

Software-Based Algorithm for Modeling and Correction of Gradient Nonlinearity Distortions in Magnetic Resonance Imaging

Tom S. Lee, Keith E. Schubert

*Department of Computer Science and Engineering
California State University, San Bernardino
San Bernardino, CA, USA*

Email: tlee@permedics.com, schubert@csci.csusb.edu

Reinhard W. Schulte

*Department of Radiation Medicine
Loma Linda University Medical Center
Loma Linda, CA, USA*

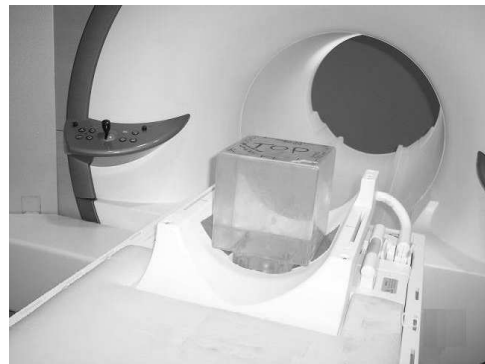
Email: rschulte@dominion.llumc.edu

Abstract– Functional radiosurgery is a noninvasive stereotactic technique that requires magnetic resonance image (MRI) sets with high spatial resolution. Gradient nonlinearities introduce geometric distortions that compromise the accuracy of MRI-based stereotactic localization. We present a gradient nonlinearity correction method based on a cubic phantom MRI data set. The approach utilizes a sum of spherical harmonics to model the geometrically warped planes of the cube and applies the model to correct arbitrary image sets acquired with the same scanner. In this paper, we give a detailed description of the Matlab distortion correction program, report on its performance in stereotactic localization of phantom markers, and discuss the possibility to accelerate the code using General-Purpose Computing on Graphics Processing Units (GPGPU) techniques.

Keywords– *Magnetic Resonance Imaging; Distortion Correction Algorithm; Matlab; Functional Radiosurgery.*

I. INTRODUCTION

We describe the implementation of a software-based method to correct gradient nonlinearity distortion in Magnetic Resonance Imaging (MRI), to provide submillimeter accuracy in localization of anatomical regions for functional radiosurgery. Although current methods are successful in handling gradient nonlinearity distortion [1], [2], [3], none of them have been proven to provide the high level of geometric accuracy required to *confidently* perform high-precision functional radiosurgery on brain targets. We have developed a software package for a phantom-based distortion method originally suggested by Langlois, et al. [4]. This report describes the structure and implementation of the code, as well as first results of its performance in stereotactic localization.



(a) Cube phantom.



(b) Lucy[®] phantom.

Figure 1. Phantoms in Siemens Sonata MRI scanner.

II. MATERIALS AND METHODS

A. Phantoms and Stereotactic Equipment

MRI data sets of two different phantoms were acquired in this work: the first phantom was used for characterizing the scanner distortion, and the second for testing and verification of the distortion correction method. The phantom utilized for distortion characterization is a Plexiglas cube filled with oil, Figure 1(a). The images obtained represent the signal

response of the oil, corresponding to the interior dimensions of the phantom ($159.50\text{mm} \times 159.70\text{mm} \times 158.11\text{mm}$). The second phantom, the Lucy[®] quality assurance phantom (Standard Imaging, Inc.), was placed inside a Leksell Coordinate Frame Model G and MR Indicator (Elekta, Sweden) to perform localization of 20 MRI-compatible target markers, Figure 1(b). A certified metrology laboratory inspected the marker locations. All MRI images were acquired on a 1.5 T whole body scanner (Sonata Maestro, Siemens) with a standard head coil.

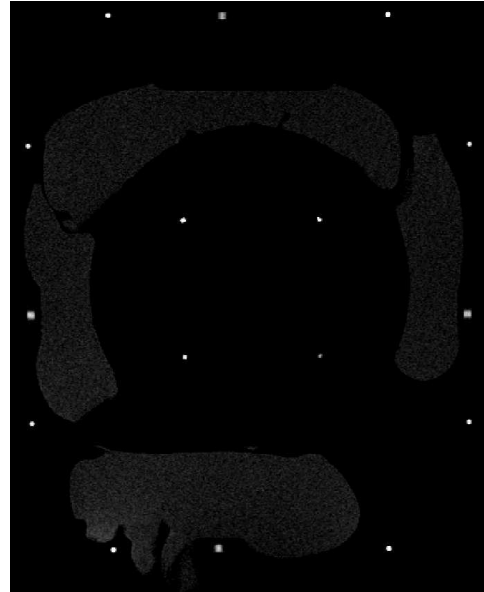
The cube phantom was centered on the isocenter of the scanner with respect to its lateral and longitudinal position to within 1mm. Due to space limitations within the head coil, the cube center had to be placed approximately 10 mm above the scanner isocenter. The Lucy[®] phantom was placed at approximately the center of the head coil and at three discrete longitudinal positions: at isocenter, 50mm superior to isocenter, and 50mm inferior to isocenter in order to test the accuracy of distortion-corrected localization both in the center and toward the periphery of the magnetic gradient field.

The automated 3D calibration of the shim coils was performed to correct for B_0 field inhomogeneities prior to scanning. Two sets of sequences were performed: one with and one without internal correction for gradient nonlinearity, to compare the effectiveness of the internal scanner correction to that of the software-based MRI gradient nonlinearity distortion correction. Scan parameters for the cube phantom were as follows: 512×512 matrix, 0.391mm pixel size, 200mm field of view, 104 slices of 2.0mm thickness. Axial, coronal, and sagittal sequences were acquired. Scan parameters for the Lucy[®] phantom were as follows: 512×512 matrix, 0.586mm pixel size, 300mm field of view, 112 slices of 2.0mm thickness. Axial and coronal sequences were acquired. Figure 2 shows representative axial and coronal images of the Lucy[®] phantom, while Figure 3 shows representative axial, coronal, and sagittal images of the cube phantom.

B. Computational Algorithm

The gradient nonlinearity distortion correction was written in Matlab version 7.0.4. Matlab was chosen as the programming environment because of its strong support for DICOM data types, as well as robust image processing functions and data visualization tools.

B.1. Data Quality Assurance. Prior to calculating the distortion model, the data underwent a series of image processing and quality checks. These checks had three goals: (1) eliminate as much image noise as possible, (2) remove extraneous and unwanted features in each MR image, and (3) ensure the data is accurate for future calculations.



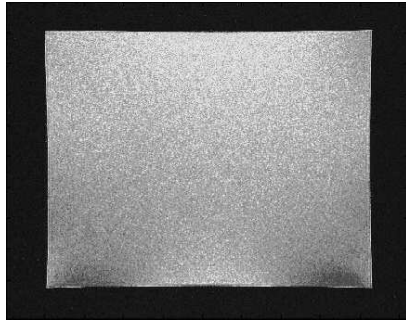
(a) Axial scan.



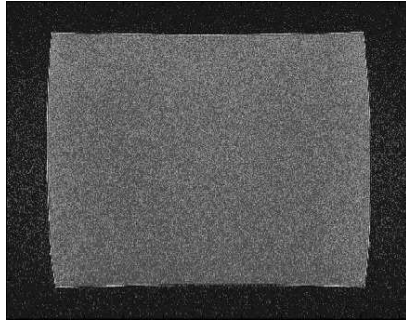
(b) Coronal scan.

Figure 2. Example scans of Lucy[®] phantom, showing fiducial points (edges) used for stereotactic localization of MRI markers (center).

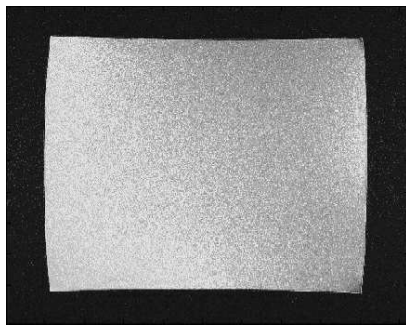
B.2. Selection of Useful Slices. The distortion correction was only based on slices that contained a full view of the phantom. To select useful slices such as those in Figure 3, the algorithm analyzed the distribution of pixel intensities of a slice near the center of the slab, which always contained a full view of the phantom. Using this reference slice, the algorithm compared the pixel intensities of each slice to determine the range of the study that contained the phantom.



(a) Axial.



(b) Coronal.



(c) Sagittal.

Figure 3. Examples of useful slices of oil-filled phantom, with distortion clearly visible in the outer regions.

This effectively eliminated slices that did not contain the phantom, or contained a partial view of the phantom. The indexes of the first and last useful slices were reported, and slices outside this range were excluded from analysis.

B.3. Edge Detection. Edge detection was performed to provide the representation of the distorted physical edges of the phantom for the gradient nonlinearity distortion correction. The Canny edge detector provided by Matlab was chosen because it was least sensitive to noise, and the edge images produced were sharper and more accurate, thus delivering the most consistent results from study to study when compared to other edge detection methods implemented in Matlab. Matlab's Canny edge detector automatically calculates a threshold for edge detection; however,

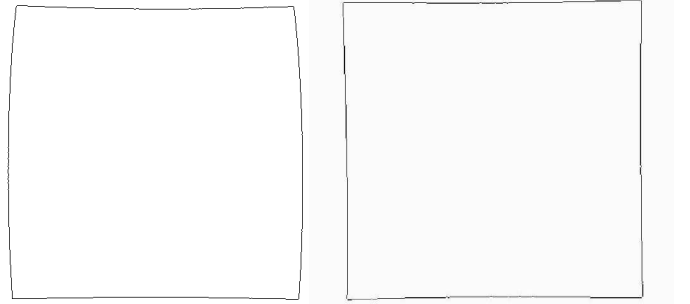


Figure 4. Representative examples of edge images.

this threshold considers every single intensity shift within the image, and in doing so, finds fictitious edges within the phantom. We implemented a threshold algorithm that converged on the optimal threshold value in an iterative series of edge detection trials. The threshold algorithm used the reference slice chosen in the previous step, and automatically configured its settings (threshold increment and stop condition) based on the strength of the image gradient across the edges. Starting with Matlab's automatically calculated threshold, the threshold algorithm incremented the threshold in small successive steps (0.5 - 0.75). As the threshold was increased, the number of fictitious edges decreased, leaving only the strong image features behind. The optimal threshold was assumed when the number of edge points did not change by more than 1% over several successive iterations (5-10, depending on the stop condition). This optimized threshold was then used with Matlab's Canny edge detector on the entire set of useful slices. By utilizing Canny edge detection on several MRI image studies (≈ 300 -400 images), we found that Canny's edge detection performance was far superior to the other edge detection methods included with Matlab: Sobel, Prewitt, Roberts, Laplacian of Gaussian, and zero-cross. Canny preserved the physical edges of the phantom in the edge images far more accurately than other methods. Although Canny created high-quality edge images ideal for our work (see Figure 4), the remaining unwanted image features needed to be dealt with separately, as described below.

B.4. Removing Extraneous Features. The edge detection threshold did not remove all extraneous or noisy features from the edge images. The cube phantom featured a *drain plug*, which corrupted the edge images. A specially-designed *drain plug removal algorithm* erased the plug and drew a straight-line approximation of the affected regions, see section B.5. In some slides oil leaks corrupted one quadrant of the image, so an *edge-verification algorithm* was implemented, see section B.6. Edge images were analyzed for *air bubbles* within the oil adhering to the edges of the phantom creating rounded defects in the detected edges. A

bubble removal algorithm erased these bubbles and drew a straight-line approximation to correct the affected region, see section B.7. Remaining minute *holes* in the edges were corrected with a *hole repair algorithm* that analyzed the four edges in all edge images for discontinuities, filling any gaps, see section B.8. With these quality checks and repair algorithms, the data were ensured to be free of practically all defects, providing an accurate representation of the distorted phantom surfaces.

B.5. Drain Plug Removal. One image feature in the scan of our specific phantom that was not removed during edge detection was the phantom’s drain plug—a screw located underneath the center of the posterior face. The threads of the screw were present in edge images acquired from axial and sagittal sequences. Tests showed that failure to remove the image of the plug would result in improper calculations in all subsequent data processing steps. Therefore, we implemented a *plug removal algorithm*.

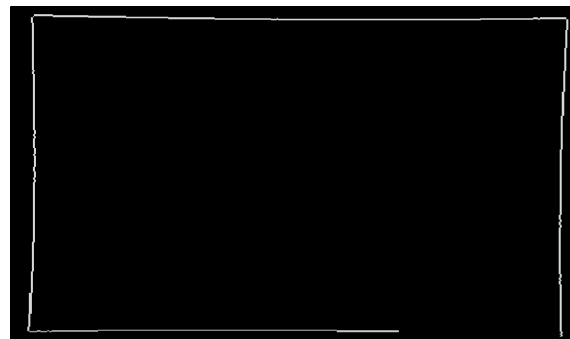
In the first step of the drain-plug removal process, edge images were oriented such that the plug appeared on the bottom face of the image. Removal of the plug was then performed by placing a window that has a width of 25% of the phantom width about the center column of each image. The plug removal algorithm removed all image data within this window, leaving behind a hole in the top and bottom phantom edges. Both holes were located by the algorithm and filled by a straight line. Since the deleted areas were located in the center of these faces, the distortion was small; at worst, the corrected portion of the edges was shifted by one pixel to either side of the edge. It was found that in all phantom scans, the plug removal algorithm properly connected the edge with a straight line, without introducing any gaps.

B.6. Edge Verification and Repair. The next step involved the detection of incomplete edge images, i.e., those with missing edge data. For example, in Figure 5, the phantom had leaked oil along the left posterior surface. The edge detection resulted in a large defect in the bottom right corner of several edge images.

In the edge-verification step, all edge images in an image sequence were checked for completeness. First, the location and size of the phantom was calculated by our *edge-verification algorithm*. Based on this information, the phantom edge image was split into four equal quadrants, and the total number of horizontal and vertical edge points in each quadrant was counted. In a complete edge image, the phantom will contain nearly the same number of edge points in each quadrant. On the other hand, in an incomplete edge image, the number of edge points in one quadrant will be different. These images, and the quadrant that appears anomalous, were flagged. Based on user preferences, problematic images could either be discarded from further



(a) Slice with oil leakage in left posterior corner.



(b) Detected edges of slice with oil leakage.

Figure 5. Phantoms in Siemens Sonata MRI scanner.

consideration, or the images could be repaired. Image repair was accomplished by copying and grafting an intact quadrant from the opposite side in place of the flagged quadrant. It was found that the resulting image was very similar to that of a “natural” edge image, and was sufficient for use in further calculations. This repair worked successfully for all slices affected by the leaking oil, shown in Figure 5.

B.7. Detection and Removal of Air Bubbles. Due to the use of oil as a medium for the phantom, it was virtually impossible to remove all air bubbles. Small air bubbles that float within the confines of the phantom were not problematic, but those attached to the phantom surfaces distorted the phantom edges. Since the signal response seen in the MR images was that of the oil, air bubbles along the edges appeared as concave defects of varying size. Air bubbles attached to the phantom corners gave the false impression of a rounded corner. The study performed on the leaking phantom mentioned above resulted in a large air bubble in the right anterior corner of the phantom, as shown in Figure 6(a).

Air bubbles were detected in several steps. First, the locations of the four corners in each image were determined using our implementation of a *corner detection algorithm*. The algorithm defined a corner as the point shared by a horizontal and vertical edge (i.e., where those two edges

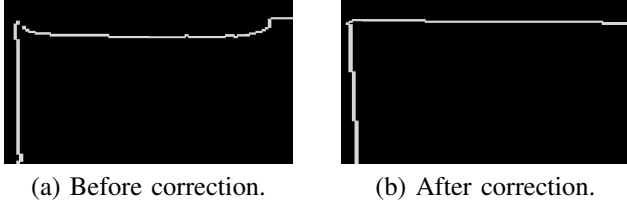


Figure 6. Demonstration of bubble correction on large corner bubble.

met). The corners were located by scanning up and down the vertical edges of the phantom (depending on the corner to be located) until the start of the adjacent horizontal edge was found. When the horizontal edge was found, *and* there were no additional edge points found along the vertical edge, the true corner was considered located. This extra step was executed to avoid the algorithm falsely identifying an air bubble along that edge, as air bubbles may resemble corners. By scanning beyond a possible corner location, the algorithm disregard air bubbles along an edge as possible corners, and only reported the location of the true corner.

The corner data was then used by our *edge tracking algorithm*, which was implemented to detect air bubbles. Starting from one corner, it moved along the edge, pixel by pixel, recording the location of each edge point. Horizontal edges were scanned from left to right, and vertical edges were scanned from top to bottom. The edge tracker stopped once it reached the opposite corner. Next, the edge *slope* was calculated for edge intervals of 5 points. Along a “normal” edge, the edge slope did not exceed one pixel over this range. On the other hand, an air bubble caused a sharp positive (>1 pixel) rise in slope, followed by a similar drop on the opposite side of the bubble. The range between the rise and drop of the slope was considered as the limits of the bubble.

If the bubble was located on the edge rather than in a corner, it was erased, creating a hole in the edge, and the image was processed by our *hole repair algorithm*, described in the next section. On the other hand, air bubbles located in a corner were treated by replacing the entire corner with a copy of the opposite bottom corner, to avoid inserting another bubble in place of the original. The likelihood of having air bubbles in *both* bottom corners of one image was considered as *very* low. The bottom corners were tested for air bubbles first, and were corrected by the mirrored opposite bottom corner, if necessary. Next, the top corners were tested for bubbles and replaced by the corresponding bottom corner, if air bubbles were detected.

Figure 6 demonstrates the results of removing the large air bubble present in the study with the leaking phantom.

B.8. Hole Repair. Holes could occur in an edge either as a result of noise, or some previous action taken by another correction algorithm (i.e., bubble repair). A hole is simply

a discontinuity in an edge. Holes that occurred as a result of noise were quite small, usually spanning only one or two pixels. On the other hand, holes that resulted from previous correction procedures could be many pixels wide.

We implemented a *hole-repair algorithm*, which drew a straight-line approximation of the edge to fill the hole. First, holes were located by using our edge tracking algorithm. The error condition of the edge tracker occurred when a hole was reached. Here, the edge tracker returned the list of edge points, up to the start location of the hole, and quit.

The hole-repair algorithm then located the end location of the hole by scanning beyond the hole start location in the direction of the edge until the opposite side of the hole was found. Next, the algorithm calculated the horizontal and vertical distance between the edge points bordering the hole. A straight line that connects the border points of the hole was then calculated and used to fill the hole defect.

C. Plane Calculation

C.1. Midplanes. The correction method was based on a set of calculated “ideal” planes. These planes represented the appearance of the phantom faces without distortion. Each pair of opposite edge points in the edge images was used to calculate a midpoint. Midpoint calculation along an entire edge produced a *midline*, see Figure 7. Edge lengths could vary depending on the characteristics of the distortion in each image, so observing opposite edges of slightly different lengths was acceptable; the algorithm removed unpaired points to obtain edges of equal length. Two midlines were calculated for each edge image: one horizontal, one vertical. Due to the symmetry of gradient nonlinearity distortion, these midlines were nearly straight, providing a suitable representation of the orientation of the undistorted edges in each image, disregarding the positioning of the edges.

Studies typically contained around 30,000 midpoints for each of the three pairs of phantom surfaces. The *midplane*, representing the ideal shape, size, and orientation of the particular surface of the phantom, was found by calculating the plane that resulted in the best fit of the midplane data set in the least squares sense. The Hessian normal form of a plane is: $\bar{N}^T p = d$, where $\bar{N} = [N_x \ N_y \ N_z]^T$ is the normal vector to the plane with $\|\bar{N}\| = 1$, d is distance of the plane from the origin, and p is the set of vectors satisfying the plane equation. For example, fitting a plane to the top and bottom edges of the phantom in an axial study results in a plane oriented in xz . The component N_y is close to 1 in this case, so one may divide both sides of the equation by it to produce a component normalized form:

$$\left[\begin{array}{ccc} \frac{N_x}{N_y} & 1 & \frac{N_z}{N_y} \end{array} \right] \left[\begin{array}{c} x \\ y \\ z \end{array} \right] = \frac{d}{N_y}$$

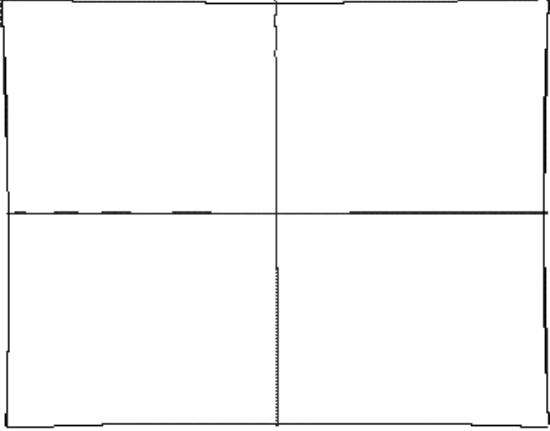


Figure 7. Calculated midlines (center horizontal and vertical straight lines) in a typical edge image.

$$\begin{bmatrix} -C_x & 1 & -C_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = N^T p = C_c \quad (1)$$

The introduction of minus signs on the coefficients, C_x and C_z , allows the equation to be easily written as $y = C_x x + C_z z + C_c$. With this form, one needs to store only three components rather than four. Based on the determined orientation, the algorithm organized the data into a large $n \times 3$ matrix and a large n -element array, with n being the number of midpoints. The $n \times 3$ matrix with row vectors of the form $[x \ z \ 1]$ was the *design matrix*, denoted by A , and the n -element array of y values was the *observation vector*, denoted by b . The resulting linear least squares problem was of the form $A\chi = b$ where $\chi = [C_x \ C_z \ C_c]^T$ is the vector of the plane coefficients. Solving this equation for χ yielded the midplane coefficients.

C.2. Ideal Planes. The midplanes described the correct orientation of the ideal phantom planes, but needed to be shifted along the corresponding normal vector by one-half the physical dimensions of the phantom to their correct location in 3-dimensional space. The three phantom dimensions were derived from dimensional inspection and stored prior to algorithm execution. Shifting the midplane only involved calculating a new offset term as the other coefficients must remain the same to maintain the midplane's original orientation. To calculate the offset, consider the Hessian normal form of a plane in terms of the coefficients defined in Eq. 1 and a point, $p_0 = [0 \ C_c \ 0]^T$ that lies on the midplane in our example. It is possible to determine the position of the corresponding point p_s on the shifted plane as follows:

$$\begin{aligned} p_2 &= \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = p \pm \delta \bar{N} \\ &= \begin{bmatrix} \frac{\pm -C_x \delta}{\sqrt{C_x^2 + 1 + C_z^2}} \\ C_c \pm \frac{\delta}{\sqrt{C_x^2 + 1 + C_z^2}} \\ \frac{\pm -C_z \delta}{\sqrt{C_x^2 + 1 + C_z^2}} \end{bmatrix} \end{aligned} \quad (2)$$

where δ is one half the phantom dimension, and the “ \pm ” determines positive or negative shift needed to specify both ideal planes resulting from the midplane. Note that we have to use the unit vector $\bar{N} = \frac{N}{\|N\|}$ and not N in order to maintain the correct distance. Inputting p_s into the component normalized plane expression, Eq. 1, makes it possible to solve for the offset term for the shifted planes.

$$\begin{aligned} C_s &= N^T p_s \\ &= N^T (p_0 \pm \delta \bar{N}) \\ &= N^T p_0 \pm \delta N^T \bar{N} \\ &= C_c \pm \delta N^T \frac{N}{\|N\|} \\ &= C_c \pm \delta \|N\| \end{aligned} \quad (3)$$

Therefore, the offset term to obtain the ideal planes from the midplane is equal to $\delta \sqrt{C_x^2 + 1 + C_z^2}$. The same method was followed for the other planes. The result was a set of six ideal planes, one for each of the six square faces of the phantom. These ideal planes represented the faces of the phantom in the absence of gradient nonlinearity distortion.

D. Distortion Modeling

The final stage involved performing the actual gradient nonlinearity distortion correction. In MRI, the spatial distribution of nuclear spin density is encoded in frequency (k) space using three orthogonal gradient fields. A Fourier transform of the radiofrequency signal obtained by exciting the nuclear spins in the presence of the gradient fields yields a reconstruction of spin density in the spatial domain. Because the spatial dependence of the gradient components is nonlinear, the assumption of linearity during reconstruction results in distorted images.

The distortion model, describing the spatial dependence of the three gradient fields, was based on a finite sum of *sum of spherical harmonics* (when expressed in spherical coordinates) [2], or a sum of homogeneous polynomials (when expressed in Cartesian coordinates) [4]. Using the ideal planes obtained previously, theoretical “undistorted” data points were calculated using the ideal plane equations and acquired data points. The undistorted data points were then used to calculate the coefficients of the distortion

model. Three sets of coefficients were necessary, one for each dimension. Once the model was known it was applicable to all studies performed on the particular scanner.

Let x_i , y_i , and z_i be the distorted coordinates of a point i measured by an MRI scanner. Further, let \bar{x}_i , \bar{y}_i , and \bar{z}_i be the undistorted coordinates that correspond to x_i , y_i , and z_i , that is, $(\bar{x}_i, \bar{y}_i, \bar{z}_i)$ is the point on the ideal plane that produced the point (x_i, y_i, z_i) in the image. Including spherical harmonics up to second order, the distortion model, expressed in Cartesian coordinates, becomes [4]:

$$\begin{aligned} \alpha_i = & \bar{\alpha}_i (1 + K_{\alpha_0} (\bar{x}_i^2 + \bar{y}_i^2) + K_{\alpha_1} \bar{z}_i^2 + \\ & K_{\alpha_2} \bar{z}_i^2 (\bar{x}_i^2 + \bar{y}_i^2) + K_{\alpha_3} (\bar{x}_i^2 + \bar{y}_i^2)^2 \\ & + K_{\alpha_4} \bar{z}_i^4) \end{aligned} \quad (4)$$

where $\alpha \in \{x, y, z\}$. Note that all the terms in these equations are known except the K_{x_j} , K_{y_j} , and K_{z_j} terms—the desired distortion parameters. Solving for the K_{x_j} requires using the ideal planes whose normal is in the x direction, i.e., the yz planes, and similarly for the K_{y_j} and K_{z_j} terms. Since the original example was for the top and bottom edges of the images in an axial study results, i.e., a plane oriented in xz , the solution for this example will follow that case. Note that \bar{y}_i can be calculated using the ideal plane. Since with proper orientation of the cube the distortion for this face is essentially all in the y direction, the ideal x and z coordinates (\bar{x}_i and \bar{z}_i) are assumed to be the same as the distorted (x_i and z_i).

$$\begin{aligned} C_c \pm \delta \|N\| &= N^T p_{ideal} \\ &= \begin{bmatrix} -C_x & 1 & -C_z \end{bmatrix} \begin{bmatrix} x_i \\ \bar{y}_i \\ z_i \end{bmatrix} \\ \bar{y}_i &= C_c \pm \delta \|N\| + C_x x_i + C_z z_i \end{aligned} \quad (5)$$

Knowing the ideal point $(\bar{x}_i, \bar{y}_i, \bar{z}_i)$ and the distorted points (x_i, y_i, z_i) , substitute into the spherical harmonics expansion, noting the linearity in the K terms. Using all n points of both planes oriented in xz yields:

$$TK_y = \Delta Y \quad (6)$$

where,

$$\begin{aligned} K_y &= \begin{bmatrix} K_{y_0} \\ \dots \\ K_{y_4} \end{bmatrix} \\ \Delta Y &= \begin{bmatrix} (y_0 - \bar{y}_0) \\ \vdots \\ (y_{2n-1} - \bar{y}_{2n-1}) \end{bmatrix} \\ T &= \begin{bmatrix} T_{0,0} & \dots & T_{0,4} \\ \vdots & \ddots & \vdots \\ T_{2n-1,0} & \dots & T_{2n-1,4} \end{bmatrix} \end{aligned}$$

$$\begin{aligned} T_{i,0} &= \bar{y}_i (\bar{x}_i^2 + \bar{y}_i^2) \\ T_{i,1} &= \bar{y}_i (\bar{z}_i^2) \\ T_{i,2} &= \bar{y}_i (\bar{x}_i^2 + \bar{y}_i^2) \bar{z}_i^2 \\ T_{i,3} &= \bar{y}_i (\bar{x}_i^2 + \bar{y}_i^2)^2 \\ T_{i,4} &= \bar{y}_i (\bar{z}_i^4) \end{aligned}$$

Solving for K_y was accomplished using linear least squares by QR factorization (Matlab “\” operator). This process was repeated for the other distortion parameters K_x and K_z . Once the distortion parameters were obtained, they were applied to MR images of the performance study, described below.

III. RESULTS AND DISCUSSION

A. Performance Study

The performance of the distortion correction code was tested by localizing the 20 markers in the Lucy[®] phantom, which had known stereotactic coordinates derived from precision dimensional inspection. The corrected locations of the markers were compared to the measured physical locations. This step required performing a *stereotactic transformation*, to convert the coordinates in the images to a stereotactic coordinate system created by the fiducial points on the attached stereotactic frame. Target localization was performed in two ways: The first method obtained the coordinates of the MRI fiducial points and marker points (Figure 2) from the distorted images and their corrected positions were calculated by iteratively solving Eq. 4 using the *minerr* function of Mathcad version 13. The corrected fiducial marker coordinates were then processed by a custom stereotactic localization program written in Mathcad and the parameters of a 3D stereotactic transformation (rotation matrix and translation vector) were obtained according to the algorithm described in [6]. The second method processed the entire MR image set in the following way. Starting with an empty voxel grid of identical size to the original study, the location of each voxel in the distorted space was calculated according to Eq. 4 and the mean MRI signal intensity from nearest voxel neighbors to the distorted voxel location was calculated and assigned to the undistorted voxel location. This way, the entire undistorted image space was rebuilt. The distortion-corrected MRI set was then read into Odyssey[®] (PerMedics, Inc.), a commercially-available radiation treatment planning system, and the built-in stereotactic localization feature of Odyssey[®] was used to determine the stereotactic marker coordinates. Both methods gave similar results within voxel accuracy.

The *localization error*, defined as the x, y, and z components of the vector between the MRI-determined and the metrology-lab measured marker locations was determined

for three scenarios: (1) error associated with our phantom-based MRI gradient nonlinearity distortion correction, (2) error when no correction was performed, and (3) error associated with a built-in internal scanner correction method for gradient field non-linearities. The latter feature was turned off for the first two scenarios.

B. Numerical Results

Figure 8 demonstrates the mean and standard deviation of the localization error of the 20 MRI markers. Localizing the markers without any distortion correction resulted in stereotactic localization errors typically between 1.0mm and 2.0mm. This accuracy decreased greatly with increasing Z-offset from gradient isocenter, with errors reaching nearly 3.0mm. At gradient isocenter, the internal correction yielded accuracies better than 1.0mm, but was insufficient in providing this quality at ± 50 mm Z offset. In contrast, the phantom-based MRI gradient nonlinearity distortion correction provided consistent submillimeter accuracy in all test cases, even for ± 50 mm Z offset. Considering the worst case, the correction method yielded accuracies of approximately 0.6 ± 0.3 mm. In the best case, the phantom-based MRI gradient nonlinearity distortion correction reduced localization error to nearly 0.0 ± 0.3 mm. Considering each scan image, these values correlated to approximately one pixel in the worst case, and to $\frac{1}{2}$ pixel in the best case.

IV. CONCLUSIONS

Our results confirm that correcting gradient nonlinearity distortion – a primary source of distortion in MRI – brings the accuracy in MRI-based stereotactic localization into the submillimeter domain. Without any correction the error typically exceeds 1.0mm and can reach up to 3.0mm. Our correction method provided consistent submillimeter accuracy in localizing the targeting markers even when the object was ± 50 mm off isocenter, while the internal correction method did not consistently provide submillimeter accuracy.

V. CURRENT AND FUTURE WORK

A. Correction for Other Sources of Distortion

In the present work, we have focused on the nonlinearity of the gradient fields superimposed on the main field of the MRI scanner as the main source of distortion in the periphery of the scanner images. This is most important for stereotactic localization techniques employing external fiducial markers. Another source of distortion, which depends on the presence of the scanned object in the scanner, is introduced by perturbation of the magnetic field due to

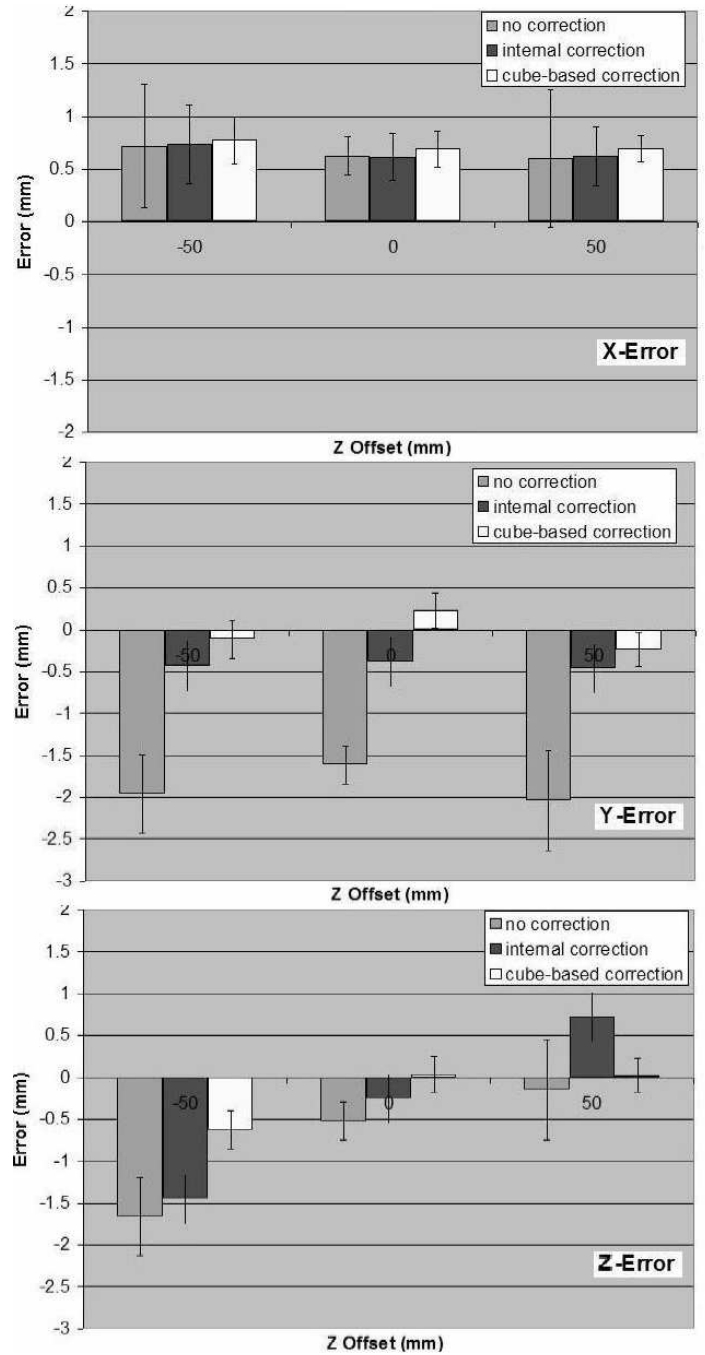


Figure 8. Stereotactic localization error of 20 markers with respect to each coordinate axis, with and without distortion correction.

local variations in the magnetic susceptibility. Inside the brain, the amount of this local distortion is relatively small and a smooth spatial function, but it can be quite large at interfaces at boundaries between high- and low susceptibility regions around the air-filled sinuses of the head. MRI with echo planar imaging (EPI) sequences, which are widely

used in functional and perfusion MRI because of their short acquisition time, are known to be particularly prone to geometric and intensity distortions due to susceptibility inhomogeneity.

A discussion of the many methods that have been suggested to correct for susceptibility-induced field distortions is beyond the scope of this chapter. In general, we feel that the need for additional correction methods needs to be tailored to the particular application. In our application, stereotactic localization relies on accurate localization of external fiducial markers and anatomical structures inside the brain. Our results show that corrections of susceptibility-related distortions beyond the automatic shimming procedure on the scanning object may not be required. However, it is possible that for other scenarios, for example, those using fast sequences such as EPI, a combination of gradient field nonlinearity and magnetic field map corrections may give the best results.

B. Scanner and Phantom

Up to this point, we have only tested our method with one scanner model (Sonata Maestro, Siemens). Several new 3T MRI scanner models (GE and Siemens) will be utilized for further testing, to confirm the correction model is valid for both the different scanner and the stronger magnetic field. Since the existing cube phantom is relatively limited in size and the fiducial points of the MR indicator are up to 10 cm beyond the cube surfaces, we are contemplating the use of a larger cube phantom (30 cm side length) or, alternatively, to place the existing phantom in locations further away from the gradient isocenter, for more accurate distortion modeling.

In this work, we have assumed that the gradient isocenter coincides with the origin of the MRI-based coordinate system as specified by the DICOM header, which may not always be the case. In discussions with the vendor of the tested scanner, we were informed that the exact location of the gradient isocenter is not known. Assuming the gradient isocenter at the wrong location compromises the accuracy of the distortion correction model. In the future, it is planned to include the unknown location of the gradient isocenter in the distortion model and to find its position using a least square fitting routine.

C. GPGPU

Although the gradient nonlinearity distortion correction has demonstrated its effectiveness, the speed at which the calculations are performed is quite slow. On average, this Matlab program requires 20-30 minutes to execute on a typical PC. While this is not a ridiculous amount of time, reducing the computation time to minutes or even seconds will greatly increase the effectiveness and feasibility of utilizing the software in a clinical environment.

To significantly reduce the computation time of the MRI gradient nonlinearity distortion correction method, specific portions of the software will be converted to use *General-Purpose Computing on Graphics Processing Units (GPGPU)* techniques. Current GPU hardware is specialized for compute-intensive, highly parallel computation, thereby allowing the GPU to attain nearly 500 GFLOPS, compared to 40 GFLOPS for current high-end desktop CPUs [7]. Typical achieved speedup with modern GPU hardware is $100\times$ or more on typical compiled programs [8]. With an increasing number of scientists and engineers utilizing GPU hardware for medical applications, especially CT [9], [10], [11], we believe that the same techniques can also be applied to our MRI distortion correction method. We are currently evaluating NVIDIA GPU hardware and NVIDIA's *CUDA (Compute Unified Device Architecture)* SDK. Accelerating this distortion correction method with GPU hardware will yield great performance gains. Since Matlab is a scripting language, computational performance is quite limited. Merely converting from Matlab to a compiled language such as C will yield $50\times$ speedup [5]. According to other literature, accelerating the compiled C code using GPGPU techniques can yield an additional $100\times$ speedup or more [7]. Combining the two together – converting the Matlab code to C with portions accelerated on GPUs – can yield nearly $500\times$ speedup, or more. Rather than waiting 20-30 minutes, the GPU-accelerated implementation could yield results in as little as 2.4-3.6 seconds. Even though these performance values are estimated, obtaining this level of performance is not unreasonable, when considering the results shown in our past research [5], as well as currently published GPGPU applications [7], [8], [9], [10], [11].

The conclusion goes here. this is more of the conclusion

References

- [1] D. Kondziolka, *Functional Radiosurgery*, Neurosurgery, vol. 44, pp. 12-20, 1999.
- [2] S.J. Doran, L. Charles-Edwards, S.A. Reinsberg, and M.O. Leach MO, *A Complete Distortion Correction for MR Images: I. Gradient Warp Correction*, Phys. Med. Biol., vol. 50, pp. 1343-1361, 2005.
- [3] D. Wang, W. Strugnell, G. Cowin, D.M. Doddrell, and R. Slaughter, *Geometric Distortion in Clinical MRI Systems Part II: Correction Using a 3D Phantom*, J. Magn. Reson. Imaging, vol. 22, pp. 1223-1232, 2004.
- [4] S. Langlois, M. Desvignes, J.M. Constans, and M. Revenu, *MRI Geometric Distortion: A Simple Approach to Correcting the Effects of Non-linear Gradient Fields*, J. Magn. Reson. Imaging, vol. 9, pp. 821-831, 1999.
- [5] Lee, T.S. *Software-Based Gradient Nonlinearity Distortion Correction*. MSc Thesis, California State University San Bernardino, December 2006.

- [6] K. Weaver, V. Smith, J.D. Lewis, B. Lulu, C.M. Barnett, S.A. Leibel, P. Gutin, D. Larson, and T. Phillips, *A CT-based computerized treatment Planning system for I-125 stereotactic brain implants* Int. J. Radiation Oncology Biol. Phys. vol. 18, pp. 445-454, 1990.
- [7] NVIDIA Corp. *CUDA Compute Unified Device Architecture: Programming Guide*. v1.1, 2007.
- [8] Luebke, David. *High Performance Computing with CUDA*. SUPERCOMPUTING 2007 Presentation, November 2007.
- [9] H. Scherl, B. Keck, M. Kowarschik, and J. Hornegger. *Fast GPU-Based CT Reconstruction using CUDA*. Nuclear Science Symposium Conference Record, Vol. 6, 4464-4466, 2007.
- [10] K. Mueller, F. Xu, and N. Neophytou, *Why Do Commodity Graphics Hardware Boards (GPUs) Work So Well for Acceleration of Computed Tomography?* in SPIE Electronic Imaging Conference, San Diego, 2007, (Keynote, Computational Imaging V).
- [11] N. Neophytou, F. Xu, and K. Mueller. *Hardware Acceleration vs. Algorithmic Acceleration: Can GPU-Based Processing Beat Complexity Optimization for CT?* SPIE Medical Imaging '07, San Diego, 2007.