# Realistic Analytical Polyhedral MRI Phantoms

Tri M. Ngo,[1] George S. K. Fung,[2] Shuo Han,[1] Min Chen,[3,4] Jerry L. Prince,[1,5] Benjamin M. W. Tsui,[2] Elliot R. McVeigh,[1] and Daniel A. Herzka[1]*

**Purpose:** Analytical phantoms have closed form Fourier transform expressions and are used to simulate MRI acquisitions. Existing three-dimensional (3D) analytical phantoms are unable to accurately model shapes of biomedical interest. The goal of this study was to demonstrate that polyhedral analytical phantoms have closed form Fourier transform expressions and can accurately represent 3D biomedical shapes.

**Methods:** The Fourier transform of a polyhedron was implemented and its accuracy in representing faceted and smooth surfaces was characterized. Realistic anthropomorphic polyhedral brain and torso phantoms were constructed and their use in simulated 3D and two-dimensional (2D) MRI acquisitions was described.

**Results:** Using polyhedra, the Fourier transform of faceted shapes can be computed to within machine precision. Smooth surfaces can be approximated with increasing accuracy by increasing the number of facets in the polyhedron; the additional accumulated numerical imprecision of the Fourier transform of polyhedra with many faces remained small. Simulations of 3D and 2D brain and 2D torso cine acquisitions produced realistic reconstructions free of high frequency edge aliasing compared with equivalent voxelized/rasterized phantoms.

**Conclusion:** Analytical polyhedral phantoms are easy to construct and can accurately simulate shapes of biomedical interest. **Magn Reson Med 76:663–678, 2016. © 2015 Wiley Periodicals, Inc.**

**Key words:** analytical phantom; magnetic resonance imaging; simulation; Fourier transform

## INTRODUCTION

In MRI acquisition and reconstruction development, it is useful to be able to quickly generate Fourier measurements of test objects. Physical test objects are referred to as physical MRI phantoms, or simply MRI phantoms. Realistic physical phantoms can be time consuming, cumbersome, and expensive to construct (1). In contrast, digital phantoms, sometimes referred to as computerized phantoms (2), are computer models. Digital phantoms are convenient because test data can be generated from simulated acquisitions, but they currently have a number of shortcomings that limit their use.

Digital phantoms can be divided into analytical or rasterized phantoms. Analytical phantoms are based on functions in the image domain that also have closed form Fourier transform (FT) expressions. These expressions allow the FT to be computed accurately at arbitrary spatial frequencies. An ideal analytical phantom for MRI development should accurately approximate the borders as well as the intensity profiles of shapes comprising biomedically relevant objects. However, existing three-dimensional (3D) analytical phantoms are limited in the types of boundaries and intensity variations they can model. For instance, Koay's implementation of the 3D Shepp–Logan head phantom (3,4) is constructed using ellipsoids with uniform intensity. Although Guerquin-Kern et al. (5) recently provided a closed form FT for 2D B-spline analytical phantoms with simulated coil sensitivity profiles and Zhu et al. (6) used polygonal models to create an analytical vocal tract phantom, these phantoms are confined to 2D simulations.

In contrast to analytical phantoms, rasterized or voxelized phantoms (2,7–9) are based on voxels. They can be generated by discretely sampling continuous functions in the image domain but can also include simulations that evaluate the Bloch equation on a per-voxel basis (10,11). Unlike the exact FT of analytical phantoms, rasterized phantoms can only provide an approximation of the FT of the original function by computing the discrete Fourier transform (DFT) of the image domain samples. The approximation accuracy is bounded by the rasterized phantom resolution. It can be made more accurate by oversampling (ie, increasing the image domain sampling density), which in turn reduces high frequency aliasing in *k*-space. However, because the computational cost of the DFT increases with matrix size, and keeping in mind that that all image space samples are required to compute a single *k*-space sample, motion simulations can be computationally intensive. Separate image domain matrices are required to capture different motion states. Simulating motion between *k*-space samples requires computing the DFT of the matrix corresponding to the state of the object at sample time. Because the object may be moving continuously, the object may only spend a few *k*-space samples between state changes. Thus, the DFT of many matrices must be computed to simulate even a few k-space samples, which can be time consuming. Additionally, non-Cartesian *k*-space trajectories are less straightforward to simulate because the DFT produces *k*-space points on a Cartesian grid.

[1]Department of Biomedical Engineering, The Johns Hopkins University School of Medicine, Baltimore, Maryland, USA.

[2]Division of Medical Imaging Physics, The Russell H. Morgan Department of Radiology and Radiological Science, Johns Hopkins Medical Institutions, Baltimore, Maryland, USA.

[3]Image Analysis and Communications Laboratory, Department of Electrical and Computer Engineering, The Johns Hopkins University, Baltimore, Maryland, USA.

[4]Translational Neuroradiology Unit, National Institute of Neurological Disorders and Stroke, Bethesda, Maryland, USA.

[5]Department of Electrical & Computer Engineering, The Johns Hopkins University, Baltimore, Maryland, USA

Grant sponsor: American Heart Association; Grant number: 11SDG5280025; Grant sponsor: National Institutes of Health; Grant number: NIH T32 training grant EB-010021.

*Correspondence to: Daniel A. Herzka, PhD, 720 Rutland Ave, 726 Ross Building, Baltimore, MD 21205. E-mail: daniel.herzka@jhu.edu

Received 23 February 2015; revised 30 June 2015; accepted 24 July 2015
DOI 10.1002/mrm.25888
Published online 19 October 2015 in Wiley Online Library (wileyonlinelibrary.com).

Here we introduce 3D analytical phantoms constructed using polyhedra with uniform intensities, which although limited in their ability to model intensity variations, are able to accurately model the boundaries of many biomedically relevant shapes. A polyhedron is any closed surface comprised of polygonal faces and is general enough to accurately approximate the boundaries of many 3D objects. Additionally, because closed triangular meshes are also polyhedra, an analytical polyhedral phantom can be easily constructed using existing 3D computer graphics modeling tools that manipulate triangular meshes. Other 3D surface representations such as nonuniform rational B-splines (NURBS) can be converted to triangular meshes, which can then be used in an analytical polyhedral phantom.

We describe the theory and construction of analytical polyhedral phantoms, expanding on previously published work (12). First, the closed form FT expressions for polyhedra are reviewed. Next, we describe an implementation of the FT of a polyhedron. We then evaluate the accuracy of the FT computation for objects that can be exactly described using a polyhedra (eg, a cube) and objects that can only be approximated (eg, an ellipsoid). Finally, we describe the construction of analytical polyhedral phantoms that accurately model a brain and a torso and demonstrate their application in MRI simulations.

## THEORY

We herein review the analytical expressions for the FT of a polyhedron (13). A detailed derivation can be found in the Appendix. Vector quantities are in bold, scalars in plain typeface, and when both bold and plain typeface versions of a symbol exist (eg, $\boldsymbol{k}$ and $k$), the plain typeface represents the magnitude of the corresponding vector.

### FT of a Polyhedron

A polyhedron is comprised of $F$ polygonal faces. The boundary of the face $f$ is composed of $E_f$ vertices, $\boldsymbol{r}^{(V_{f1})} \cdots \boldsymbol{r}^{(V_{fE_f})}$, ordered in a counterclockwise fashion when viewed with the face normal $\widehat{\boldsymbol{N}}_f$ pointing at the observer. $\boldsymbol{L}_{fe} = \boldsymbol{r}^{(V_{f(e+1)})} - \boldsymbol{r}^{(V_{fe})}$ is a vector oriented in the direction of the e$^{\text{th}}$ edge, pointing from vertex $\boldsymbol{r}^{(V_{fe})}$ to $\boldsymbol{r}^{(V_{f(e+1)})}$ with the same length as the edge. $\widehat{\boldsymbol{t}}_{fe} = \boldsymbol{L}_{fe}/L_{fe}$ is the unit vector oriented in the direction of the e$^{\text{th}}$ edge. $\widehat{\boldsymbol{n}}_{fe}$ is a unit vector normal to the e$^{\text{th}}$ edge and pointing outward from the interior of the polygon. In addition, the first and last vertices are connected, thus $\boldsymbol{r}^{(V_{f(E_f+1)})} = \boldsymbol{r}^{(V_{f1})}$.

$S_{3D}(\boldsymbol{k})$ is the analytical expression for the FT of a unitary intensity polyhedron $\rho(\boldsymbol{r})$ at spatial frequency $\boldsymbol{k}$ and is given in Equation [1]. If $\boldsymbol{k} = \boldsymbol{0}$ (ie, the DC offset), the FT is the volume polyhedron $V$. At non-zero $k$-space frequencies, $S_{3D}(\boldsymbol{k})$ is proportional to the sum of face contributions $S_f^*(\boldsymbol{k})$, where $f \in [1, F]$. When a face $f$ is normal to $\boldsymbol{k}$ (ie, its normal vector $\widehat{\boldsymbol{N}}_f$ is parallel to $\boldsymbol{k}$, that face contribution is proportional to the area of the face, $P_f$ modulated by a complex exponential whose argument is the dot product of $\boldsymbol{k}$ and the first vertex in the face $\boldsymbol{r}^{(V_{f1})}$. Otherwise, when $\boldsymbol{k} \neq k\widehat{\boldsymbol{N}}_f$, the face contribution is proportional to the sum of contributions from each edge comprising the face. The contribution of edge $e$ is proportional to the product of the length of the edge $L_{fe}$, a sinc and a complex exponential whose argument contains $\boldsymbol{r}^{(C_{fe})}$, the midpoint of the edge. The number of terms in the overall sum for $S_{3D}(\boldsymbol{k})$ and thus the computational cost is proportional to the total number of edges in the polyhedron. Because the FT is linear, the expressions for nonunitary intensity polyhedra can be obtained by multiplying $S_{3D}(\boldsymbol{k})$ by the desired intensity. Figure 1 and Table 1 illustrate and define these quantities.

$$
\begin{aligned}
S_{3D}(\boldsymbol{k}) &= \begin{cases} -\dfrac{1}{(2\pi k)^2}\displaystyle\sum_{f=1}^{F} S_f^*(\boldsymbol{k}), & \boldsymbol{k} \neq \boldsymbol{0} \\ V, \ \boldsymbol{k} = \boldsymbol{0} \end{cases} \\
S_f^*(\boldsymbol{k}) &= \begin{cases} \dfrac{\boldsymbol{k}\cdot\widehat{\boldsymbol{N}}_f}{k^2 - (\boldsymbol{k}\cdot\widehat{\boldsymbol{N}}_f)^2}\displaystyle\sum_{e=1}^{E_f} L_{fe}\boldsymbol{k}\cdot\widehat{\boldsymbol{n}}_{fe}\frac{\sin(\pi\boldsymbol{k}\cdot\widehat{\boldsymbol{t}}_{fe}L_{fe})}{\pi\boldsymbol{k}\cdot\widehat{\boldsymbol{t}}_{fe}L_{fe}} exp\left(-2\pi i\boldsymbol{k}\cdot\boldsymbol{r}^{(C_{fe})}\right), & \boldsymbol{k} \neq k\widehat{\boldsymbol{N}}_f \\ -2\pi i\boldsymbol{k}\cdot\widehat{\boldsymbol{N}}_f exp\left(-2\pi i\boldsymbol{k}\cdot\boldsymbol{r}^{(V_{f1})}\right)P_f, & \boldsymbol{k} = k\widehat{\boldsymbol{N}}_f \end{cases} \\
P_f &= \frac{1}{2}\left|\widehat{\boldsymbol{N}}_f \cdot \sum_{e=1}^{E_f}\{\boldsymbol{r}^{(V_{fe})} \times \boldsymbol{r}^{(V_{fe+1})}\}\right|.
\end{aligned}
\qquad [1]
$$

V is the volume of the polyhedron (14), which is given in terms of its vertices as

$$
\begin{aligned}
S_{3D}(\mathbf{k} = \mathbf{0}) &= V \\
&= \frac{1}{6}\left|\left[\sum_{f=1}^{F}(\boldsymbol{r}^{V_{f1}}\cdot\widehat{N}_f)\left|\widehat{N}_f\cdot\left\{\sum_{e=1}^{E_f}\boldsymbol{r}^{(V_{fe})}\times\boldsymbol{r}^{(V_{fe+1})}\right\}\right|\right]\right|
\end{aligned}
\qquad [2]
$$

## METHODS

### Implementation

We implemented the FT of a polyhedron in two forms. First, a multithreaded C++ MATLAB (MathWorks, Natick, Massachusetts, USA) plugin was created. Because the face contributions are independent, they were computed in parallel using the available CPU threads. This strategy works well for motion simulations where the geometry of a polyhedral phantom with many faces changes every few $k$-space samples. Alternatively, if many

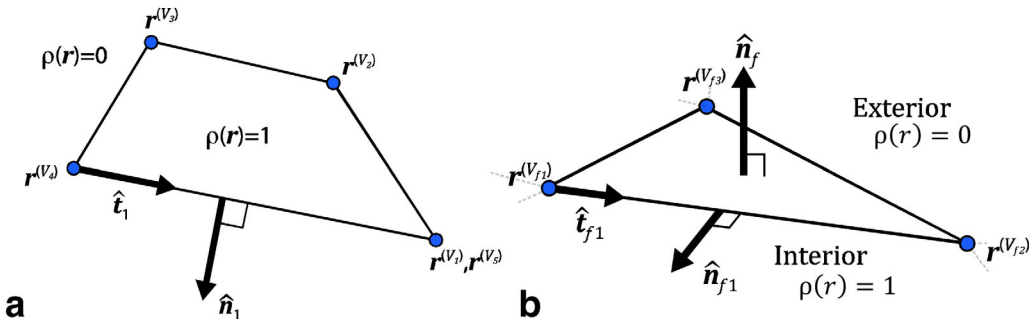


FIG. 1. Illustration of a polygon, a face of a polyhedron, and their associated parameters. (**a**) Unitary intensity 2D polygon comprised of four vertices $V_1 \cdots V_4$ oriented in a counterclockwise fashion. (**b**) Single triangular face of a polyhedron comprised of vertices $V_{f1} \cdots V_{f3}$. The remaining parameters are described in Table 1.

*k*-space samples are needed for a single state of the object (eg, in static simulations), an alternative efficient implementation can divide the computation among *k*-space samples instead of faces. Unless stated otherwise, this implementation was used for testing in this study. The second implementation is comprised of a set of MATLAB functions that take advantage of built-in parallelization and required no additional external libraries. This implementation, apart from being platform-independent, is parallelized along both the faces and k-point dimensions, using either vectorization or innate multithreading.

Multiple components comprising a polyhedral phantom can be modeled using separate polyhedra, similar to the multiple ellipsoids comprising the Shepp–Logan phantom. The FT of the composite phantom is obtained by computing the weighted sum of the individual FT of all component polyhedra, where the weights are the desired intensities of each component. Overlapping components will cause summation of intensities in the image domain.

Attention should be paid to the quality of polyhedra comprising a phantom. Polyhedra with undetected surface gaps, self-intersections, or incorrectly ordered vertices can produce unexpected results in the computed FT. The effect of a gap is proportional to its surface area. The gap can be interpreted as a missing face. Because the magnitude of the missing face contribution is proportional to the length of its edges, a larger gap has proportionately greater effect. Self-intersections cause reversal of internal and external regions, reversing the orientation of faces. The reversed orientation introduces a negative sign into the face contribution. As a result, a region of self-intersection can have the complex phase of its intensity reversed. The effect of a self-intersection is proportional to the volume of the regions involved. Finally, vertices that comprise a face are assumed to be ordered in a counterclockwise fashion with the vector normal to the face pointing toward the observer. Reverse ordering also reverses the vector normal to the face, again inverting inner and external regions similar to self-intersections. Again, artifacts associated with reverse ordering are proportional to the size of the faces involved. Because these abnormalities affect the FT expression in a continuous fashion, the threshold at which they become noticeable will depend on machine numerical precision.

## Validation

### *Accuracy*

We compared the FT of a unit cube–shaped polyhedron against the known gold standard FT of a unit cube. The

Table 1
Symbols

| Symbol | Definition |
|---|---|
| $S$ | Fourier Domain MR signal |
| $\boldsymbol{k}$ | *k*-space vector |
| | 2D polygon $\in R^2$ |
| $E$ | Total number of vertices or edges |
| $e$ | index of e$^{th}$ vertex or edge |
| $V_e$ | e$^{th}$ vertex, first and last vectors connected,i.e. $V_{E+1} = V_1$ and $V_E = V_0$. |
| $\boldsymbol{r}^{(V_e)}$ | position vector associated vertex $V_e$ |
| $\boldsymbol{L}_e$ | $\boldsymbol{L}_e = \boldsymbol{r}^{(V_{e+1})} - \boldsymbol{r}^{(V_e)}$, vector oriented in the direction the e$^{th}$ edge, pointed from vertex $V_e$ to $V_{e+1}$ with the same length as the edge |
| $\hat{\boldsymbol{t}}_e$ | $\hat{\boldsymbol{t}}_e = \boldsymbol{L}_e / L_e$, unit vector oriented in the direction of the e$^{th}$ edge |
| $\hat{\boldsymbol{n}}_e$ | is a unit vector normal to the e$^{th}$ edge and pointing outward from the polygon |
| $\boldsymbol{r}^{(C_e)}$ | $\boldsymbol{r}^{(C_e)} = \boldsymbol{r}^{(V_e)} + \hat{\boldsymbol{t}}_e L_e/2$, position vector of the midpoint between vertices $V_e$ and $V_{e+1}$ |
| $P$ | Area of Polygon |
| | 3D polyhedron $\in R^3$ |
| $F$ | total number of faces in the polyhedron |
| $f$ | index of f$^{th}$ face |
| $\hat{\boldsymbol{N}}_f$ | outward normal of the f$^{th}$ face |
| $E_f$ | total number of edges or vertices of the f$^{th}$ face |
| $V_{fe}$ | e$^{th}$ vertex of f$^{th}$ face, first and last vectors connected, i.e. $V_{f,E_f+1} = V_{f1}$ and $V_{fE_f} = V_{f0}$ |
| $\boldsymbol{r}^{(V_{fe})}$ | position vector associated vertex $V_{fe}$ |
| $\boldsymbol{L}_{fe}$ | $\boldsymbol{L}_{fe} = \boldsymbol{r}^{(V_{f,e+1})} - \boldsymbol{r}^{(V_{fe})}$, vector oriented in the direction the e$^{th}$ edge, pointed from vertex $V_{fe}$ to $V_{f,e+1}$ with the same length as the edge |
| $\hat{\boldsymbol{t}}_{fe}$ | $\hat{\boldsymbol{t}}_{fe} = \boldsymbol{L}_{fe} / L_{fe}$, unit vector in the direction of the e$^{th}$ edge of the f$^{th}$ face |
| $\hat{\boldsymbol{n}}_{fe}$ | outward normal to the e$^{th}$ edge of the f$^{th}$ face and is in the plane of the face |
| $\boldsymbol{r}^{(C_{fe})}$ | $\boldsymbol{r}^{(C_{fe})} = \boldsymbol{r}^{(V_{fe})} + \hat{\boldsymbol{t}}_{fe} L_{fe}/2$, position vector of the midpoint between vertices $V_{fe}$ and $V_{f,e+1}$ |
| $P_f$ | Area of f$^{th}$ face |
| $V$ | Volume of Polyhedron |

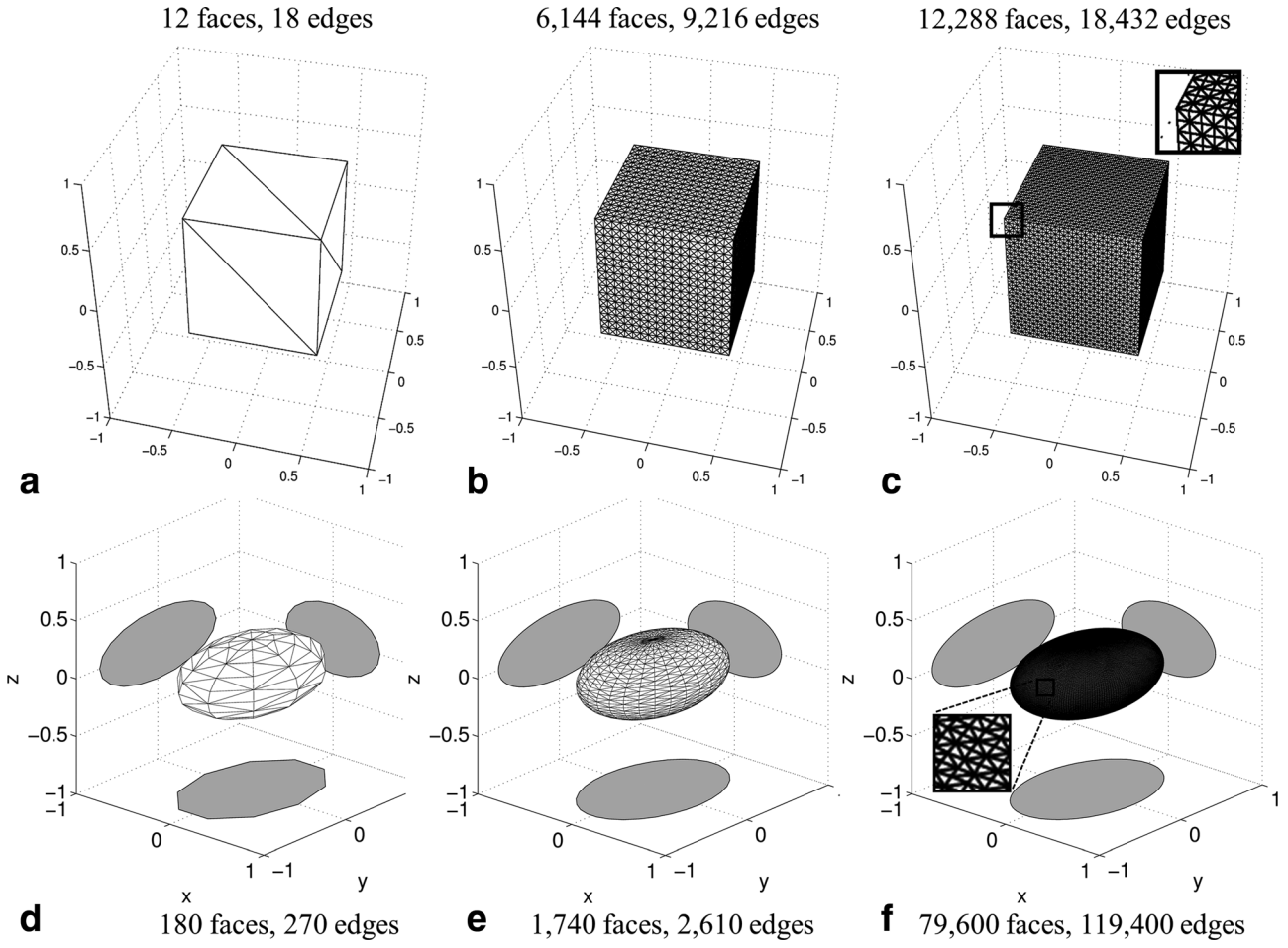


FIG. 2. Examples of triangular meshes with varying number of faces used to model a unit cube (**a–c**) and an ellipsoid (**d–f**). Unit cube–shaped triangular meshes containing varying number of faces were used to validate the implementation of the FT of a polyhedron against the gold standard of the FT of a 3D *rect* function. The magnified inset illustrates the small triangular faces comprising the mesh in panel c. Triangular meshes with varying number of faces (d–f) were used to approximate the FT of an ellipsoid against the closed form FT. The gray shadows show cross sections of the mesh in the x = 0, y = 0, and z = 0 planes. The magnified inset for the 79,600 face mesh illustrates dense triangles in the surface mesh.

unit cube, $\rho_{gs}$, has a closed form FT, $S_{gs}$, that is the product of sincs:

$$rect(x) = \begin{cases} 0, & if \ |x| > \frac{1}{2} \\ \frac{1}{2}, if|x| = \frac{1}{2} \\ 1, if|x| < \frac{1}{2} \end{cases}, \mathcal{F}(rect(x)) = sinc(k) = \frac{\sin(\pi k)}{\pi k}$$

$$\rho_{gs}(x,y,z) = rect(x) \cdot rect(y) \cdot rect(z)$$

$$S_{gs}(\boldsymbol{k}) = \mathcal{F}(\rho_{gs}) = W_x W_y W_z \cdot sinc(W_x k_x) \cdot sinc(W_y k_y) \cdot sinc(W_z k_z). \quad [3]$$

Here, $W_x, W_y, W_z$ define the length of the *rect* in the x, y, and z directions, respectively, and $k_x, k_y, k_z$ are elements of the $\boldsymbol{k}$ vector. The lengths were set to unity ($W_x = W_y = W_z = 1$), thus the function is a unit cube with unitary intensity centered at the origin. The unit cube can be exactly modeled using a triangular mesh, thus the FT can also be computed with our implementation of the FT of a polyhedron. The unit cube–shaped triangular mesh used in this validation had 12 triangular faces, two triangles defining each of the six square faces as illustrated in Figure 2. For the calculation of the FTs, we used a field of view (FOV) of 2 corresponding to a *k*-space sampling density of $\Delta k = 1/FOV = 0.5$ and a sampling matrix of 64 × 64 × 64. These same parameters, unless otherwise specified, were used for all validation experiments.

### *Floating Point Precision*

Edges are the basic unit in the computation of the FT of a polyhedron. Polyhedra with many faces, and thus many edges, are useful for modeling complex biomedically relevant geometry. We therefore assessed whether the FT of a polyhedron accumulated prohibitively high floating point precision errors as the number of edges increased. The gold standard of the FT of a unit width

3D *rect* was compared with the FT of a unit cube–shaped polyhedra containing 12 to 98,304 triangular faces or equivalently, 18 to 147,456 edges respectively (number of edges = 1.5× number of faces for triangular meshes). Figure 2a–2c illustrates example meshes containing 12, 6144, and 12,288 triangular faces or 18, 9216, and 18,432 edges, respectively. Additionally, the cube was shifted away from the origin by adding a random value between −0.5 and 0.5 to one or more of the x, y, and z components of the vertices, and the gold standard *k*-space was multiplied by a corresponding linear phase.

### *Approximating Smooth Surfaces*

To approximate smooth surfaces with a polyhedron, progressively smaller facets can be used until the desired approximation accuracy is obtained. We quantified the geometric approximation error incurred when modeling a smooth surface using a polyhedron. We examined how the error decreases when approximating an ideal ellipsoid with a polyhedron with increasing number of faces. We chose an ellipsoid because it is a smooth surface with a known closed form FT (3). The ellipsoidal triangular meshes were generated using MATLAB in two steps. First points were generated on the surface of an ellipse (using "ellipsoid"). Then, a triangular mesh was generated from these points (using "convhull"). The meshes contained between 180 and 79,600 faces or 270 and 119,400 edges, respectively. Figure 2d–2f illustrates the meshes containing 180, 1740, and 79,600 faces.

## Brain Phantom

### *3D MRI Simulation*

Triangular meshes representing the outer and inner cortical brain surfaces were generated using anatomical data. The source data was a magnetization-prepared rapid gradient echo MRI volume from the Open Access Series of Imaging Studies (OASIS) dataset (15). The Topology Preserving Tissue Classification of Magnetic Resonance Brain Images (TOADS) algorithm (16) was used to segment the gray matter. The Cortical Reconstruction Using Implicit Surface Evolution (CRUISE) algorithm (17) was then applied to this segmentation to generate triangular meshes corresponding to the outer and inner cortical surfaces. The outer and inner cortical meshes contained 536,684 and 354,908 triangular faces, respectively, and are illustrated in Figure 3a and 3b. The number of faces was determined automatically by the TOADS and CRUISE algorithms and was dependent on factors such as the volume of the brain and the complexity of the cortical surface. Although subcortical structures such as ventricles were not included in this phantom, there are no technical limitations that prohibit their inclusion.

We simulated a 3D MRI acquisition using these meshes. The *k*-space matrix size was 128 × 128 × 128, and the FOV in the right, anterior, and superior directions was 137.35, 170.15, and 117.20 mm, respectively. The outer and inner cortical meshes were assigned intensities of 74 and 38, respectively, producing (after summation due to overlap) intensities of 74 in the cortex and 112 in the subcortical regions. Finally, the volume was reconstructed using a 3D IDFT.

### *2D MRI Simulation*

We simulated a 2D MRI acquisition wherein a slice of the 3D object is selectively excited before *k*-space is acquired. To simulate ideal slice selection where an infinitesimally thin slice of the brain is excited by a slice selective radio-frequency pulse, a 2D plane perpendicular to the slice selection axis, the z-axis, was intersected with the meshes generating sets of 2D slice contours. The location of the slice plane lies midway between the top and bottom faces of the slab mesh in Figure 3a and 3b. We used the Computational Geometry Algorithms Library (CGAL) (18) to compute these intersections. CGAL is a C/C++ library containing peer-validated, highly accurate geometry related functions. Figure 3c illustrates the resulting contours obtained through simulated ideal slice selection.

We also simulated slices with finite thickness, as is typical in 2D MRI acquisitions. Although real MRI slices usually have a Gaussian or sinc profile along the slice-encoding direction, this can be approximated with a *rect* function that can be modeled as a slab mesh. Alternatively, one could compute the 3D FT of the slab and convolve it with the FT of the desired arbitrary slice profile. However, if a closed form solution does not exist for the convolution, it must be approximated by sampling the FTs, which would introduce approximation errors. Using CGAL's Boolean surface operations, we computed the intersection between a 1-mm-thick slab mesh (Fig. 3a, 3b) and the cortical meshes to generate a set of 1-mm-thick sliced meshes for the inner and outer surfaces (Fig. 3d). The slab mesh was perpendicular to the slice encoding z-axis and large enough in the x and y directions to encompass the brain. Because the intersection can disconnect sections of the original contiguous cortical mesh, creating "islands," the result of the intersection between slab and cortical mesh is not a single closed sliced mesh but a set of closed meshes. Triangular faces in the original cortical mesh that are intersected by the top and bottom planes are converted to polygons with additional edges. The top and bottom surface of the sliced meshes are non-triangular polygons that outline the intersection of the top and bottom slab plane with the cortical mesh. The inner and outer cortical sliced mesh sets contained a total of 4798 and 3812 faces, respectively.

Using these contours and sliced meshes, we simulated a 2D MRI acquisition. We used the FT of polygons and polyhedra to compute the *k*-space of the infinitesimally thin slice 2D contours and finite thickness slice meshes, respectively. The same acquisition parameters were used for both simulations. As in a real 2D MRI acquisition, we sampled *k*-space on a Cartesian grid of matrix sizes 64 × 64, 128 × 128, and 256 × 256 in the $k_x$–$k_y$ plane (ie, only the $k_z = 0$ mm$^{-1}$ plane was sampled). Because the largest extent of the contours and sliced meshes was 122.78 mm in the anterior–posterior dimension, the imaging FOV was set to 1.1 times larger at 135.06 mm to prevent image domain aliasing. Sampling only the $k_z = 0$ mm$^{-1}$ plane integrates the finite slice in the z direction,

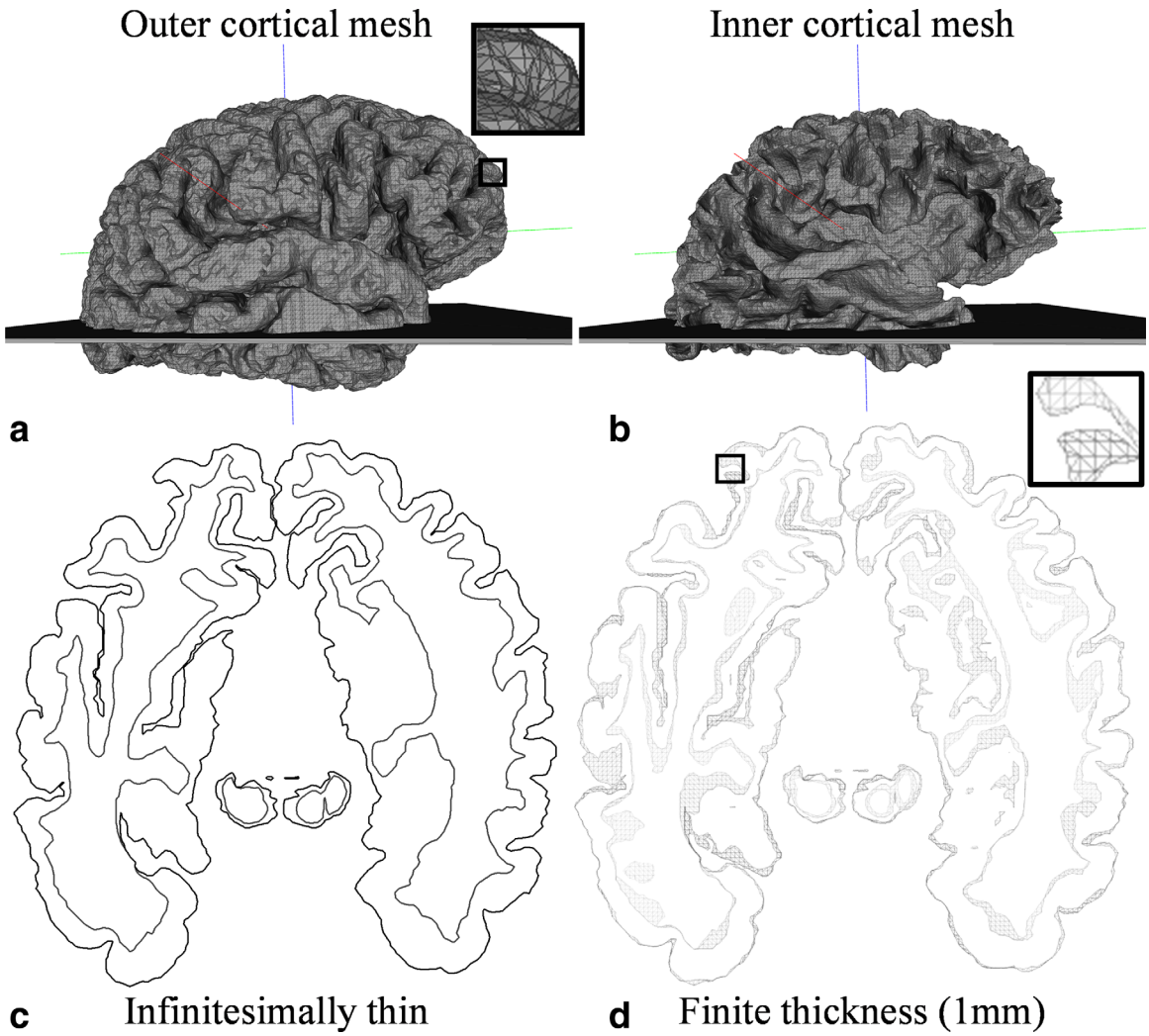


FIG. 3. Triangular meshes comprising a 3D brain analytical polyhedral phantom used in 3D and 2D MRI simulations. The brain is oriented such that the x, y, z axes correspond to the right, anterior, and superior anatomical axes, respectively. (**a**) Outer cortical mesh with magnified inset illustrating triangles in the mesh. (**b**) Inner cortical mesh with overlaid slab mesh used in 2D MRI simulation with finite slice thickness. Although not shown, the slice plane used to simulate idealized infinitesimally thin slice selection is centered in the slab mesh. (**c**) Resulting 2D contours from the intersection of all meshes with the ideal slice plane. (**d**) Resulting 3D sliced meshes from the intersection of all meshes with a 1-mm-thick slab mesh. The magnified inset illustrates triangles in the mesh.

which results in slice encoding related partial volume effects. The same tissue intensity assignments were used as in the 3D MRI simulation.

For comparison, we also generated a rasterized phantom by sampling the 2D contours in image space. The imaging parameters were equivalent to the analytical simulations. The image domain of each contour was sampled by assigning a value of one or zero for points inside or outside the contour, respectively. Each rasterized contour was then scaled by the same tissue intensities used in the analytical simulations before summation in image space. We did not perform a finite thickness slice selection rasterized simulation, because approximating through slice integration via image space sampling would introduce through-slice sampling density as another experimental parameter.

We used the IDFT to reconstruct images from the analytical phantom simulations. To compare the ideal slice and finite slice simulation images, the finite slice reconstruction was normalized by the slice thickness, since the intensity is integrated through the slice. No image reconstruction was required for the rasterized phantom.

### Torso Phantom

We simulated the 2D MRI acquisition of a torso with cardio-respiratory motion to demonstrate the application of a polyhedral analytical phantom in cardiac MRI simulations. First, triangular meshes for the polyhedral torso phantom were generated with source geometry and motion derived from the eXtended Cardiac-Torso (XCAT) phantom (2,19,20). XCAT is a four-dimensional (4D) NURBS and subdivision surface–based anthropomorphic phantom. These NURBS and surfaces model anatomy from the high-resolution anatomical images of the Visible Male and Female National Library of Medicine datasets. XCAT also models cardiac motion based on 100 time frames over a cardiac cycle of high resolution (0.32 mm

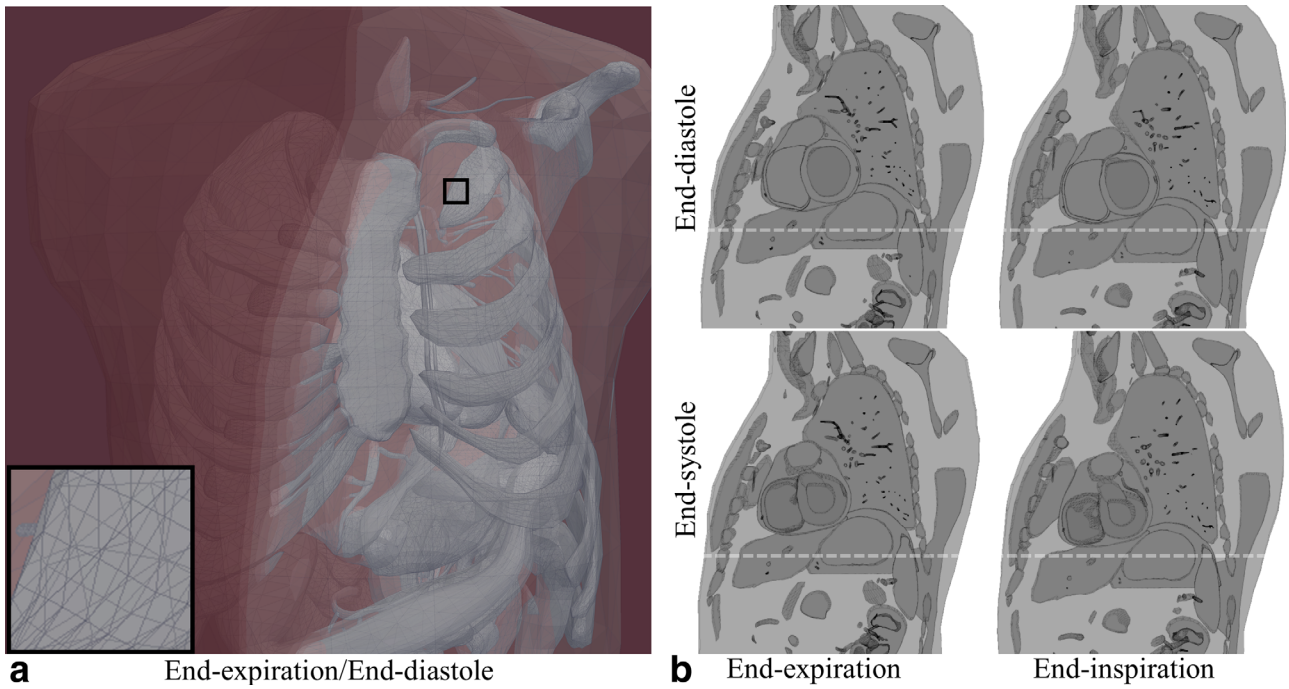


FIG. 4. Rendering of triangular mesh torso phantom with intersecting slice plane and resulting sliced meshes for different time points in the cardio-respiratory cycle. (**a**) 3D rendering of torso phantom mesh at end-expiration/end-diastole. Magnified inset illustrates triangular facets. The slice plane is highlighted in red. (**b**) Resulting sliced meshes after mesh subtraction and intersection operations. Although meshes were generated for 90 time points throughout the cardio-respiratory cycle, only the extremes of the cardio-respiratory cycle are illustrated here. Supporting Video S1 illustrates all 90 times points. The dashed lines indicate the relative motion of structures.

pixel size, 0.4 mm slice thickness) cardiac-gated multidetector CT data. Respiratory motion is also modeled based on respiratory-gated CT data. We used the male XCAT model for this simulation, extracting 90 sets of NURBS surfaces corresponding to uniformly spaced time points fully spanning a cardio-respiratory cycle.

The NURBS surfaces from the XCAT phantom were converted to triangular meshes using Rhinoceros (Robert McNeel & Associates, Seattle, Washington, USA), a computer graphics program specialized in manipulating NURBS. Each object in the XCAT phantom was represented by at least one mesh. Hollow objects such as the ventricles of the heart had separate meshes for inner and outer surfaces. The number of triangular faces per mesh varied. For instance, the left ventricular inner mesh contained 1322 triangles, whereas the left lung mesh contained 4086 faces. The total number of faces across all meshes for a single time point varied but was approximately 23,580. Figure 4a shows the resulting triangular mesh torso phantom at end-expiration/end-diastole. Conversion with Rhinoceros produced some self-intersecting meshes. Because Rhinoceros is proprietary software, it is unclear why conversion from NURBS to triangular meshes would in some cases result in self-intersections, holes, and other abnormalities. The CGAL library contains algorithms that detect self-intersections, and in using it we created a tool to determine which meshes were abnormal. We first attempted to repair these meshes using Polymender, a program that repairs defects in triangular meshes (21). After the attempted repair, we reran the self-intersection detection tool. If the repaired mesh was still self-intersecting, we used Meshlab (http://meshlab.sourceforge.net), a program that specializes in manipulating triangular meshes, for a second attempt at repairing the mesh. Unfortunately, some meshes could not be repaired by either program and were excluded from all time points of the phantom.

Some surfaces from XCAT overlapped during respiratory and cardiac motion, which caused undesired addition of intensities in the image domain. These overlapping regions were subtracted using mesh Boolean operations. Consequently, meshes that contained other meshes were "hollowed out." For instance, the left ventricular epicardial surface mesh derived from XCAT contained the endocardial surface mesh. The epicardial surface mesh, which describes a solid, was converted to a solid shell that had an outer and inner surface that followed the endocardial surface precisely and thus was able to accommodate the endocardial surface mesh without overlap.

Next, finite thickness slice selection was simulated for all time points. A 4-mm-thick slab mesh representing the slice plane positioned at the base of the heart in the short axis orientation was intersected with all meshes for each time point (Fig. 4a). We used CGAL functions to compute this intersection resulting in 4-mm-thick sliced meshes. This process was repeated for all time points in the cardio-respiratory cycle. Figure 4b illustrates the resulting sliced meshes at the extremes of the cardio-respiratory cycles.

A 2D MRI acquisition was simulated individually for all time points. The *k*-space of the sliced meshes were sampled using a $128 \times 128$ Cartesian matrix and FOV of 400 mm. The intensities of the polyhedra were chosen to simulate the relative intensity differences between

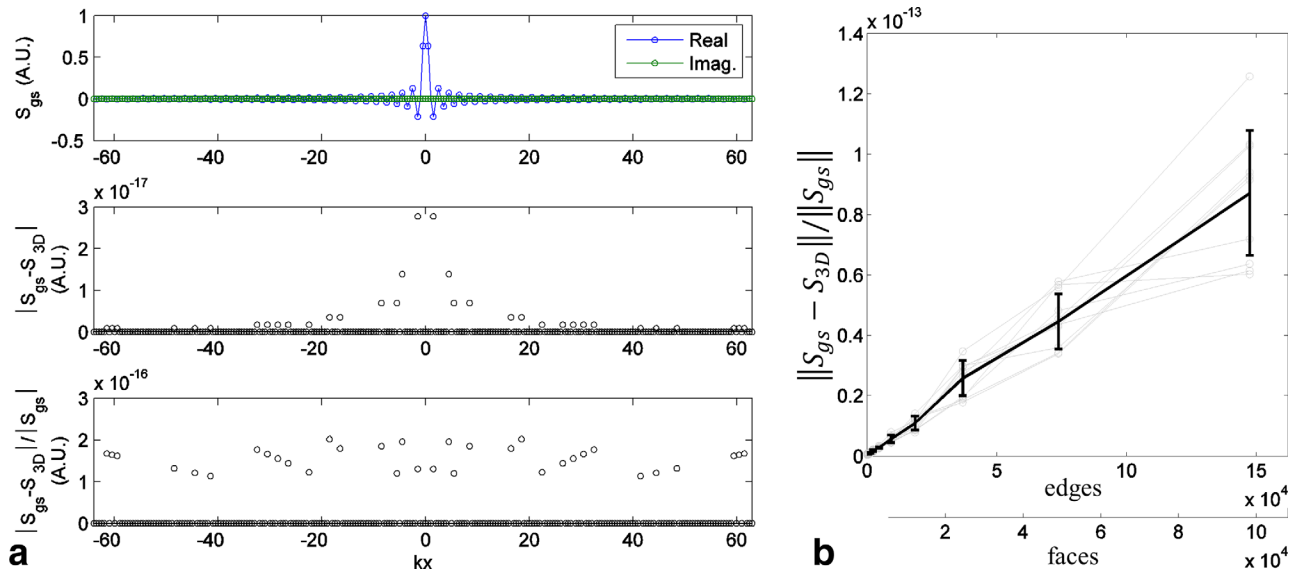


FIG. 5. Evaluation of FT error with increasing edges in unit cube–shaped triangular meshes. (**a**) Sample-wise evaluation of the FT error. Top row: Plot of the gold standard computed using the product of sincs. Values are given in arbitrary units (A.U.) The real and imaginary components are displayed separately. Middle row: Graph of magnitude of error versus $\boldsymbol{k}_x$-axis location. Bottom row: Graph of normalized magnitude of error versus $\boldsymbol{k}_x$-axis location, excluding points where the gold standard coefficient magnitude is zero since this would result in an undefined value. (**b**) Graph of the normalized $\ell^2$-norm of the error versus the number of faces in the unit cube mesh. Gray lines indicate different shifts of the cube from the origin. The black line with error bars indicate the mean and standard deviation of the normalized error taken across all displacements.

tissues in a bright-blood gradient echo MRI sequence. Finally, the IDFT was used to reconstruct images for each time point.

## RESULTS

### Validation

#### *Accuracy*

The FT of a unit cube triangular mesh with 12 faces (18 edges) computed using the proposed implementation closely matched the gold standard. To estimate differences between the proposed method and the gold standard, the magnitude of the error $|S_{gs}(\boldsymbol{k}) - S_{3D}(\boldsymbol{k})|$ and the normalized magnitude of the error $|S_{gs}(\boldsymbol{k}) - S_{3D}(\boldsymbol{k})|/|S_{gs}(\boldsymbol{k})|$, $|S_{gs}(\boldsymbol{k})| \neq 0$ were evaluated along the $k_x$-axis. As the middle row of Figure 5a illustrates, the magnitude of the error was small ($<3 \times 10^{-17}$ for all sample locations). Samples with larger coefficient magnitudes seemed correlated with higher magnitudes of error. The normalized magnitude of error was also very small ($<3 \times 10^{-18}$) as illustrated in the bottom row of Figure 5a. The normalized error was uncorrelated with the magnitude of the coefficient.

#### *Floating Point Precision*

The error due to limited floating point precision was small, suggesting that detailed meshes with many edges can be used to construct analytical polyhedral phantoms. We computed the $\ell^2$-norm of the difference between the FT of the polyhedron and gold standard normalized by the $\ell^2$-norm of the gold standard. Figure 5b shows that the normalized error increases approximately linearly with the number of edges with small variations between different displacements of the cube from the origin. Gray lines in the figure correspond to different displacements of the cube from the origin. The solid line with error bars indicates the mean and standard deviation of the error taken across all displacements. For the mesh with 147,456 edges, the mean normalized error is just $0.8717 \times 10^{-13}$. These computations were performed on a quad core Intel Xeon 2.13 Ghz system. A five-run average of the FT of the 12-face cube mesh (18 edges) over the $64 \times 64 \times 64$ $k$-space matrix using one, two, three, and four processor threads required 7.78, 4.02, 2.84, and 2.32 s or 1.65, 0.85, 0.60, and 0.49 μs per $k$-space sample per edge, respectively. Because run times were not recorded for all simulations, the computational rate corresponding to four threads will be used to estimate run times for other simulations. This is a reasonable estimate, because the computational cost of the FT of a polyhedron is linear in the number edges and $k$-space samples.

#### *Approximating Smooth Surfaces*

We computed the $\ell^2$-norm of the difference between the FT of the ellipsoidal polyhedron and ideal ellipsoid normalized by the $\ell^2$-norm of the FT of the ideal ellipsoid. The normalized error decreased quickly with increasing number of faces. The log of the $\ell^2$-norm of the error over the entire volume normalized by the $\ell^2$-norm of the gold standard volume decreased rapidly with increasing face number as illustrated in Figure 6a. At 79,600 faces (119,400 edges) the normalized error is only $6.21 \times 10^{-4}$. The geometric approximation error is magnitudes larger than the floating point precision error, suggesting the number of faces can be increased significantly although with an increase in computation time. The top row of

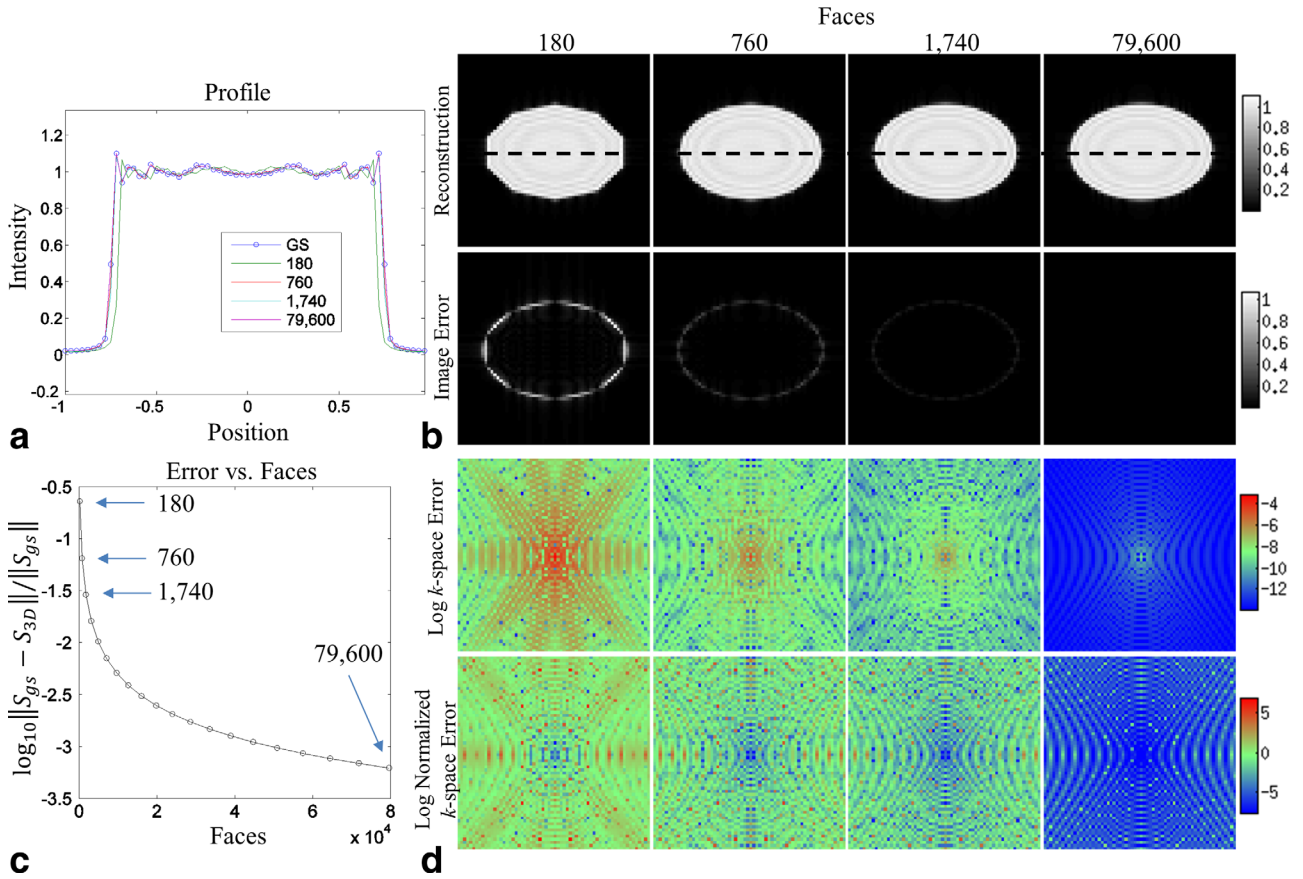


FIG. 6. The behavior of the error incurred when approximating the smooth surface of an ideal ellipsoid using polyhedra with increasing number of faces. The gold standard is an ideal ellipsoid with a known closed form FT. (**a**) Log of the $\ell^2$-norm of the error computed over the entire volume, normalized by the $\ell^2$-norm of the gold standard volume versus the number of faces in the triangular mesh. Arrows indicate error corresponding to meshes in Figure 2. (**b**, **c**) Image space (b) and *k*-space (c) details for the meshes. The top row of panel b shows the z = 0 slice of reconstructed 3D volume and image space error. The intensity profiles corresponding to the dotted lines on the slices are shown in panel c. The bottom row of panel b shows the magnitude of the image space error for the $z = 0$ slice. (**d**) Log of the absolute value of the *k*-space error corresponding to the $\boldsymbol{k}_z = 0$ plane (top row) and normalized by the absolute value of the gold standard (bottom row).

Figure 6b shows the $z = 0$ slice of the reconstructed FT of meshes containing 180, 760, 1740, and 79,600 faces; the corresponding error is indicated by arrows in Figure 6a. Gibbs ringing due to finite *k*-space sampling can be observed in the reconstructed images in Figure 6b top row. Figure 6c illustrates this artifact in more detail using an intensity profile corresponding to the dashed line in Figure 6b. In image space, the error—as expected—is concentrated at the edges and is shown in the bottom row of Figure 6b. In *k*-space, the largest absolute errors are concentrated at the lower frequencies (Fig. 6d, top row). However, when the absolute value of the error is normalized by the absolute value of the gold standard, higher frequencies tend to have more normalized error (Fig. 6d, bottom row).

## Brain Phantom

### *3D MRI Simulation*

The analytical polyhedral brain phantom was able to accurately model the gyri of the brain. Although the surfaces were comprised of triangular facets, they were small enough to approximate the smooth curves of the brain at this resolution. Gibbs ringing from finite sampling of *k*-space can also be observed in the inhomogeneity of intensities near the edges, an artifact found in real MRI images. Figure 7 illustrates representative slices from the reconstructed volume. The estimated computational time for the FT of the 3D brain phantom was approximately 15.96 d. The exponential increase in the number of samples in the 3D 128 × 128 × 128 sampling matrix compared with a 2D matrix and the large number of faces increased computation time substantially.

### *2D MRI Simulation*

The results of the 2D MRI simulations using the 2D analytical infinitesimally thin and 3D finite slice thickness analytical phantoms were compared with a 2D rasterized phantom generated from the contours of the 2D infinitesimally thin slice phantom (Fig. 8). Differences between the 2D analytical and the 2D rasterized phantoms were solely due to aliasing effects of image space sampling in the rasterized phantom. However, differences between the 3D finite slice analytical phantom and 2D rasterized phantom additionally include through-plane intensity

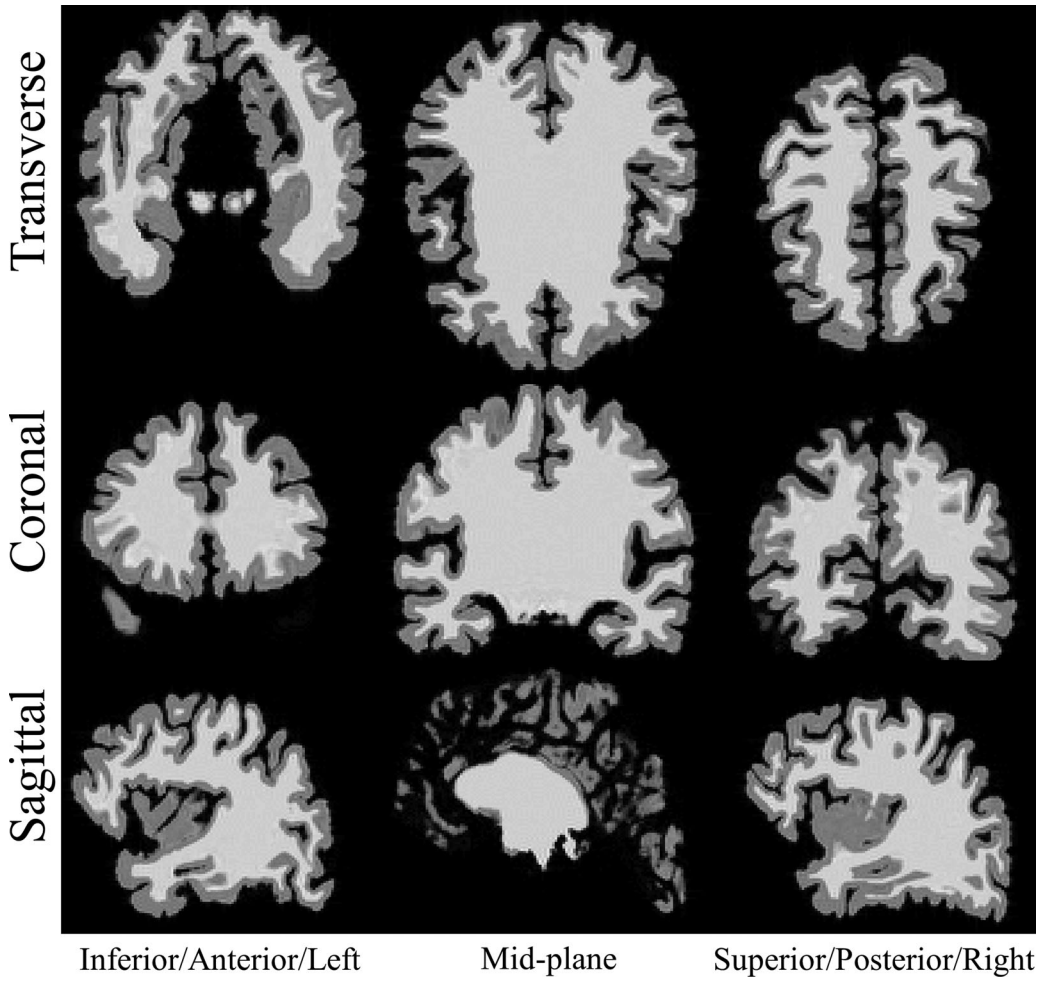


FIG. 7. Representative slices of reconstructed volume from 3D MRI simulation using analytical polyhedral brain mesh phantom. Images for the transverse, coronal, and sagittal planes are shown. From left to right: the inferior, mid-plane, and superior slices are shown for the transverse plane; the anterior, mid-plane, and posterior slices are shown for the coronal plane; and the left, mid-plane, and right slices are shown for the sagittal plane.

variations that accurately reflect underlying mesh geometry. To demonstrate the effects of finite slice thickness, rasterized phantom must densely sample the through plane direction to approximate these variations (data not shown). Using the C++-based MATLAB plugin, the time required to compute Fourier samples for the finite thickness slice selection 3D brain MRI simulation for the $64 \times 64$, $128 \times 128$, and $256 \times 256$ matrix sizes was 0.43, 1.73, and 6.93 min, respectively. We have omitted the slice plane and slab intersection operations because they required a negligible amount of time compared with the FT computation.

The finite thickness slice selection 2D brain MRI simulation exhibited partial volume effects seen in real 2D MRI acquisitions. The arrowheads in Figure 8 highlight partial volume effects at the edges seen only in the finite thickness simulation. Oblique edges of the mesh caused changes in intensity along the slice-encoding direction, which was integrated simulating partial volume effects in the finite thickness slice selection simulation. Aside from the edges, regions of low intensity can be observed in the interior when using finite slices, illustrated by the full arrows in Figure 8. In the finite thickness simulation, normalizing the integrated intensity by the slice thickness produces the mean intensity through the slice plane. Thus, regions that were predominantly low intensity in the slice direction may have lower intensity than the infinitesimally thin slice simulation, which does not have intensity variation in the slice-encoding direction.

Differences between the simulations using the rasterized phantom and the analytical phantoms were concentrated at the edges (Fig. 8c). The discrete image space samples of the rasterized phantom caused aliasing of high spatial frequencies of *k*-space. In the analytical phantom, regions of sudden intensity changes, such as edges, were accompanied by Gibb's ringing and partial volume effects, both of which are not simulated by the rasterized phantom. The mean difference between the rasterized and analytical phantom decreased with matrix size. The spacing between image space samples of the rasterized phantom decreases for larger matrices, reducing high frequency aliasing, thus the DFT of the rasterized phantom approximated the FT of the analytical phantoms more closely. Edge differences were more

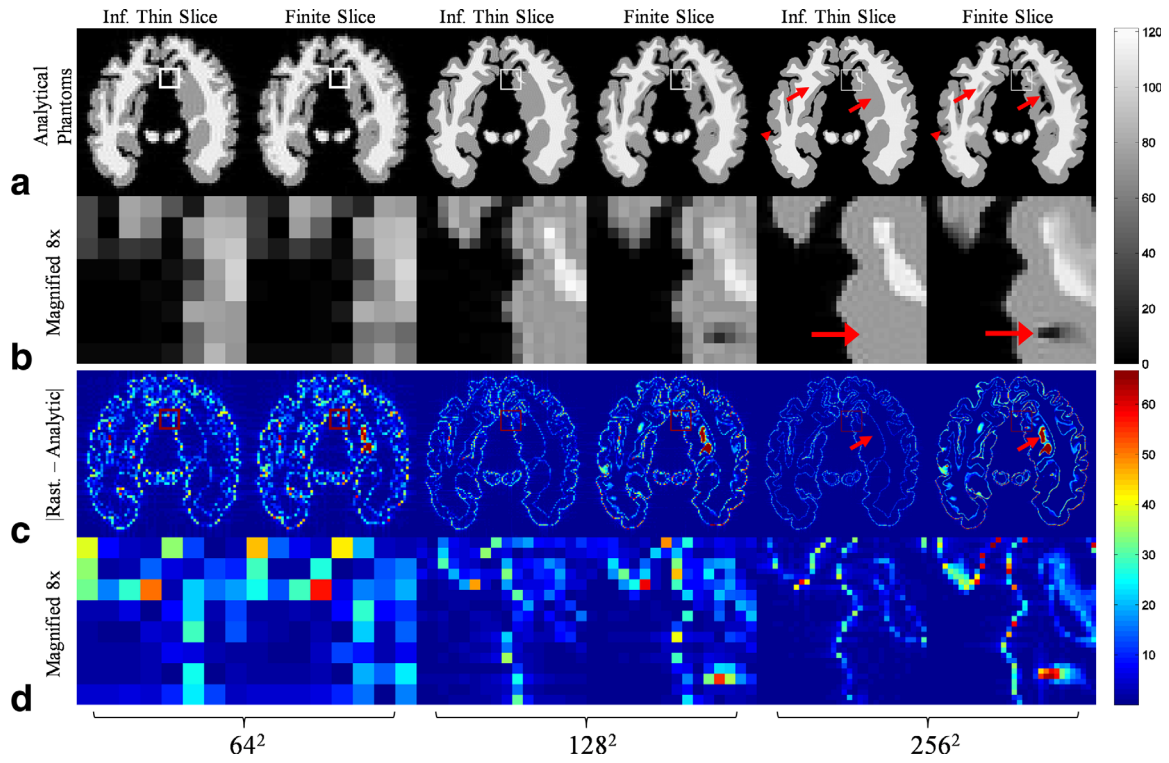


FIG. 8. Comparison of a simulated 2D MRI brain acquisition using an analytical polyhedral phantom with infinitesimally thin and finite thickness slice selection for matrix sizes 64 × 64, 128 × 128, and 256 × 256. All images have the same intensity scaling given by the color bars on the right. For each set of two columns, the thin and finite thickness slice selections are shown. The red arrowheads and red arrows highlight partial volume effects at the edges and in the interior, respectively. (**a**) IDFT reconstruction of ideal and finite slice simulations. (**b**) Magnified section corresponding to the region outlined by boundary pixels in row a. The boundary pixels become proportionately smaller as a side effect of increasing matrix size. (**c**) Magnitude of the difference between a rasterized simulation (not shown) generated from the infinitesimally thin slice selection contours and the analytical simulations. The rasterized simulation image matrix is the same size as the *k*-space matrix used for analytical phantoms. (**d**) Magnified section corresponding to the region outlined by boundary pixels in row c, which is the same region outlined in row a and magnified in row b.

pronounced in the finite slice simulation due to partial volume effects of oblique edges that were not modeled in the rasterized phantom.

The partial volume effect of finite thickness slice selection can be easily observed in Figure 8c. The full arrows illustrate a partial volume effect that was seen in the finite thickness simulation but was absent in the thin slice simulation. Because the rasterized phantom was based on the contours of the thin slice simulation, it did not exhibit the partial volume effects seen in the finite thickness slice simulation, and this can be observed in these difference images.

### Torso Phantom

The torso phantom demonstrates an analytical polyhedral phantom that incorporates motion. Using the C++-based MATLAB plugin, the FT for each time point required an estimated 4.75 min to compute and thus 7.12 h for all 90 time points. Again, we have omitted the time required to convert, slice, and fix the meshes because this was insignificant compared with the FT computation time. Figure 9 shows the reconstructed simulated acquisition of the phantom at the extremes of a cardio-respiratory cycle. Supporting Movie S1 shows the reconstructed simulated acquisition for 90 frames spanning a cardio-respiratory cycle. In-plane motion can be observed during the contraction of the heart between end-diastole and end-systole. The blood pool of the pulmonary artery seems to merge with the right ventricle due to their close vicinity in end-diastole, but they separate in end-systole. Additionally, motion of objects through the spatially fixed slice plane can be observed. The arrowheads in Figure 9 illustrate a portion of a coronary artery that came into the slice plane at end-inspiration/end-systole that was out of plane at end-expiration/end-systole. The vessels and bronchioles of the lung also illustrate through-plane motion. The full arrows in Figure 9 illustrate a vessel in the lung that came into the slice plane at end-expiration/end-systole but was out of plane at end-inspiration/end-systole.

The torso simulation also demonstrates that an analytical polyhedral phantom can be constructed using many meshes with greatly differing sizes. There were 180 meshes in each set of sliced torso meshes that ranged from the large body surface mesh to small coronary artery meshes. A rasterized phantom would cause aliasing of the large high frequency components of small meshes, which does not occur using the closed form FT of the analytical polyhedral phantom.

## DISCUSSION

Analytical polyhedral phantoms are useful in a number of areas of MRI development as evidenced by the

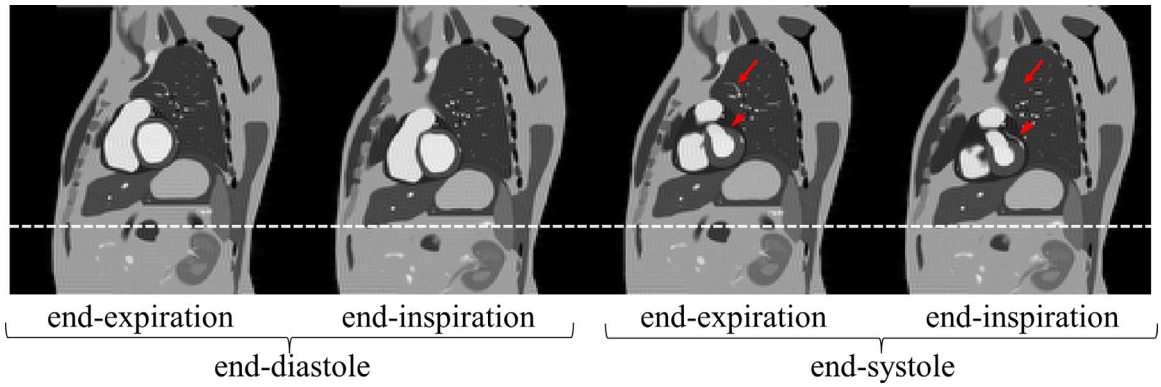


FIG. 9. IDFT reconstruction of a simulated 2D MRI acquisition using an analytical polyhedral phantom of the torso at the extremes of the cardio-respiratory cycle. The dashed line indicates relative motion between different time points. The red arrowheads highlight through-plane movement of a coronary artery; the red arrows highlight through-plane movement of a vessel in the lungs.

repeated use of the Shepp–Logan phantom and its derivatives based on ellipses or ellipsoids. Realistic 4D (3D space and one-dimensional time) simulations may be more accurately computed in comparison to using an oversampled (ie, super-sampled rasterized) phantom. In some cases (eg, simulation of motion), the polyhedral phantoms may be more computationally efficient than super-sampled rasterized phantoms. For instance, with the torso phantom, we only simulated motion between each complete 2D *k*-space matrix. However, we can also simulate motion between individual *k*-space samples. If motion information is available throughout the cardio-respiratory cycle, as it is in the XCAT phantom, every *k*-space sample can be computed using a set of meshes corresponding to the current sampling time, thereby modeling motion on an individual *k*-space sample resolution. To perform the equivalent motion simulation accurately using a rasterized phantom would require densely oversampling the entire 3D image space matrix for every *k*-space sample, which can be computationally intensive for 4D simulations. Consider such a motion simulation with a matrix size of $256 \times 256 \times 256$ using the outer cortical mesh in the 3D brain phantom and assuming two times oversampling factor for the rasterized phantom. The time required to compute a single k-space for a single motion state corresponds to the time required to compute the FFT of a $512 \times 512 \times 512$ matrix. This required our system approximately 4.44 s and one gigabyte of memory. In contrast, the time required to compute a single k-space point using the analytical phantom was 0.40 s and required 14.58 megabytes of memory. The number of operations required to compute $N$ *k*-space points where every point is a different motion state using the FFT, ignoring oversampling, is $O(N^2 \log N)$, whereas computing the same points using a polyhedral phantom is $O(N)$, linear with the number of *k*-space points. Note that though the polyhedral FT is significantly more computationally intensive than the standard FFT, it does not have the advantage of significant optimization, as does the FFT via the *fftw* libraries (27). Further optimization in the implementation of the phantom, including efficient parallelization, should significantly reduce computation time.

Analytical polyhedral phantoms are also useful for developing motion compensation techniques. Periodically rotated overlapping parallel lines with enhanced reconstruction (PROPELLER) (22) and projection navigators (23–25) use non-Cartesian *k*-space trajectories and detect features shared between images or projections captured at different motion states. Using a polyhedral phantom, these features can be realistically modeled, and non-Cartesian *k*-space trajectories can be simulated readily. In this study, we chose not to include examples of non-Cartesian trajectories such as 3D radial because that would introduce additional experimental factors, such as the choice of non-Cartesian reconstruction algorithm. However, because the 2D torso phantom simulation samples 3D *k*-space along an oblique plane, this trajectory demonstrates that the polyhedral phantom supports sampling of arbitrary *k*-space frequencies. Polyhedral phantoms may also have uses in CT simulations through the relationship between the FT and the Radon transform via the Fourier slice theorem. Artifacts from using facets to approximate smooth surfaces can be alleviated by increasing the number of facets.

Any in vivo configuration that can be described in image space by altering the triangular mesh can be simulated using a polyhedral phantom. Hence, nonlinear deformation of tissues is reduced to relocation of vertices before sampling of Fourier space. Other nonlinear behavior such as eddy currents can also be simulated with this framework. Eddy currents, which introduce offsets in intended *k*-space sample locations, can be modeled by creating a mismatch between the *k*-space coordinates used to sample the phantom and the coordinates used to reconstruct the data. However, to simulate phase accrual due to flow (as used in phase contrast imaging) is more difficult, because this could require subdivision of the physiologically relevant polyhedral into significantly smaller polyhedrons (as done in finite element methods). The position of each small element within a gradient at a given time point can then be used to weight the contribution of that element to the final Fourier transform, with the weight including both magnitude and phase, and the phase term being determined by the first moment of the experienced gradient. Though computationally intense, this approach could allow MR simulation of any finite element model–derived structure. Although we did not perform experiments, nonuniform meshes which increase the density of vertices in areas of high curvature, could more accurately capture the details of biomedical objects while minimizing computational cost in comparison to uniform meshes.

### Limitations

As demonstrated in the 3D brain simulation, computation time can be an issue for large matrices and meshes with many edges. Using our basic C++-based plug-in, calculation of the polyhedral FT for a large number of faces (~50,000) required on average 0.134 μs per kpoint per triangular face. Hence, calculation of a 256 × 256 × 256 k-space matrix would take ~30 h. Further optimization of our C++-based implementation could certainly increase computation speed, especially if more efficient multithreading is included. Additionally, the FT of a polyhedron is highly parallelizable because the computation of the contribution of each face is completely independent and therefore well suited for an optimized graphics processing unit implementation. Though not explored in this study, reusing any shared computations between edge and face contribution calculations may further increase performance (26).

Currently, this phantom and the associated framework are unable to model coil sensitivity profiles. The coil profile produces a modulation in image space and therefore a convolution of the FT of the phantom with the FT of the coil sensitivity profile. Though a solution for this problem has been presented by Guerquin-Kern et al. (5) for a 2D analytical phantom, to our knowledge, no solution exists for 3D phantoms that results in a purely analytical expression. Another limitation of the current implementation of the polyhedral FT phantom involves tissue intensity gradients. More computationally intensive simulations are required as the presented equations assume piecewise constant regions of intensity. Nevertheless, we can approximate gradients by dividing objects into adjacent regions with gradually varying intensity. Furthermore, a simple reformulation of the polyhedral FT can be used to efficiently calculate these gradients (26).

## CONCLUSIONS

In this study, we reviewed the derivation of the FT of a polygon and polyhedron and described multithreaded implementation for the later. We evaluated accumulated error due to limited floating point precision and determined that it remained small even for polyhedra with many faces. We determined that the *k*-space error of polyhedral approximations of smooth surfaces decreased rapidly with increasing number of faces. We used polyhedra to construct realistic 3D brain and 4D torso phantoms and demonstrated the application of these phantoms in simulated 2D and 3D MRI acquisitions. A MATLAB implementation of the phantom can be downloaded at http://www.mathworks.com/matlabcentral/fileexchange/51911-realistic-analytical-polyhedral-mri-phantoms.

## APPENDIX

### Abbe Transform Theorem

For a function $\phi$ that satisfies the Helmhotz equation $\nabla^2\phi + m^2\phi = 0$, the N-dimensional volume integral of $\phi$ can be expressed as an N-1 dimensional integral of the divergence of $\phi$ over the surface enclosing the integration volume.

$$\int_V^{(N)} \phi \; d^N r = -\frac{1}{m^2} \int_{\partial V}^{(N-1)} \nabla\phi \cdot \hat{\boldsymbol{n}} \; d\sigma \qquad [A1]$$

**Proof:** Let V be a finite region in a space of N dimensions, $\partial V$ is the boundary of V, and $d\sigma$ an infinitesimal element of $\partial V$. Also, let $\phi(\boldsymbol{r})$, be a scalar valued function of the position vector $\boldsymbol{r}$ which satisfies the Helmholtz equation:

$$\nabla^2\phi + m^2\phi = 0 \qquad [A2]$$

Where $m$ is some constant.

$\phi$ satisfies the Helmholtz equation, implying $\phi = -\nabla^2\phi/m^2$. By substitution, the integral of the function $\phi$ over the volume V can be expressed as

$$\int_V^{(N)} \phi \; d^N r = -\frac{1}{m^2} \int_V^{(N)} \nabla^2\phi \; d^N r \qquad [A3]$$

Additionally, according to Gauss's theorem,

$$\int_V^{(N)} \nabla \cdot A \; d^N r = \int_{\partial V}^{(N-1)} A \cdot \hat{\boldsymbol{n}} \; d\sigma \qquad [A4]$$

If we let $A = \nabla\phi$ and $\nabla \cdot \mathrm{A} = \nabla^2\phi$, the above equation becomes

$$\int_V^{(N)} \nabla^2\phi \; d^N r = \int_{\partial V}^{(N-1)} \nabla\phi \cdot \hat{\boldsymbol{n}} \; d\sigma \qquad [A5]$$

Substituting this result into the right side of Equation [A3] produces the general case of the Abbe transform:

$$\int_V^{(N)} \phi \; d^N r = -\frac{1}{m^2} \int_{\partial V}^{(N-1)} \nabla\phi \cdot \hat{\boldsymbol{n}} \; d\sigma \qquad [A6]$$

### Applying the Abbe Transform to the Fourier Transform

To apply the Abbe transform to the Fourier integral, let $\phi(\boldsymbol{r}) = exp(-2\pi i \boldsymbol{k} \cdot \mathbf{r})$ be the kernel of the Fourier transform (FT) so the integral on the left of Equation [A1] is the N dimensional FT of the constant valued volume V. $\phi(\boldsymbol{r})$ satisfies the Helmholtz equation ($\boldsymbol{k}$ is a considered a constant) with $m^2 = (2\pi\boldsymbol{k})^2$, where k is the magnitude of the k space vector:

$$\nabla\phi = \nabla exp(-2\pi i \boldsymbol{k} \cdot \mathbf{r}) = -2\pi i \, exp(-2\pi i \boldsymbol{k} \cdot \mathbf{r})\boldsymbol{k} \qquad [A7]$$

$$\begin{aligned} \nabla^2\phi &= \nabla \cdot [-2\pi i \, exp(-2\pi i \boldsymbol{k} \cdot \mathbf{r})\boldsymbol{k}] \\ &= -(2\pi k)^2 exp(-2\pi i \boldsymbol{k} \cdot \mathbf{r}) \end{aligned} \qquad [A8]$$

$$\begin{aligned} \nabla^2\phi + m^2\phi = &-(2\pi k)^2 exp(-2\pi i \boldsymbol{k} \cdot \mathbf{r}) \\ &+ (2\pi k)^2 exp(-2\pi i \boldsymbol{k} \cdot \mathbf{r}) = 0. \end{aligned} \qquad [A9]$$

Thus, we can apply the Abbe transform to convert the N-dimensional Fourier volume integral of a constant

valued region V to an equivalent N-1 dimensional surface integral.

$$S(\boldsymbol{k}) = \int_V^{(N)} exp(-2\pi \boldsymbol{k}\cdot\boldsymbol{r})\ d^N r = \frac{i}{2\pi k^2}\int_{\partial V}^{(N-1)} exp(-2\pi i\boldsymbol{k}\cdot\mathbf{r})k\cdot\hat{\boldsymbol{n}}\ d\sigma \quad [A10]$$

## Derivation of the Fourier Transform of a Polygon

First, we will derive the closed form expression for the FT of a plane polygon. This expression will be used in the face contribution of the polyhedron to its overall transform.

In 2D, we wish to find the FT $S_{2D}(\boldsymbol{k})$ of a polygon $\rho(\boldsymbol{r})$ whose boundary is composed of E vertices $V_1 \cdots V_E$ oriented in a counterclockwise fashion, where $\rho(\boldsymbol{r}) = 1$ *if* $\boldsymbol{r}\epsilon P$, *else* $\rho(\boldsymbol{r}) = 0$. The Abbe transform expresses the FT $\rho(\boldsymbol{r})$ as a line integral along the boundary L where *dl* is a small segment of that boundary:

$$S_{2D}(\boldsymbol{k}) = \int_P^{(2)} exp(-2\pi i\boldsymbol{k}\cdot\boldsymbol{r})\ d^2 r = \frac{i}{2\pi k^2}\int_L^{(1)} exp(-2\pi i\boldsymbol{k}\cdot\boldsymbol{r})k\cdot\hat{\boldsymbol{n}}\ dl. \quad [A11]$$

Notice that $\hat{\boldsymbol{n}}$, the outward pointing normal to the edge of the polygon, and $\boldsymbol{r}$ are implicitly functions of the position on the boundary.

Let $\boldsymbol{L}_e = \boldsymbol{r}^{(V_{e+1})} - \boldsymbol{r}^{(V_e)}$ be the vector oriented in the direction of the e$^{th}$ edge, pointing from vertex $V_e$ to $V_{e+1}$ with the same length as the edge, where $\boldsymbol{r}^{(V_e)}$, $\boldsymbol{r}^{(V_{e+1})}$ are the position vectors associated with the $V_e$ and $V_{e+1}$ vertices of the polygon, respectively. $\hat{\boldsymbol{t}}_e = \boldsymbol{L}_e/L_e$ is the unit vector oriented in the direction of the e$^{th}$ edge. $\hat{\boldsymbol{n}}_e$ is a unit vector normal to the e$^{th}$ edge and pointing outward from the interior of the polygon. Additionally, the first and last vertices are connected, thus $V_{E+1} = V_1$ and likewise $V_E = V_0$.

A polygon's boundary can be broken up into edges and the line integral can be expressed as the sum of the contributions $S_e(\boldsymbol{k})$ from each of the edges.

$$S_{2D}(\boldsymbol{k}) = \sum_{e=1}^{E} S_e(\boldsymbol{k}) \quad [A12]$$

$$S_e(\boldsymbol{k}) = \frac{i}{2\pi k^2}\int_0^{L_e} exp(-2\pi i\boldsymbol{k}\cdot\boldsymbol{r})\boldsymbol{k}\cdot\hat{\boldsymbol{n}}_e\ dl \quad [A13]$$

Here, $\hat{\boldsymbol{n}}_e$ has replaced $\hat{\boldsymbol{n}}$ and is constant along a given edge.

A point on the e$^{th}$ edge can be parameterized by setting $\boldsymbol{r} = \boldsymbol{r}^{(V_e)} + \boldsymbol{t}_e l$, where *l* is the distance from the vertex $V_e$, which makes explicit the dependence of the position $\boldsymbol{r}$ on the boundary to the scalar variable *l*. The contribution from the right edge can be rewritten as

$$S_e(\boldsymbol{k}) = \frac{i}{2\pi k^2}\boldsymbol{k}\cdot\hat{\boldsymbol{n}}_e\, exp\left(-2\pi i\boldsymbol{k}\cdot\boldsymbol{r}^{(V_e)}\right)\int_0^{L_e} exp(-2\pi i\boldsymbol{k}\cdot\hat{\boldsymbol{t}}_e l)\ dl \quad [A14]$$

$$S_e(\boldsymbol{k}) = \frac{1}{(2\pi k)^2}\frac{\boldsymbol{k}\cdot\hat{\boldsymbol{n}}_e}{\boldsymbol{k}\cdot\hat{\boldsymbol{t}}_e} exp\left(-2\pi i\boldsymbol{k}\cdot\boldsymbol{r}^{(V_e)}\right)\left(1 - exp(-2\pi i\boldsymbol{k}\cdot\hat{\boldsymbol{t}}_e L_e)\right). \quad [A15]$$

Let $\boldsymbol{r}^{(C_e)} = \boldsymbol{r}^{(V_e)} + \hat{\boldsymbol{t}}_e L_e/2$ to rewrite $S_e$ in terms of phasors related to the midpoint of the edge.

$$S_e(\boldsymbol{k}) = \frac{1}{(2\pi k)^2}\frac{\boldsymbol{k}\cdot\hat{\boldsymbol{n}}_e}{\boldsymbol{k}\cdot\hat{\boldsymbol{t}}_e} exp\left(-2\pi i\boldsymbol{k}\cdot\left(\boldsymbol{r}^{(C_e)} - \frac{\hat{\boldsymbol{t}}_e L_e}{2}\right)\right)\left(1 - exp(-2\pi i\boldsymbol{k}\cdot\hat{\boldsymbol{t}}_e L_e)\right) \quad [A16]$$

$$S_e(\boldsymbol{k}) = \frac{1}{(2\pi k)^2}\frac{\boldsymbol{k}\cdot\hat{\boldsymbol{n}}_e}{\boldsymbol{k}\cdot\hat{\boldsymbol{t}}_e} exp\left(-2\pi i\boldsymbol{k}\cdot\boldsymbol{r}^{(C_e)}\right)\left(exp(\pi i\boldsymbol{k}\cdot\hat{\boldsymbol{t}}_e L_e) - exp(-\pi i\boldsymbol{k}\cdot\hat{\boldsymbol{t}}_e L_e)\right) \quad [A17]$$

$$S_e(\boldsymbol{k}) = \frac{i}{2\pi k^2} L_e\boldsymbol{k}\cdot\hat{\boldsymbol{n}}_e\frac{\sin(\pi\boldsymbol{k}\cdot\hat{\boldsymbol{t}}_e L_e)}{\pi\boldsymbol{k}\cdot\hat{\boldsymbol{t}}_e L_e} exp\left(-2\pi i\boldsymbol{k}\cdot\boldsymbol{r}^{(C_e)}\right) \quad [A18]$$

Summing over all edge contributions $S_e(\boldsymbol{k})$, we obtain the final form for the expression for the FT of a polygon:

$$S_{2D}(\boldsymbol{k}) = \frac{i}{2\pi k^2}\sum_{e=1}^{E} L_e\boldsymbol{k}\cdot\hat{\boldsymbol{n}}_e\frac{\sin(\pi\boldsymbol{k}\cdot\hat{\boldsymbol{t}}_e L_e)}{\pi\boldsymbol{k}\cdot\hat{\boldsymbol{t}}_e L_e} exp\left(-2\pi i\boldsymbol{k}\cdot\boldsymbol{r}^{(C_e)}\right),\ \boldsymbol{k}\neq 0. \quad [A19]$$

The above expression is finite as long as $\boldsymbol{k} \neq \mathbf{0}$ (ie, the center of *k*-space). The value at $\boldsymbol{k} = \mathbf{0}$ is equal to the area of the polygon. The area of a 2D planar polygon in terms of its vertices is

$$S_{2D}(\boldsymbol{k} = \mathbf{0}) = \frac{1}{2}\left|\sum_{e=1}^{E} r_x^{(V_e)} r_y^{(V_{e+1})} - r_x^{(V_{e+1})} r_y^{(V_e)}\right|. \quad [A20]$$

## Derivation of the Fourier Transform of a Polyhedron

We wish to find an analytical solution for the FT of a polyhedron

$$S(\boldsymbol{k}) = \iiint \rho(\boldsymbol{r}) exp(-2\pi i\boldsymbol{k}\cdot\boldsymbol{r})\ d^3 r \quad [A21]$$

where $\rho(\boldsymbol{r}) = 1$ *if* $\boldsymbol{r}\epsilon V$, *else* $\rho(\boldsymbol{r}) = 0$.

Using the Abbe transform, the volume integral in the FT of a polyhedron can be expressed as an integral over the surface $\partial V$.

$$S_{3D}(\boldsymbol{k}) = \iiint_V^3 exp(-2\pi \boldsymbol{k}\cdot\boldsymbol{r})\ d^3r = \frac{i}{2\pi k^2}\oint\!\!\oint_{\partial V}^{(2)} exp(-2\pi i k\cdot\boldsymbol{r})\boldsymbol{k}\cdot\hat{\boldsymbol{n}}\ d\sigma \quad [A22]$$

We wish to express the surface integral as the sum of the contributions of the faces of the polyhedron $S_f(\boldsymbol{k})$. Let $\boldsymbol{r} = \boldsymbol{r}^{0f} + \boldsymbol{r}_f$, where $\boldsymbol{r}_f$ is a vector in the plane of the face of $P_f$ whose origin is at $\boldsymbol{r}^{0f}$ and $\boldsymbol{r}^{0f}$ is the position vector of an arbitrarily chosen point $0f$ in the face $P_f$. Additionally, let $\widehat{\boldsymbol{N}}_f$ be the outward normal of the f$^{th}$ face and F be the total number of faces in the polyhedron.

Similar to the FT of a polygon, we can express the FT of a polyhedron as the sum of the contributions from each of its individual faces.

$$S_{3D}(\boldsymbol{k}) = \sum_{f=1}^{F} S_f(\boldsymbol{k}) \quad [A23]$$

$$S_f(\boldsymbol{k}) = \frac{i}{2\pi}\frac{\boldsymbol{k}\cdot\widehat{\boldsymbol{N}}_f}{k^2} exp(-2\pi i\boldsymbol{k}\cdot\boldsymbol{r}^{0f}) I_f(\boldsymbol{k}) \quad [A24]$$

$$I_f(\boldsymbol{k}) = \oint\!\!\oint_{P_f}^{(2)} exp(-2\pi i\boldsymbol{k}\cdot r_f)\ d^2r \quad [A25]$$

The integral $I_f(\boldsymbol{k})$ over the surface $P_f$ of the f$^{th}$ polygonal face can be evaluated using the formula for the FT of a polygon if $\boldsymbol{k}_f$ is substituted for $\boldsymbol{k}$, the projection of $\boldsymbol{k}$ onto the plane $P_f$ or the component of . that exists purely in the plane of $P_f$. Because $\boldsymbol{k}\cdot\boldsymbol{r}_f = (\boldsymbol{k}_f + \boldsymbol{k}_{f\ per})\cdot\boldsymbol{r}_f = \boldsymbol{k}_f\cdot\boldsymbol{r}_f$, $I_f(\boldsymbol{k}_f) = I_f(\boldsymbol{k})$ and $\boldsymbol{k}_f$ exists purely in the plane of the face (ie, the plane of integration), we can apply the formula for the FT of a polygon. Additionally, let $E_f$ be the number of edges of the f$^{th}$ face, let $\boldsymbol{r}^{(C_{fe})}$ be the position vector of the midpoint of the e$^{th}$ edge on the f$^{th}$ face, let $\widehat{\boldsymbol{t}}_{fe}$ be the unit vector in the direction of the e$^{th}$ edge of the f$^{th}$ face, let $L_{fe}$ be the length of the e$^{th}$ edge of the f$^{th}$ face, and let $\hat{\boldsymbol{n}}_{fe}$ be the outward normal to the e$^{th}$ edge of the f$^{th}$ face in the plane of the face.

$$I_f(\boldsymbol{k}) = I_f(\boldsymbol{k}_f) = \frac{i}{2\pi k_f{}^2}\sum_{e=1}^{E_f} L_{fe}\boldsymbol{k}_f\cdot\hat{\boldsymbol{n}}_{fe}\frac{\sin(\pi i\boldsymbol{k}_f\cdot\widehat{\boldsymbol{t}}_{fe}L_{fe})}{\boldsymbol{k}_f\cdot\widehat{\boldsymbol{t}}_{fe}} exp\left(-2\pi i\boldsymbol{k}_f\cdot\boldsymbol{r}_f{}^{(C_{fe})}\right) \quad [A26]$$

This expression has a singularity when $\boldsymbol{k}_f = \boldsymbol{0}$. In this case, $I_f$ is the area of the f$^{th}$ face, $P_f$. The following equalities can be used to rewrite the above equation in terms of $\boldsymbol{k}$ and $\boldsymbol{r}^{(C_e)}$ in the global coordinate system: $k_f{}^2 = k^2 - (\boldsymbol{k}\cdot\widehat{\boldsymbol{N}}_f)^2$, $\boldsymbol{k}_f\cdot\hat{\boldsymbol{n}}_{fe} = \boldsymbol{k}\cdot\hat{\boldsymbol{n}}_{fe}$, $\boldsymbol{k}_f\cdot\widehat{\boldsymbol{t}}_{fe} = \boldsymbol{k}\cdot\widehat{\boldsymbol{t}}_{fe}$, $\boldsymbol{k}_f\cdot\boldsymbol{r}_f{}^{(C_{fe})} = \boldsymbol{k}\cdot\boldsymbol{r}_f{}^{(C_{fe})}$, and $\boldsymbol{r}_f{}^{(C_{fe})} = \boldsymbol{r}^{(C_e)} - \boldsymbol{r}^{0f}$.

$$I_f(\boldsymbol{k}) = \frac{i}{2\pi[k^2 - (\boldsymbol{k}\cdot\widehat{\boldsymbol{N}}_f)^2]}\sum_{e=1}^{E_f} L_{fe}\boldsymbol{k}\cdot\hat{\boldsymbol{n}}_{fe}\frac{\sin(\pi i\boldsymbol{k}\cdot\widehat{\boldsymbol{t}}_{fe}L_{fe})}{\boldsymbol{k}\cdot\widehat{\boldsymbol{t}}_{fe}} \times exp\left(-2\pi i\boldsymbol{k}\cdot\boldsymbol{r}^{(C_{fe})}\right) \quad [A27]$$

The singularity at $\boldsymbol{k}_f = \boldsymbol{0}$ becomes a singularity at $\boldsymbol{k} = k\widehat{\boldsymbol{N}}_f$ because $\boldsymbol{k}$ perpendicular to the f$^{th}$ face implies $\boldsymbol{k}_f = \boldsymbol{0}$. In this case, $I_f(k\widehat{\boldsymbol{N}}_f)$ is the surface area $P_f$ of the polygonal face f given in the global coordinate system as

$$I_f(k = k\widehat{\boldsymbol{N}}_f) = P_f = \frac{1}{2}\left|\widehat{\boldsymbol{N}}_f\ \cdot\sum_{e=1}^{E}\{\boldsymbol{r}^{(V_{fe})}\times\boldsymbol{r}^{(V_{fe+1})}\}\right| \quad [A28]$$

Substituting $I_f$ into $S_f$ produces

$$S_f(\boldsymbol{k}\neq\boldsymbol{0},\ k\widehat{\boldsymbol{N}}_f) = -\frac{1}{(2\pi k)^2}\frac{\boldsymbol{k}\cdot\widehat{\boldsymbol{N}}_f}{k^2 - (\boldsymbol{k}\cdot\widehat{\boldsymbol{N}}_f)^2}\sum_{e=1}^{E_f} L_{fe}\boldsymbol{k}\cdot\hat{\boldsymbol{n}}_{fe}\frac{\sin(\pi\boldsymbol{k}\cdot\widehat{\boldsymbol{t}}_{fe}L_{fe})}{\pi\boldsymbol{k}\cdot\widehat{\boldsymbol{t}}_{fe}L_{fe}} \times exp\left(-2\pi i\boldsymbol{k}\cdot\boldsymbol{r}^{(C_{fe})}\right)$$

$$S_f(\boldsymbol{k} = k\widehat{\boldsymbol{N}}_f) = \frac{i}{2\pi}\frac{\boldsymbol{k}\cdot\widehat{\boldsymbol{N}}_f}{k^2} exp(-2\pi i\boldsymbol{k}\cdot\boldsymbol{r}^{0f})P_f \quad [A29]$$

where $\boldsymbol{r}^{0f}$ can be any arbitrary point in the plane of the face (eg, it can be set to the first vertex $\boldsymbol{r}^{(V_{f1})}$).

Thus, summing over the appropriate contribution of each face, factoring out $-1/(2\pi\boldsymbol{k})^2$ from $S_f$ to produce $S_f^*$ and substituting $\boldsymbol{r}^{(V_{f1})}$ for $\boldsymbol{r}^{0f}$, the final simplified expression for the FT of a polyhedron is:

$$S_{3D}(\boldsymbol{k}) = \begin{cases} -\frac{1}{(2\pi k)^2}\sum_{f=1}^{F} S_f^*(\boldsymbol{k}), & k\neq\boldsymbol{0} \\ V, & \boldsymbol{k}=\boldsymbol{0} \end{cases}$$

$$S_f^*(\boldsymbol{k}\neq k\widehat{\boldsymbol{N}}_f) = \frac{\boldsymbol{k}\cdot\widehat{\boldsymbol{N}}_f}{k^2 - (\boldsymbol{k}\cdot\widehat{\boldsymbol{N}}_f)^2}\sum_{e=1}^{E_f} L_{fe}\boldsymbol{k}\cdot\hat{\boldsymbol{n}}_{fe}\frac{\sin(\pi\boldsymbol{k}\cdot\widehat{\boldsymbol{t}}_{fe}L_{fe})}{\pi\boldsymbol{k}\cdot\widehat{\boldsymbol{t}}_{fe}L_{fe}} \times exp\left(-2\pi i\boldsymbol{k}\cdot\boldsymbol{r}^{(C_{fe})}\right)$$

$$S_f^*(\boldsymbol{k} = k\widehat{\boldsymbol{N}}_f) = -2\pi i\boldsymbol{k}\cdot\widehat{\boldsymbol{N}}_f exp\left(-2\pi i\boldsymbol{k}\cdot\boldsymbol{r}^{(V_{f1})}\right)P_f$$

$$P_f = \frac{1}{2}\left|\widehat{\boldsymbol{N}}_f\ \cdot\sum_{e=1}^{E_f}\{\boldsymbol{r}^{(V_{fe})}\times\boldsymbol{r}^{(V_{fe+1})}\}\right| \quad [A30]$$

where V is the volume of a polyhedron (14), which is given in terms of its vertices as

$$S_{3D}(k=\boldsymbol{0}) = V = \frac{1}{6}\left|\left[\sum_{f=1}^{F}(\boldsymbol{r}^{V_{f1}}\cdot\widehat{\boldsymbol{N}}_f)\left|\widehat{\boldsymbol{N}}_f\cdot\left\{\sum_{e=1}^{E_f}\boldsymbol{r}^{(V_{fe})}\times\boldsymbol{r}^{(V_{fe+1})}\right\}\right|\right]\right| \quad [A31]$$

## REFERENCES

1. Irwan R, Russel I, Sijens P. Fast 3D coronary artery contrast-enhanced magnetic resonance angiography with magnetization transfer contrast, fat suppression and parallel imaging as applied on an anthropomorphic moving heart phantom. Magn Reson Imaging 2006;24:895–902.

2. Segars WP, Tsui BMW. MCAT to XCAT: the evolution of 4-D computerized phantoms for imaging research. Proc IEEE 2009;97:1954–1968.
3. Koay CG, Sarlls JE, Özarslan E. Three-dimensional analytical magnetic resonance imaging phantom in the Fourier domain. Magn Reson Med 2007;58:430–436.
4. Shepp L. Computerized tomography and nuclear magnetic resonance. J Comput Assist Tomogr 1980;4:94–107.
5. Guerquin-Kern M, Lejeune L, Pruessmann KP, Unser M. Realistic analytical phantoms for parallel magnetic resonance imaging. IEEE Trans Med Imaging 2012;31:626–636.
6. Zhu Y, Narayanan SS, Nayak KS. Flexible Dynamic Phantoms for Evaluating MRI Data Sampling and Reconstruction Methods. In Proceedings of the 21st Annual Meeting of ISMRM, Salt Lake City, Utah, USA, 2013. p. 4355.
7. Collins DL, Zijdenbos AP, Kollokian V, Sled JG, Kabani NJ, Holmes CJ, Evans AC. Design and construction of a realistic digital brain phantom. IEEE Trans Med Imaging 1998;17:463–468.
8. Wissmann L, Segars WP, Kozerke S. MRXCAT: Realistic Numerical Phantoms for Cardiac MRI. In Proceedings of the 22nd Annual Meeting of ISMRM, Milan, Italy, 2014. p. 2413.
9. Tobon-Gomez C, Sukno FM, Bijnens BH, Huguet M, Frangi AF. Realistic simulation of cardiac magnetic resonance studies modeling anatomical variability, trabeculae, and papillary muscles. Magn Reson Med 2011;65:280–288.
10. Kwan R, Evans A, Pike G. An extensible MRI simulator for post-processing evaluation. Vis Biomed Comput 1996;1131:135–140.
11. Yoder DA, Zhao Y, Paschal CB, Fitzpatrick JM. MRI simulator with object-specific field map calculations. Magn Reson Imaging 2004;22:315–328.
12. Ngo TM, Fung GSK, Tsui BMW, McVeigh ER, Herzka DA. Three Dimensional Digital Polyhedral Phantom Framework with Analytical Fourier Transform and Application in Cardiac Imaging. In Proceedings of the 19th Annual Meeting of ISMRM, Montreal, Quebec, Canada, 2011. p. 1310.
13. Komrska J. Algebraic expressions of shape amplitudes of polygons and polyhedra. Optik (Stuttg) 1988;80:171–183.
14. Arvo J. Graphics Gems II (Graphics Gems–IBM, No. 2). (1st edition). San Diego, CA: Academic Press; 1991.
15. Marcus DS, Wang TH, Parker J, Csernansky JG, Morris JC, Buckner RL. Open Access Series of Imaging Studies (OASIS): cross-sectional MRI data in young, middle aged, nondemented, and demented older adults. J Cogn Neurosci 2007;19:1498–1507.
16. Bazin P-L, Pham DL. Topology-preserving tissue classification of magnetic resonance brain images. IEEE Trans Med Imaging 2007;26:487–496.
17. Han X, Pham DL, Tosun D, Rettmann ME, Xu C, Prince JL. CRUISE: cortical reconstruction using implicit surface evolution. Neuroimage 2004;23:997–1012.
18. Computational Geometry Algorithms Library. http://www.cgal.org. Accessed on Oct 1, 2013.
19. Segars WP, Sturgeon G, Mendonca S, Grimes J, Tsui BMW. 4D XCAT phantom for multimodality imaging research. Med Phys 2010;37:4902–4915.
20. Segars WP, Lalush DS, Tsui BM. A realistic spline-based dynamic heart phantom. IEEE Trans Nucl Sci 1999;46:503–506.
21. Ju T. Robust repair of polygonal models. ACM Transactions on Graphics 2004;23:888–895.
22. Pipe J. Motion correction with PROPELLER MRI: application to head motion and free-breathing cardiac imaging. Magn Reson Med 1999;42:963–969.
23. Guo L, Sayin O, Derbyshire JA, Herzka DA. Navigator-Free Self-Gated Dynamic Cine Imaging Using 2D Cartesian Golden Step Phase Encoding. In Proceedings of the 20th Annual Meeting of ISMRM, Melbourne, Victoria, Australia, 2012, 515.
24. Guo L, Segundo AJM, Derbyshire JA, Carrino JA, Herzka DA. Self-Navigated Kinematic Imaging of the Knee. In Proceedings of the 19th Annual Meeting of ISMRM, Montreal, Quebec, Canada, 2011, 384.
25. Guo L, McVeigh ER, Lederman RJ, Derbyshire JA, Herzka DA. Dual-Projection Cardiac and Respiratory Self-Navigated Cine Imaging Using SSFP. In Proceedings of the 18th Annual Meeting of ISMRM, Stockholm, Sweden, 2010, 78.
26. Han S, Herzka DA. Polyhedral Phantom Framework with Analytical Fourier Transform with Intensity Gradients. In Proceedings of the 23rd Annual Meeting of ISMRM, Toronto, Ontario, Canada, 2015. p. 2472.
27. FFTW (Fastest Fourier Transform in the West). http://www.fftw.org. Accessed on July 10, 2013.

## SUPPORTING INFORMATION

Additional Supporting Information may be found in the online version of this article.

**Supporting Movie S1.** Reconstruction of a simulated acquisition of the torso phantom over 90 points spanning a cardio-respiratory cycle. Jitter occurring at the body surface is due to the thin slice plane and large polygonal faces in the body surface mesh. This jitter can be reduced by increasing the number of faces during the body surface NURBs to mesh conversion or increasing the thickness of the slice plane. In contrast, jitter at the lungs is due to differences between repaired self-intersecting meshes produced by Polymender and Meshlab. Removing this jitter requires a NURBS to triangular mesh conversion that does not produce self-intersections or a tool that can correct meshes for all time frames with minimal differences.