L$ topologically.
3. Mark the cross sections of $L$ on the image in the sorted order.
4. Compute the transformation $T_s$ for each element $e_s \in L$
5. For each image pixel $P_i$
    5a. If $P_i$ is marked with element $e_s$, then set the active transformation $T$ to $T_s$
    5b. Find the intensity of $P_i$ by interpolating $I^0$ at $TP_i$
```

F. Computational Analysis

In the algorithm above, the loop in step 5 processes every image pixel, while the previous steps are used to compile the intermediate data structures to accelerate this step. Consider an n-pixel image intersecting a total of m elements in a 3-D mesh and recall that $n \gg m$ for typical medical images and FEM tessellations. Note that the synthesis of an image requires some form of processing of each of its pixels, thus the computation of any synthesis algorithm has a lower bound of $\Omega(n)$. Nevertheless, the individual computational cost for every pixel may render an algorithm infeasible as demonstrated in Section III-B. Indeed, a method that is suboptimal for our application may demand over 30 s just to locate pixels in mesh elements. The computational analysis of the methods and the data structures introduced in this paper are presented below.

Compilation of the intersected element set L starts from an arbitrary initial intersected element, such as one touching the probe. Finding this initial element, which can be done simply by traversing the top surface of the mesh, takes insignificant time. Once one intersected element is found, the traversal of the rest using a precompiled neighborhood list in step 1 takes constant time per element. Computing each transformation in step 4 is also a constant time operation per element. Topological sort takes linear time with respect to the total number of cross sections and partial relations. Nonetheless, in our case the relations are set by the shared edges between the m cross sections and thus the number of such edges is bounded from above by a multiple of m. As a result, all the steps 1, 2, and 4 compute in $\mathcal{O}(m)$ time.

The computation of step 3, where the pixels on cross sections are marked, depends on various implementation choices, such as the specific line drawing algorithm. Nevertheless, it can be approximated by the number of pixels actually marked on the image. This can in turn be regarded as approximately $\sqrt{m}$ rows of elements being marked by Bresenham's algorithm on their upper halves. Considering a row of elements has on the order of

$\sqrt{n}$ pixels to be marked, step 3 thus requires $\mathcal{O}(\sqrt{nm})$ time to compute.

Note that the significantly lower computational order of steps 1–4 justify the anticipated speed gain during step 5. In particular, step 5a of identifying the enclosing element reduces down to a single memory access and, in step 5b, the transformation of pixels by T_s from discrete 2-D image coordinates to the 3-D reconstructed volume requires only 12 multiplications per pixel.

G. Deformation Model

The deformation model employed in our particular implementation is a linear-strain quasi-static finite-element model. For this, first a mesh representation of the region of interest is obtained. Using the Young's modulus and Poisson's ratios of the materials involved, a stiffness matrix $\mathbf{K}$ relating the nodal displacements u of this mesh to the nodal forces f is compiled such that $f = \mathbf{K}u$ [36]. Fundamental boundary constraints (the fixed nodes of the mesh) are next applied on this $\mathbf{K}$ by zeroing its corresponding rows/columns. $\mathbf{K}$ is then inverted and saved for use in the online simulation, where the probe surface is applied as displacement constraints on the nodes that are in contact with it so that the displacements u of all nodes are calculated at each iteration. Modeling such contacts and achieving conforming meshes with contact surfaces is an active field of research with various approaches having been proposed for different deformation models (e.g., local mesh refinement [36], multiresolution meshes [37], condensation [3], force coupling [6]). In our experiments, it was assumed that the contact locations were known *a priori* so that the mesh was generated ensuring that nodes do exist on those interfaces.

IV. Results

For our experiments, a $60 \times 90 \times 90$ mm tissue-mimicking gelatin phantom, having a soft cylindrical inclusion of 25 mm in diameter, was constructed. The phantom was meshed using the GiD meshing software [38] yielding 493 nodes and 1921 tetrahedra. The elasticity parameters for the FEM simulation were set to the approximate values known for the gelatin concentrations used. This phantom was imaged using a SonixRP ultrasound machine from Ultrasonix Medical Corporation with a linear probe mounted on a precision motion stage. Vertical parallel slices with 1 mm separation were acquired with care not to deform the phantom surface. The dimensions and the mesh of our phantom and our imaging setup are seen in Fig. 9.

Images, that physically span an area of 37.5×70 mm, were acquired at a resolution of 220×410 from the display pipeline of the ultrasound machine. This is a typical *B*-mode resolution that this ultrasound machine outputs to the screen for the given probe and default imaging parameters. For our experiment, 75 images were collected at a 1 mm interval while moving the probe in its elevational axis. Accordingly, our reconstructed voxel volume is chosen to be the collection of these parallel slices, which constitutes an average of 8750 voxels per element in our particular phantom mesh.

Deformation was applied by indenting the phantom with the probe. It was simulated using the FEM with the bottom side of the phantom being the fixed fundamental displacement boundary constraint. The probe indentation was applied as a displacement constraint to the mesh nodes coinciding with the probe on the top surface. In the FEM, the Young's moduli were set to 15 KPa and 5 KPa for the background and the circular inclusion, respectively, and a Poisson's ratio of 0.48 was used for both. The ultrasound images were then synthesized using the techniques presented. Some of the acquired and the simulated images are presented in Fig. 10.

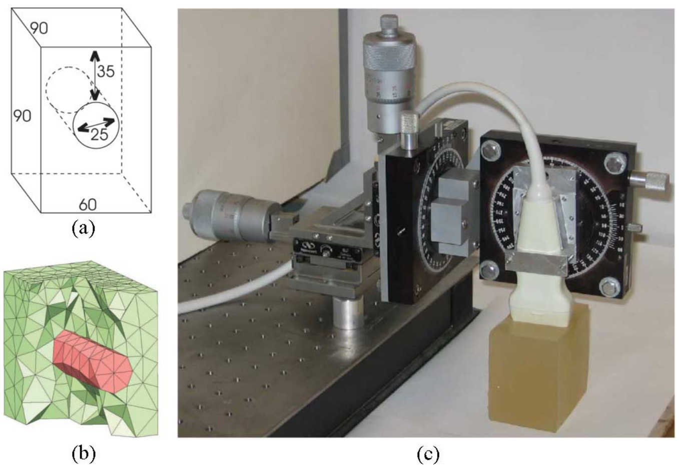


Fig. 9. (a) The phantom design (in millimeters), (b) its mesh, and (c) the image acquisition setup.

Images of a simulated probe indentation were compared to the images during an identical physical indentation experiment using their *mutual information* (MI) and sum of squared differences (SSD). Eleven images were both simulated and acquired in 1 mm steps up to a 10 mm compression. The MI and SSD between each pair of simulated and acquired images are presented in Fig. 11. In this paper, to present these MI values as a ratio of an absolute measure in our experiments, the values were normalized with the average MI of two images acquired 1 mm apart in the elevational direction. As seen in Fig. 11(a), the simulated images and the acquired images for the same indentations have the highest MI and the lowest SSD, as expected. This shows that the simulation can successfully synthesize an image closely resembling a real one. For reference, in our phantom the average MI between an ultrasound image and one that is shifted vertically by one-pixel is 117% of the average MI of two images 1 mm apart (forming the normalization factor) and it is 67% between two acquired images with 1 mm probe indentation, when normalized as defined above.

The method can generate slices of given 220×410 resolution using nearest-neighbor NNI in less than 13 ms on a 1.87 GHz Pentium computer. Using trilinear interpolation (TLI), the simulation of a *B*-mode frame on the same computer takes approximately 25 ms. In order to evaluate the effect of the number of image pixels n and the number of intersected elements m on the computational speed, the frame synthesis time for the NNI was measured using different image parameters. As presented by the computational analysis, a linear dependency on n and a negligible effect of m were expected. First, the image resolution was decreased at 10% decrements of the original resolution while keeping the physical span of the image frame, and then the physical image span was decreased at 10% decrements of the original span while keeping the number of pixels the same. Note that,

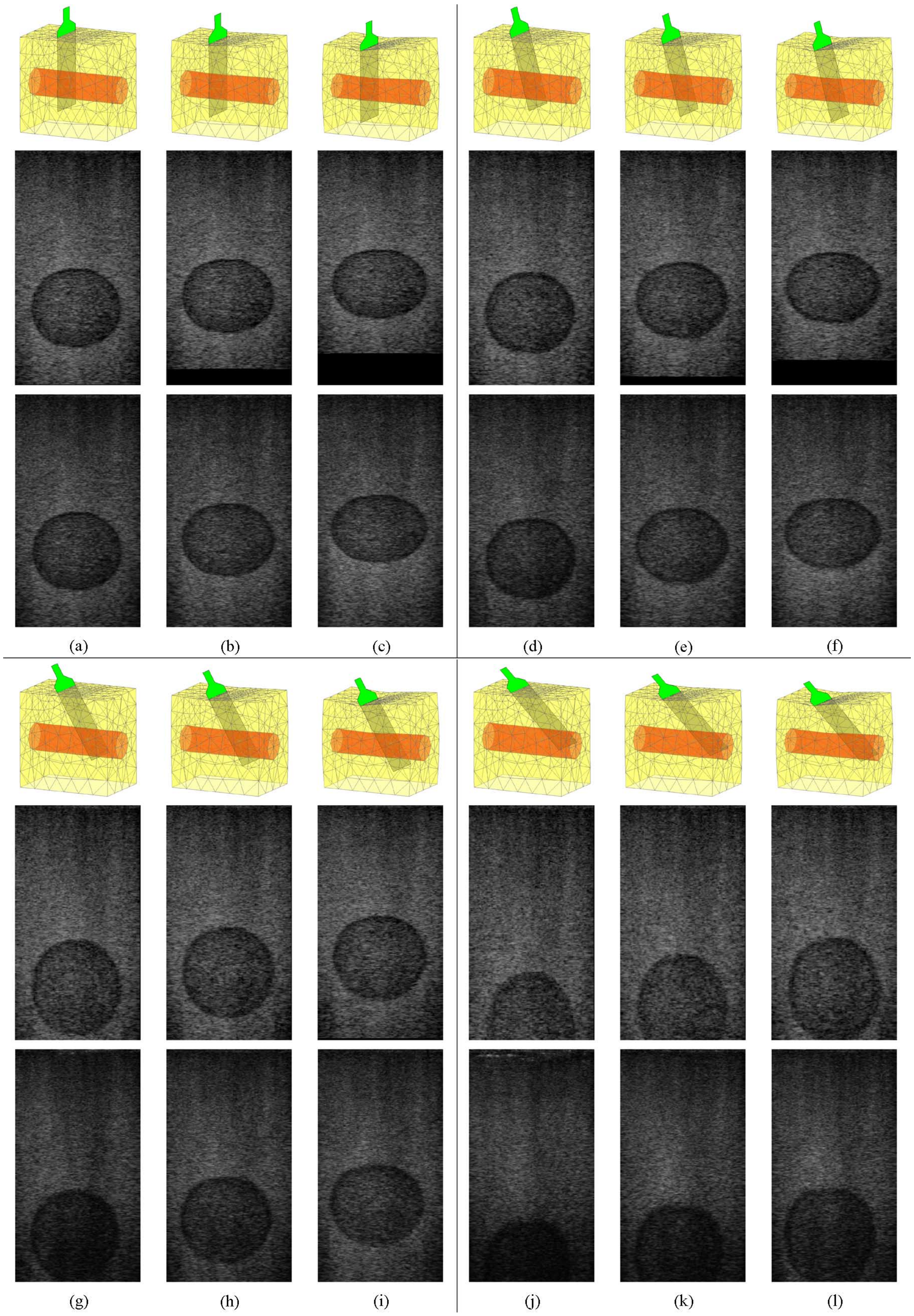


Fig. 10. Simulated (upper) and acquired (lower) images with 0, 5, and 10 mm indentations for probe tilted at (a)–(c) 0°, (d)–(f) 15°, (g)–(i) 30°, and (j)–(l) 45°.

effectively, the former alters n for a constant m and the latter alters m for a constant n. Consequently, the former decreases the pixel density per element, i.e., the number of pixels falling into each element cross section whereas the latter increases it. As presented in Fig. 12, the number of pixels n was observed to determine the speed in the $\mathcal{O}(n)$ manner as expected, whereas the number of intersected elements m exhibited little effect on speed. Note that the slope of this linear dependency on n is de-

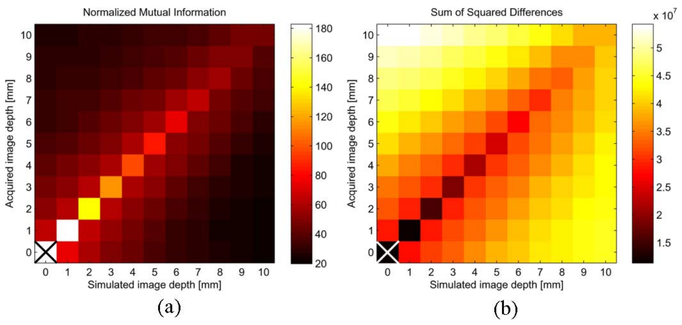


Fig. 11. Normalized mutual information (a) and sum of squared differences (b) of the images simulated and acquired at 1 mm incremental probe indentations (the values at $[0, 0]$, which match perfectly, are not presented here to preserve the color map).

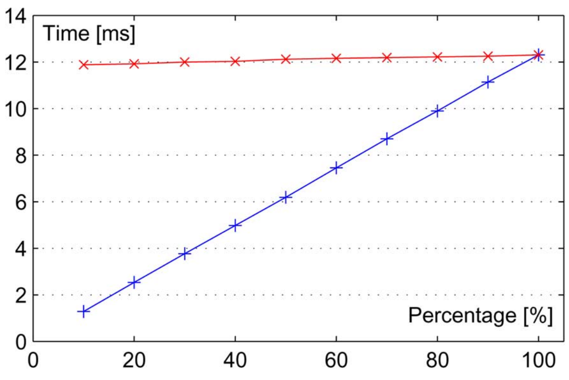


Fig. 12. Change of frame synthesis time when varying the number of pixels n $(+)$ and the number of intersected elements m (x) expressed in percentages of the full-size phantom images presented in this paper.

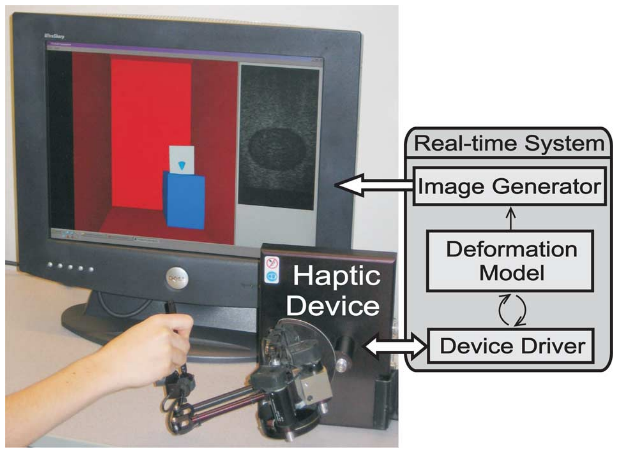


Fig. 13. Real-time ultrasound scanning simulator.

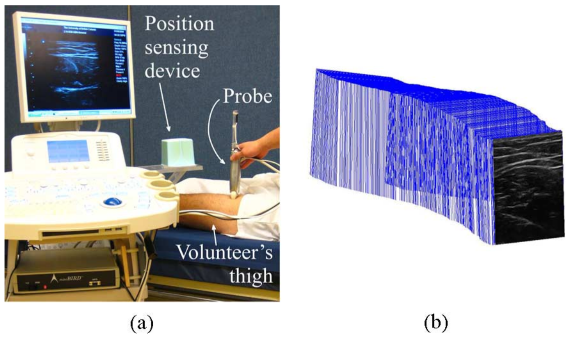


Fig. 14. (a) Setup for *in vivo* data collection and (b) the spatial arrangement of the collected ultrasound images.

fined by the hidden cost of processing each pixel (step 5). Our approach of introducing additional steps and data structures in order to reduce this hidden cost to a few multiplications is the key to accelerating such a method.

The technique above has been implemented for interactive ultrasound visualization with simulated deformation as seen in Fig. 13. In this system, a SensAble Phantom Premium device mimicking the probe is manipulated by the user while a visual interface displays the tissue mesh and the probe in 3-D. At the same time, the ultrasound images are also synthesized by our algorithm and displayed at real-time visual rates (over 40 Hz even when additional computation time for FEM simulation and haptic feedback are added on top of the 13 ms/frame image synthesis time). A simple feedback force normal to the nominal tissue surface and dependent on the current indentation depth was applied to user's hand. The probe indentation was modeled as displacement constraints on the closest surface tissue nodes and the mesh deformation was computed using a precomputed inverse stiffness matrix.

For an *in vivo* assessment of this simulation method, *B*-mode images of the thigh of a volunteer were also simulated. The 3-D volume was reconstructed from *B*-mode images acquired by the same SonixRP ultrasound machine. The spatial positions of these images were recorded using a magnetic position sensor, Ascension MiniBIRD, attached to a linear ultrasound probe so that a freehand scan can be conducted. The anterior upper thigh of a volunteer was then scanned as seen in Fig. 14(a) to a depth of 45 mm by translating the probe orthogonal to its imaging plane similarly to the phantom experiment. The spatial arrangement of the collected images can be seen in Fig. 14(b). The volume represented by these images is then resampled on a regular grid of 0.1 mm spacing using the Stradwin software [26].

Considering the thigh tissue locally, the femur is the fixed displacement boundary condition, that plays a major role in the way the thigh deforms. The femur anterior surface was segmented from a deeper ultrasound scan of the same thigh region. An FEM mesh of 3805 elements and 997 nodes was generated using the GiD software and displacement boundary conditions were defined by spatially fixing the nodes on the femur surface. For the purpose of this paper, all soft tissue was given a fixed Young's modulus of 15 KPa and a Poisson's ratio was set to 0.48. Ultrasound images of two different probe orientations were simulated using tri-linear interpolation. Simulated images at different indentation depths are seen in Fig. 15, where the anterior thigh anatomy is observed to deform under probe pressure.

V. Discussion

In the literature, there have been *in-plane image deformation* strategies for image registration, deformation correction for volume reconstruction [10], [25], and a training simulation for DVT [18], [23]. However, these 2-D approaches cannot simulate out-of-plane deformations. Even if the deformation is driven by motion only in the imaged plane, the boundary conditions may couple image plane motion with motion orthogonal to the imaged plane, and such methods simply cannot account for motion orthogonal to the imaging plane. Indeed, consider the case of brachytherapy, where the prostate is imaged with the transversal crystal of the endo-rectal probe while the needle may push the prostate through the imaging plane [see Fig. 1(c)]. For such deformation induced orthogonally to the imaging plane, 2-D approaches simply do not apply and our method becomes essential.

Our choice of phantom geometry aims at presenting out-of-plane deformation. During the indentation experiments with the probe tilted, not only do the phantom structures compress, but also the image plane moves through the volume in its elevational axis.

Our simulation method is not limited to linear imaging geometry, but can also accommodate other probe geometries such as sector probes. Indeed, regardless of the transducer crystal geometry, the conventional format for displaying, processing, and storing *B*-mode images is Cartesian-based in compliance with the common screen hardware and image processing algorithms. Consequently, our method simulates the pixels on a regular-grid regardless of the original data acquisition format, similarly to other interpolation-based implementations in the literature. The simulation of a sector image, for instance, can be accommodated in the current technique by simulating all pixels internal to the bounding box of the nonrectangular image footprint and then masking off the pixels lying outside the actual sector image region.

Locating the points of a 3-D grid was studied in [39] to resample a 3-D mesh to use in the finite difference method for simulating seismic wave propagation. However, note that we are interested in a 2-D plane that intersects the mesh in arbitrary orientation. Furthermore, our case involves deformed meshes and registering an image plane manifold between different mesh configurations.

A deformation simulator must have two components: one that predicts how tissue deforms based on a physically valid model and one that can display the result at a high enough frame rate to match the simulation context, i.e., be "real-time." Considering the former, many medical simulations employ the FEM to produce the mesh deformation in response to tissue forces or changes in the tissue constraints. Coupled with this FEM simulation, the user of a medical simulator should be able to examine the simulated tissue in a manner that is accurate and real-time. The techniques presented in this paper serve to this latter need.

It is important to realize that our presented method is independent of the FEM simulation and does not introduce any additional approximations and errors beyond those intrinsic to the FEM simulation, i.e., there is no trade-off between image synthesis acceleration and image quality. The image synthesis method uses the same mesh and the same interpolation functions as the FEM simulation, and the resampling is performed over a grid of the same resolution as the original medical image data while keeping the highest possible image resolution given the reference image data set. Considering the *B*-mode image synthesis in particular, the only assumption employed is the isotropy for the ultrasound interaction model. In addition, note that the method introduced for image pixel location gives an exact solution, not an approximation. Such a method is superior to the previous techniques in locating regularly-spaced points at the expense of a small additional storage and its initialization time.

Our method aims at the development of a real-time realistic image synthesis technique, which can be integrated with training systems that involve tissue deformation. In this context, the simulated images were found to be adequate for medical simulators by physicians following a preliminary visual inspection. This will be assessed further by expert sonographers in specific clinical procedures, such as prostate brachytherapy, in the future.

As explained in Section III-F and justified by the results in Fig. 12, the frame synthesis time of our simulation depends mainly on the number of pixels, but is also affected by the number of elements intersecting the image to a small degree. Nevertheless, note that it is independent both from the entire physical span of the FEM mesh and from the total number of nodes/elements in it. It is indeed further independent from the entire physical span of the reconstructed image data set, assuming it fits in the computer memory. Consequently, much larger tissue representations can be handled successfully. Also note that the extent of the reference image volume does not have to match the extent of the FEM mesh. For instance, in our experiment the bottom part of the phantom was not imaged for the reference volume due to the depth setting of the probe and hence the black regions appear at the bottom of the simulated images when indentation is applied. However, the deformation simulation was considering the entire phantom volume for an accurate simulation.

Observing the relatively small effect of the number of intersected elements on synthesis time in Fig. 12, it is seen that more elements on the plane will not be detrimental to the simulation. Consequently, techniques such as local mesh refinement will not impede the speed, since $n \gg m$ is still valid. Indeed, one can analyze the data on such a figure to estimate maximum image resolution for a given mesh simulated on a particular machine.

In general, 3-D ultrasound reconstruction involves resampling the acquired images in order to generate volume data on a grid-structure. In our phantom image collection, ultrasound slices were parallel and relatively dense creating a regular grid, therefore no further processing was needed for reconstruction. For the *in vivo* data, the Stradwin software was used to resample the data in a grid structure. In this paper, the nearest-neighbor and tri-linear interpolations were employed in the image synthesis step. Other interpolation schemes are also applicable depending on the computational limitations for achieving a required frame-rate in a particular implementation.

Note that, for significantly large deformations, the acquired and the simulated images may not match exactly using the linear-strain approximation due to rotationally-variant linear elements. Nevertheless, there exist numerical treatments in the literature to achieve rotational invariance [40]. Nonlinear elasticity with dynamic models has also been proposed in the literature [4]. All these models provide mesh node displacements at given time instants, which can be used by our technique to synthesize images as in the given implementation. Moreover, even other deformation models that provide node displacements, such as finite differences or spring networks, can be used since the barycentric transformation proposed does not assume, nor is bound to, any property of the FEM.

We assume that linear interpolation is used within the FEM mesh, therefore four-node tetrahedra were employed in our simulation. The interpolation accuracy is maintained by reducing

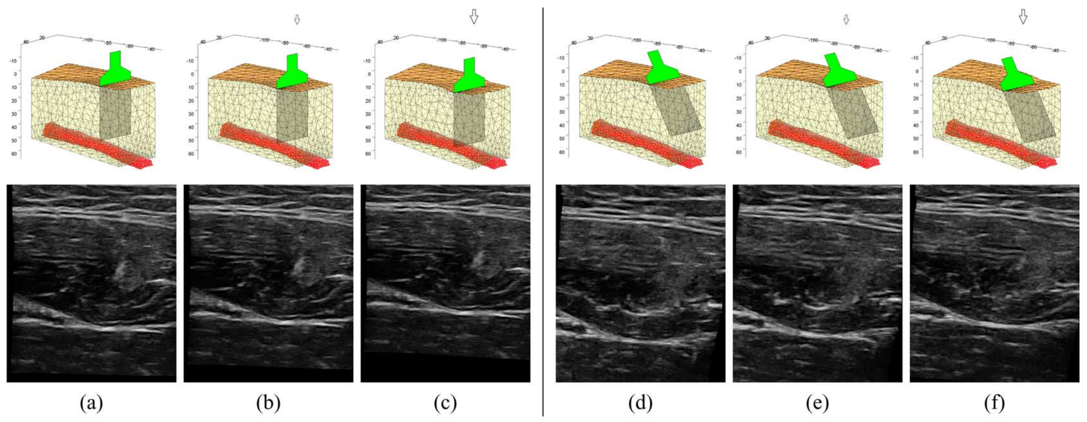


Fig. 15. Simulated ultrasound images of the thigh with 0, 2, and 5 mm indentations (shown with arrows) for probe tilted at (a)–(c) 0° and (d)–(f) 30°.

the size of the mesh elements as necessary. The presented techniques still apply to deformations computed by higher-geometry elements, such as ten-node tetrahedra, by using only the corner nodes for image synthesis. Considering element geometries other than tetrahedra, e.g., hexagons, or meshes with mixed elements used for FEM deformation, note that the image plane cross sections of such convex polyhedra are also convex. Thus, these cross sections can still be found and marked using the techniques presented. However, an alternative undeforming transformation will be needed instead of using the barycentric coordinates.

Our phantom has the same cross section longitudinally, therefore two 1 mm apart images look (and should look) the same to the human eye. Although their speckle pattern may differ, for an observer they both carry the same information about the location, size, and other features of the inclusion. The MI of this minimal and practically in-differentiable speckle pattern change is taken as the base for normalization.

Note that, once the image is marked, the processing of each and every scan-line is independent of another. This allows for completely disjoint synthesis of each scan-line, which is a significant advantage for parallel processing of the computationally intensive step 5. For instance, on the emerging multicore CPU architectures, scan-lines can be distributed among CPU cores. Moreover, the simple computation scheme of step 5 is highly aligned with the SIMD (single instruction, multiple data) execution model for parallelization on recent GPU computation/programming technologies, such as CUDA [41].

The approach presented in this paper is an enabling technique for many medical simulations and in particular for any ultrasound simulator that will allow for tissue deformation. Fast image simulation is also essential for certain deformable registration techniques, where image slices of a deformed volume are compared to a set of reference images while the deformation constraints are being optimized iteratively. Studying accelerated image synthesis is of significant value especially in such registration schemes, where the generation of sliced images through mesh-based deformations is one of the computationally intensive steps. Faster synthesis is also crucial for the processing of high resolution images. A faster algorithm allows for the generation of possibly more than one slice at a time instant (such as to render the volume in 3-D) and also facilitates the presentation of more than one modality to a trainee during a training simulation (such as additional deformed MR and/or CT to better understand the anatomy scanned).

This paper treats the problem of image simulation with deformation using an interpolative approach due to real-time processing limitations and parametrization difficulties of generative models. An ultrasound image is indeed a product of very complex wave interactions in the tissue. In an interpolative approach, instead of attempting to model this complex wave behavior, image features (and similarly artifacts) are reproduced and placed in simulated images at the 3-D location where they were originally acquired. Similarly, our technique assumes that the *B*-mode image of a deformed tissue region is similar to the properly deformed version of a *B*-mode image taken prior to deformation. However, the visibility of an actual image feature (e.g., a sharp tissue interface) or an artifact in *B*-mode imaging may depend on the direction of the incidence of the ultrasound beam, whereas the visibility, with an interpolative approach, depends on the direction of the original data collection regardless of the simulation-time probe orientation. Therefore, interpolative *B*-mode techniques may not be optimal or applicable for certain tasks. Nevertheless, they are still of great value in the medical field and are employed in various *B*-mode applications such as volume reslicing in commercial 3-D ultrasound machines. Our approach equips the user with a fast and powerful tool for a range of applications from training simulators to fast deformable registration, after carefully considering the interpolative nature of the simulation method and its related limitations for that particular application. Note that, as opposed to ultrasound, for other imaging modalities such as MR, the assumption above may be satisfied.

Our method can be extended for applications and anatomy in which artifacts are prominent by acquiring multiple reconstructions of the same volume where the images are collected at various probe incidence angles. Subsequently, for each frame simulation, the reconstructed volume acquired by the closest probe orientation to the simulated probe will be used. This will effectively maximize the likeness and direction of artifacts in simulation. Note that such a modification introduces additional memory storage cost due to multiple reconstructed volumes; nevertheless, it does not increase the overall computational complexity of the approach.

VI. Conclusion and Future Work

In this paper, a technique for synthesizing planar images in deformed meshes of tissue models was presented. The method uses 3-D image data of predeformed tissue. A pixel enumeration technique originating from common scan-line algorithms was adopted to enable fast identification of mesh elements enclosing each image pixel during deformations given by mesh node displacements. This allows the generation of medical images of considerable size at frame rates that are suitable for real-time applications. This technique was implemented to simulate *B*-mode images of a deformable phantom and anterior thigh of a volunteer. Synthesized images of probe indentations were then compared to the corresponding images acquired by physically deforming the phantom. A real-time ultrasound simulator implemented on a haptic device was also demonstrated. The results show that the proposed method produces realistic-looking *B*-mode images. The methods presented are easily adaptable to other imaging modalities and deformation models.

Appendix A
Resolving Pixel Discretization Conflicts

The correct element for marking a pixel in conflict depends on the scan-line direction. Normally, a pixel on an edge intersection must be labeled by the element which a line scan passing through that pixel is just entering. For instance, in Fig. 7(d), with respect to our chosen scan-line direction, the pixels immediately below pixel P belong to e_7, thus P should be labeled as 7.

To resolve such discretization conflicts, we first sort the cross sections prior to marking them. Discretizing the cross-section edges on image pixels given a suitable order enables each cross-section label to overwrite a previous one by leaving the desired label on the pixel in the end. To formulate such a sorting so that the last value marking a pixel is the correct one, consider a pair of neighboring cross sections. A scan-line passing through both of these will visit first one and then the other, since they are convex polygons. Furthermore, the order will be the same for every scan-line traversing both cross sections. Note that, for a scan-line to correctly locate the pixels below a corner (or, similarly, an edge) shared by these cross sections, those shared pixels need to be marked by the last-traversed cross section. In Fig. 7(d), e_7 is the latter-traversed element cross section. Therefore, e_7's edges have to be discretized after e_6's, consequently, label 7 overwrites label 6.

This traversal precedence criterion is indeed a *partial ordering* relation in-between all the pairs of neighboring element cross-sections in the image plane. Let "$<$" denote this relation such that $e_i < e_j$ indicates that e_j is traversed later than e_i by any given scan-line. Due to the convex cross-section geometry, this relation is transitive such that

$$(e_i < e_j) \text{ and } (e_j < e_k) \implies (e_i < e_k). \quad (4)$$

Therefore, all cross sections can be represented by a *directed acyclic graph* [42], where the intersected elements are the nodes and their traversal precedence orders are the directed graph edges connecting the pairs of neighboring elements. Such a graph can be sorted linearly into a global (total) order that satisfies every given partial order relation. This procedure is called *topological sorting* [42]. Note that there may exist more than one such global ordering, any of which can be used for our purposes. For example, the illustration of cross-sections in Fig. 16(a) has a graph as shown in Fig. 16(b), and one possible ordering of this graph is as follows:

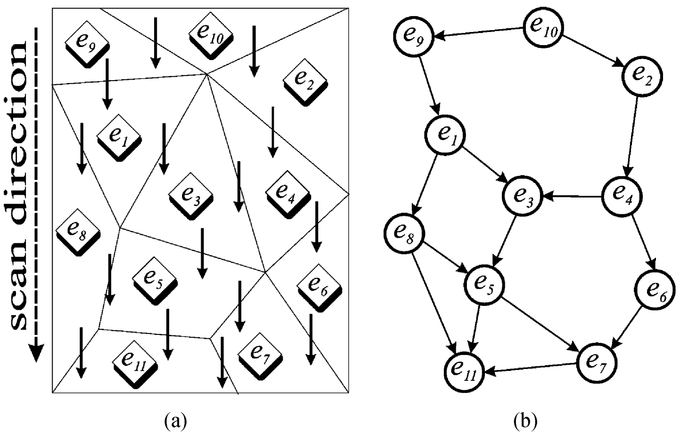


Fig. 16. (a) An illustration of element partial ordering and (b) its corresponding directed graph.

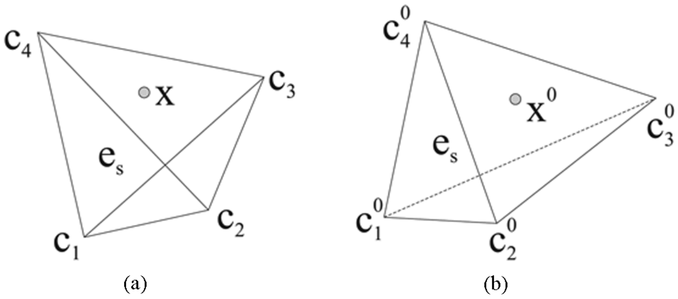


Fig. 17. (a) A deformed element at a time instant t and an image pixel X lying within and (b) the nominal predeformed geometry of this same element with the corresponding undeformed pixel location.

$$e_{10} < e_2 < e_9 < e_1 < e_8 < e_4 < e_3 < e_5 < e_6 < e_7 < e_{11} \quad (5)$$

Appendix B
Undeforming Using Barycentric Coordinates

The 3-D position of a point $\boldsymbol{x} = [x \quad y \quad z]^T$ inside a tetrahedral element e, seen in Fig. 17(a), can be expressed as follows:

$$\boldsymbol{x} = \sum_{k=1}^{4} r_k \boldsymbol{c}_k \quad (6)$$

where r_k are the barycentric coordinates with respect to the element corners $\boldsymbol{c}_k = [x_k \quad y_k \quad z_k]^T$, which are the deformed node positions. This equation can be rewritten for normalized coordinates in the following matrix form:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{bmatrix} \quad (7)$$

$$\boldsymbol{x} = C\boldsymbol{r} \quad (8)$$

so that the barycentric coordinates can be found as $\boldsymbol{r} = C^{-1}\boldsymbol{x}$.

The point $\boldsymbol{x}^0$ seen in Fig. 17(b) corresponding to the same barycentric coordinates in the nominal predeformed configuration of the same element can also be written similarly as $\boldsymbol{x}^0 =$

$C^0\boldsymbol{r}$ for the nominal node positions of $\boldsymbol{c}_k$. As a result, this corresponding nominal location is found as

$$\boldsymbol{x}^0 = C^0 C^{-1}\boldsymbol{x} \tag{9}$$
$$\boldsymbol{x}^0 = T_{e_s}\boldsymbol{x}. \tag{10}$$

REFERENCES

[1] G. Reis, B. Lappé, S. Köhn, C. Weber, M. Bertram, and H. Hagen, *Visualization in Medicine and Life Sciences*. Berlin, Germany: Springer, 2008, ch. Towards a Virtual Echocardiographic Tutoring System, pp. 99–119.

[2] H. Maul, A. Scharf, P. Baier, M. Wüstemann, H. H. Günter, G. Gebauer, and C. Sohn, "Ultrasound simulators: Experience with sonotrainer and comperative review of other training systems," *Ultrasound Obstet. Gynecol.*, vol. 24, pp. 581–585, 2004.

[3] M. Bro-Nielsen, "Finite element modeling in medical VR," *J. IEEE*, vol. 86, no. 3, pp. 490–503, 1998.

[4] G. Picinbono, H. Delingette, and N. Ayache, "Non-linear anisotropic elasticity for real-time surgery simulation," *Graphical Models*, vol. 65, pp. 305–321, 2003.

[5] O. Goksel, "Ultrasound image and 3D finite element based tissue deformation simulator for prostate brachytherapy," M.S. thesis, Univ. British Columbia, Vancouver, BC, Canada, 2004.

[6] J. Berkley, G. Turkiyyah, D. Berg, M. A. Ganter, and S. Weghorst, "Real-time finite element modeling for surgery simulation: An application to virtual suturing," *IEEE Trans. Visualizat. Comput. Graphics*, vol. 10, no. 3, pp. 314–325, 2004.

[7] J. C. Bamber and R. J. Dickinson, "Ultrasonic B-scanning: A computer simulation," *Phys. Med. Biol.*, vol. 25, no. 3, pp. 463–479, 1980.

[8] J. A. Jensen, "A model for the propagation and scattering of ultrasound in tissue," *J. Acoust. Soc. Am.*, vol. 89, pp. 182–191, 1991.

[9] R. W. Prager, A. Gee, and L. Berman, "Stradx: Real-time acquisition and visualisation of freehand 3-D ultrasound," *Med. Image Anal.*, vol. 3, no. 2, pp. 129–140, 1999.

[10] R. N. Rohling, A. H. Gee, and L. Berman, "A comparison of freehand three-dimensional ultrasound reconstruction techniques," *Med. Image Anal.*, vol. 3, no. 4, pp. 339–359, 1999.

[11] R. S. José-Estépar, M. Martín-Fernández, P. P. Caballero-Martínez, C. Alberola-López, and J. Ruiz-Alzola, "A theoterical framework to three-dimensional ultrasound reconstruction from irregularly sampled data," *Ultrasound Med. Biol.*, vol. 29, no. 2, pp. 255–269, 2003.

[12] D. Aiger and D. Cohen-Or, "Real-time ultrasound imaging simulation," *Real-Time Imag.*, vol. 4, no. 4, pp. 263–274, 1998.

[13] H.-H. Ehricke, "Sonosim3D: A multimedia system for sonography simulation and education with an extensible case database," *Eur. J. Ultrasound*, vol. 7, pp. 225–230, 1998.

[14] P. Abolmaesumi, K. Hashtrudi-Zaad, D. Thompson, and A. Tahmasebi, "A haptic-based system for medical image examination," in *Proc. 26th IEEE Eng. Med. Biol. Conf.*, San Francisco, CA, Sep. 2004, pp. 1853–1856.

[15] I. M. Heer, K. Middendorf, S. Müller-Egloff, M. Dugas, and A. Strauss, "Ultrasound training: The virtual patient," *Ultrasound Obstetrics Gynecol.*, vol. 24, no. 4, pp. 440–444, 2004.

[16] M. Weidenbach, F. Wild, K. Scheer, G. Muth, S. Kreutter, G. Grunst, T. Berlage, and P. Schneider, "Computer-based training in two-dimensional echocardiography using an echocardiography simulator," *J. Am. Soc. Echocardiogr.*, vol. 18, no. 4, pp. 362–366, 2005.

[17] SONOFit SG3 ultrasound training system [Online]. Available: http://www.sonofit.com/

[18] D. d'Aulignac, C. Laugier, J. Troccaz, and S. Vieira, "Towards a realistic echographic simulator," *Med. Image Anal.*, vol. 10, pp. 71–81, 2006.

[19] Y. Zhu, D. Magee, R. Ratnalingam, and D. Kessel, "A virtual ultrasound imaging system for the simulation of ultrasound-guided needle insertion procedures," in *Proc. Med. Image Understand. Anal.*, 2006, pp. 61–65.

[20] D. Magee, Y. Zhu, R. Ratnalingam, P. Gardner, and D. Kessel, "An augmented reality simulator for ultrasound guided needle placement training," *Med. Bio. Eng. Comput.*, vol. 45, pp. 957–967, 2007.

[21] A. Hostetller, C. Forest, A. Forgione, L. Soler, and J. Marescaux, "Real-time ultrasonography simulator based on 3D CT-scan images," in *Proc. 13th Med. Meets Virtual Reality*, 2005, pp. 191–193.

[22] F. P. Vidal, N. W. John, A. E. Healey, and D. A. Gould, "Simulation of ultrasound guided needle puncture using patient specific data with 3D textures and volume haptics," *Comput. Animat. Virtual Worlds*, vol. 19, pp. 111–127, 2008.

[23] D. Henry, J. Troccaz, J. L. Bosson, and O. Pichot, "Ultrasound imaging simulation: Application to the diagnosis of deep venous thromboses of lower limbs," in *Proc. 10th Med. Image Computat. Comput. Assist. Intervent. (MICCAI)*, 1998, pp. 1032–1040.

[24] M. Nesme, "Milieu mécanique déformable multi résolution pour la simulation interactive," Ph.D. dissertation, Univ. Joseph Fourier, Grenoble, France, 2008.

[25] M. R. Burcher, L. Han, and J. A. Noble, "Deformation correction in ultrasound images using contact force measurements," in *IEEE Workshop Math. Methods Bio. Imag. Anal.*, Kauai, HI, 2001, pp. 63–70.

[26] A. Gee, R. Prager, G. Treece, C. Cash, and L. Berman, "Processing and visualizing three-dimensional ultrasound data," *Br. J. Radiol.*, vol. 77, pp. S186–S193, 2004.

[27] O. Goksel and S. E. Salcudean, "Real-time synthesis of image slices in deformed tissue from nominal volume images," in *Proc. 10th Med. Image Computat. Comput. Assist. Intervent. (MICCAI)*, Brisbane, Australia, Oct. 2007, pp. 401–408.

[28] O. Goksel and S. E. Salcudean, "Fast B-mode ultrasound image simulation of deformed tissue," in *IEEE Eng. Med. Biol. Conf.*, Lyon, France, Aug. 2007, pp. 2159–2162.

[29] J. F. Krücker, G. L. LeCarpentier, J. B. Fowlkes, and P. L. Carson, "Rapid elastic image registration for 3D ultrasound," *IEEE Trans. Med. Imag.*, vol. 21, no. 11, pp. 1384–1394, Nov. 2002.

[30] D. Zikic, W. Wein, A. Khamene, D.-A. Clevert, and N. Navab, "Fast deformable registration of 3-D-ultrasound data using a variational approach," in *Proc. MICCAI*, Copenhagen, Denmark, 2006, pp. 915–923.

[31] J. E. Goodman and J. O'Rourke, *Handbook of Discrete and Computational Geometry*, 2nd ed. Boca Raton, FL: Chapman & Hall/CRC, 2004.

[32] D. Dobkin and R. J. Lipton, "Multidimensional searching problems," *SIAM J. Comput.*, vol. 5, no. 2, pp. 181–186, 1976.

[33] D. G. Kirkpatrick, "Optimal search in planar subdivisions," *SIAM J. Comput.*, vol. 12, pp. 28–35, 1983.

[34] C. B. Barber, D. P. Dobkin, and H. T. Huhdanpaa, "The Quickhull algorithm for convex hulls," *ACM Trans. Math. Software*, vol. 22, no. 4, pp. 469–483, 1996.

[35] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Syst. J.*, vol. 4, no. 1, pp. 25–30, 1965.

[36] O. C. Zienkiewicz and R. L. Taylor, *The Finite Element Method*. Oxford, U.K.: Butterworth Heinemann, 2000.

[37] G. Debunne, M. Desbrun, M.-P. Cani, and A. H. Barr, "Dynamical real-time deformations using space and time adaptive sampling," in *ACM Trans. Graphics (SIGGRAPH Proceedings)*, 2001, vol. 20, pp. 31–36.

[38] R. Löhner, "Progress in grid generation via the advancing front technique," *Eng. Comput.*, vol. 12, pp. 186–210, 1996.

[39] A. Fousse, E. Andres, J. Françon, Y. Bertrand, and D. Rodrigues, "Fast point locations with discrete geometry," in *SPIE Conf. Vis. Geometry*, Denver, CO, USA, Jul. 1999, pp. 33–44.

[40] M. Müller, J. Dorsey, L. McMillan, R. Jagnow, and B. Cutler, "Stable real-time deformations," in *Proc. SIGGRAPH/Eurographics Symp. Comput. Animat.*, 2002, pp. 49–54.

[41] T. R. Halfhill, Microprocessor report: Parallel processing with CUDA Reed Electronics Group: In-Stat, 2008, Tech. Rep.

[42] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA: MIT Press & McGraw-Hill, 2001.