MPI with Nonideal Selection Fields

EEE 473 - Medical Imaging - Final Project Report
Department of Electrical and Electronics Engineering
Bilkent University, Ankara, Turkey

Boran Kılıç - 22103444 Sait Sarper Özaslan - 22002861 Mehmet Emre Uncu - 22003884

Presentation Video: https://youtu.be/SAnV1orDnY4?si=iz6cd01ojxXuDtDr

***Abstract*—This project explores the impact of non-ideal selection fields on Magnetic Particle Imaging (MPI) using superparamagnetic iron oxide nanoparticles (SPIOs). MPI relies on the nonlinear magnetization response of SPIOs at a field-free point (FFP) for high-resolution imaging. The study investigates two non-ideal selection field conditions: a shifted selection field center and an inhomogeneous field gradient. X-space reconstruction was employed to analyze the resulting artifacts under these non-ideal conditions. Simulations using MATLAB were performed on phantom datasets, including point source and vasculature models. Evaluation metrics such as Full-Width-Half-Maximum (FWHM) and displacement resolution were used to quantify image degradation. Results showed that non-ideal selection fields introduce geometric warping, resolution loss, and positional shifts in the reconstructed images.**

I. Introduction

Magnetic particle imaging (MPI) is an imaging modality that uses superparamagnetic iron oxide nanoparticles (SPIO) as tracers. MPI utilizes the SPIOs nonlinear magnetization response to a magnetic field as an imaging modality, forming the basis of MPI. SPIOs nonlinear behavior is most pronounced around zero magnetic field, and MPI utilizes this point named field-free point (FFP) where there is essentially no magnetic field [1]. This principle allows MPI to achieve high resolution and remain largely unaffected by the depth of surrounding materials, unlike modalities like ultrasound and PET, which rely on signal attenuation for contrast. Additionally, MPI differs from CT and MRI as its imaging format is inherently functional rather than anatomical. MPI has been successfully applied in areas such as stem cell tracking, cancer imaging, and drug release monitoring [1].

MPI uses two ways to reconstruct an image: system function reconstruction and X-space reconstruction. This project focuses on the latter for reconstruction, which focuses on the Langevin response of the SPIOs at FFPs to scan and image along a predetermined path [2].

Under ideal situations, MPI employs a selection field magnitude of 3T to 7T. The chosen magnetic field increases linearly from one point to another, and is zero at the origin [1]. This project aims to implement a nonideal selection field and explain how responses of SPIOs change according to nonidealities. The project seeks to explain these responses, also known as selection-field-induced warping, and show how they may cause resolution loss and possible shift the positions of the SPIOs [3].

II. Methodology

A. X-space Reconstruction and Selection Field

X-space reconstruction can be summarized as a direct image reconstruction method in MPI, which maps the received signal onto the spatial domain by leveraging the nonlinear magnetization response of superparamagnetic nanoparticles at the Field-Free Point (FFP) without requiring a system function [4]. The imaging equation in X-space reconstruction can be defined as:

$$\begin{aligned}\text{IMG}(x_s(t)) &= \frac{s(t)}{B_1 \cdot m \cdot k \cdot G \cdot \dot{x}_s(t)} \\ &= \rho(x) * \left(\dot{\mathcal{L}}(kGx)\right)\Big|_{x=x_s(t)}\end{aligned}$$

- $\text{IMG}(x_s(t))$: The reconstructed MPI image at the instantaneous position of the Field-Free Point (FFP), $x_s(t)$.
- $s(t)$: The received signal from the nanoparticles as a function of time.
- B_1, m, k, and G: Represent the sensitivity of the receiver coil, the magnetic moment of the superparamagnetic nanoparticles, a proportionality constant related to the excitation field strength, and the gradient of the magnetic selection field, respectively, can be considered as constants.
- $\dot{x}_s(t)$: The velocity of the Field-Free Point (FFP).
- $\rho(x)$: The spatial distribution of the nanoparticles.
- $\dot{\mathcal{L}}(kGx)$: The derivative of the Langevin function, which describes the linear magnetization response of the nanoparticles to the applied magnetic field in the ideal case.
- $x = x_s(t)$: Evaluation of the convolution at the position of the Field-Free Point (FFP), $x_s(t)$.

The final form of the imaging equation is a convolution of the SPIO distribution with the derivative of the Langevin function at $x = x_s(t)$.

$$\int \rho(x') \cdot \dot{\mathcal{L}}(kG(x - x'))\, dx'\Big|_{x=x_s(t)}$$

As SPIO distribution is independent of selection field, focus is on the Langevin function instead. Given the nature of the

Langevin function, the derivative will yield high values during changes from one saturation point to another, which the FFP creates. This relation in convolution is mapped as $x - x_s(t)$ [4].

In the general imaging equation, the selection field can be described by this relation in the ideal case:

$$H_s = Gx$$

FFP is formed by adding a time-varying magnetic field $H_o(t)$. In such a case, the FFP is formed at:

$$H(x,t) = G(x_s(t) - x) = 0$$

where

$$x_s(t) = \frac{H_0(t)}{G}$$

The same equations can be derived for the y-direction similarly. A form of the ideal selection field can be seen in Fig. 1-A. There are no selection fields chosen in the z-direction for the simplicity of the analysis.

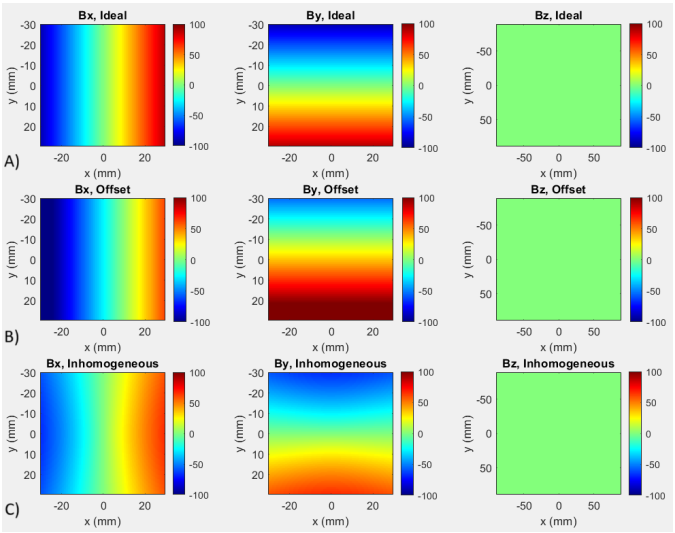

Fig. 1. Selection fields in x-, y-, and z-directions at z = 0, A) for ideal case. B) for shifted case. C) for inhomogeneous case.

In such a case, changing the Gx relation from an ideal case to a non-ideal one can create warping artifacts and shifts during the reconstruction.

B. Chosen Nonidealities

This project introduces two different nonideal selection fields:

- Selection field center may not be in the origin
- Selection field may decay non-linearly from origin to edges

1) Shift from Center: Starting with the first one, a shift from the origin can be caused by physical components of the MPI machine. The placement of the components, such as coils or magnets, may not be perfectly aligned and selection field may be off from the center [5].

The equation of FFP can be generated as follows:

$$H_s = G(x + offset)$$

Whose FFP in response turns into:

$$H(x,t) = G(x_s(t) - x - offset) = 0$$

This indicates an offset in negative value will move the reconstructed image to the right and becomes the new center.The same methodology can also be proposed in the y direction; a positive offset will cause the system to move towards negative direction during the reconstruction of the images. In Fig. 1-B, a negative offset in x-direction and positive offset in y-direction can be seen in the left and center figures respectively.

2) Inhomogeneous Selection Field: Moving to the latter selection field, the selection field may show an inhomogeneous behavior, such as an exponential decreasing change from center to edges. This is a possible case as most MPI machines are designed in a certain way where they work best under a particular field of view (FOV). As one gets to the edge and outside of the FOV, the reconstructed image quality decreases [3].

The equation of FFP, in this case, needs to be considered in 2D:

$$H_s = G \cdot x \cdot \exp\left(-k \cdot \sqrt{X^2 + Y^2}\right)$$

Whose FFP in response turns into:

$$H(x,t) = G\left(x_s(t) - x \cdot \exp\left(-k \cdot \sqrt{X^2 + Y^2}\right)\right) = 0$$

Focusing on the FFP point:

$$x_s(t) = x \cdot \exp\left(-k \cdot \sqrt{X^2 + Y^2}\right)$$

This behavior introduces a compression factor, where the reconstructed positions of SPIOs at larger distances from the center are scaled by the field's exponential decay and the linear dependence on x. Specifically, the reconstructed position X_r is compressed inward relative to the actual position X_c by the factor:

$$\text{Compression Factor} = \frac{x \cdot \exp\left(-k \cdot \sqrt{X_c^2 + Y_c^2}\right)}{x \cdot \exp\left(-k \cdot \sqrt{0}\right)}$$

Simplifying:

$$\text{Compression Factor} = \exp\left(-k \cdot \sqrt{X_c^2 + Y_c^2}\right)$$

The compressed selection fields can be seen in Fig. 1-C, where a compression towards the center can be seen visibly. This compression causes SPIOs near the edges of the FOV to appear closer to the center in the reconstructed image, even with accurate modeling of the inhomogeneous selection field. The inclusion of the x-dependent term in the field introduces a spatial scaling effect, further modulating the reconstructed positions along the x-direction.

Though the analysis was explicitly done for x-direction, the same analysis can be said for y-direction as well.

C. Set Information Regarding Modeling

For modeling the effect of non-ideal selection fields, a set of assumptions and constants has been realized. The constants and assumptions are as follows:

- 12.5 nm Core Radius
- Image size of 60x60 mm
- Gradient Strength: 3 Tesla
- FOV is chosen same as the image size.
- Drive field in x-direction:

$$B_{\text{DriveX}} = \sin\left(2\pi f_{\text{DriveX}} t\right)$$

- Drive Field in y-direction:

$$B_{\text{DriveY}} = \text{sawtooth}\left(2\pi \frac{1}{t_{\text{end}} + \Delta t} t\right)$$

  where sawtooth frequency is much higher so that scan be moved up to the next line before the next sweep starts.
- For appropriateness, no noise was injected into the system to focus on the artifacts caused by non-ideal selection gradients.

D. Dataset

The phantom images in the dataset were generated in MATLAB or manually through Microsoft Paint. There are 3 types of phantom images used for the simulation:

- A point SPIO at an arbitrary point,
- Equally spaced phantom with 25 SPIOs from edges to center,
- A vessel phantom hand-drawn in Microsoft Paint.

E. Code

Some parts of the code regarding reconstruction, such as picking units and scanning in x-direction through time-varying magnetic field, were inspired by an open-source MPI educational simulation GitHub repository named "OS-MPI\Educational_Simulation" [6]. The inspired Matlab document was under the name of "X_Space2D_Demo.m" [6]. However, the rest of the code was written by the project members.

For the necessary parts of the code, ChatGPT 4o was utilized at generating figures more accurately. Used prompts were:

- Generate Matlab figure code from the variables given below with empty labels and title.
- How can I obtain the colorbar for the figures?

F. Evaluation Metrics

For the evaluation part of the project, PSF resolution and displacement resolution were used.

1) PSF Resolution: The PSF resolution utilized the Full-Width-Half-Maximum (FWHM) of the Point Spread Function (PSF) for ideal and non-ideal cases, where FWHM is calculated theoretically and measured in both directions. The FWHM values for both directions are calculated as a function of saturation magnetization of the nanoparticle $\mathbf{M}_{sat}$, and/or particle diameter d by [7]:

$$\text{FWHM}_T \approx \frac{25 k_B T}{\mu_0 G \pi M_{sat} d^3}$$

$$\text{FWHM}_N \approx \frac{57 k_B T}{\mu_0 G \pi M_{sat} d^3}$$

Where, k_B is Boltzmann's constant and T is absolute temperature. It is evident from the FWHM equations above that the normal to the direction of the drive field has a FWHM value that is $2.3\times$ larger. In this project, the x-direction corresponds to the tangential direction, whereas the y-direction corresponds to the normal direction. For the ideal case, gradient values are chosen as $G_{xx} = G_{yy} = 3$ T/m. Theoretical FWHMs are found as: FWHM_x = 2.7 mm and FWHM_x = 6.3 mm.

The FWHM is defined as twice the distance between the two points x and x_{max} where the function value at x is half of its maximum ($\frac{1}{2} \cdot f_{\text{max}}$):

$$\frac{f(x)}{f_{\text{max}}} = \frac{1}{2}$$

and the FWHM is measured by:

$$\text{FWHM} = 2 \cdot |x_{\text{max}} - x|$$

This measures the width of the PSF at half of its peak value, providing a clear resolution metric for ideal and non-ideal conditions.

2) Displacement Resolution: The displacement resolution is calculated using the normalized Euclidean distance:

$$d = \frac{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}{n},$$

where:

- (x_1, y_1) and (x_2, y_2) are the coordinates of the ideal and nonideal points, respectively.
- n: total number of points considered in the measurement.

This formula computes the average Euclidean distance between ideal and non-ideal points by dividing the total distance by the number of points, which normalizes the result.

III. Results

For the 2D phantom with point source SPIOs placed at 15 mm separations, as seen on the left in Fig. 2, x-space MPI images were obtained under ideal and two non-ideal selection fields. In Fig. 2, the ideal image obtained appears in the middle. When calculating the B field, it was scaled with a coefficient of 1.1 to ensure that the FFP covers the entire FOV. As a result, the resulting images have minor scale differences compared to the original phantom image. On the

right in Fig. 2, the x-space image obtained under gradient offset, which is the first non-ideal selection field, appears. While obtaining this image, a 2mm offset was used for both -x and +y directions. Compared to the ideal image, it is observed that the resulting image shifts upward to the left at an angle of 45 degrees. By increasing the offset amount to 9mm in the -x direction and 12mm in the +y direction, the image on the right Fig. 3 was obtained. When this image is examined, it is seen that the image shifts as the offset values increase.

The inhomogeneous gradient was used as another non-ideal selection field. In the image on the right of Fig. 4 obtained with the coefficient k = -3.5. In the image, the point sources are misregistered, resulting in an apparent warping. This can be understood from the points in the corners being pushed towards the center. While the point in the center remains constant, the push increases as it approaches the edges of the FOV. As a result, the corners of the image are rounded. When the coefficient k was changed to -10, the image on the right in Fig. 5 was obtained. As seen here, as the coefficient increases, the SPIOs located further from the center are pushed more; thus, the image's corners are further rounded.

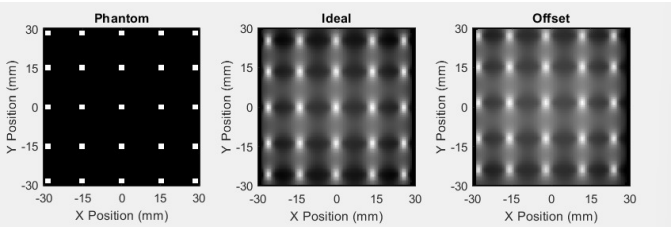

Fig. 2. Left: Phantom with point source SPIOs placed at 15 mm separations. Middle: X-space reconstructed MPI image for the ideal selection field. Right: X-space reconstructed MPI image for the case of selection field with 2mm gradient offsets in both -x and +y directions.

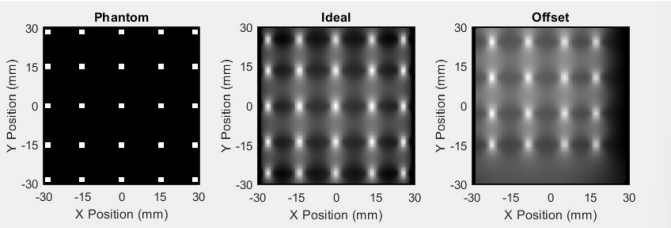

Fig. 3. Left: Phantom with point source SPIOs placed at 15 mm separations. Middle: X-space reconstructed MPI image for the ideal selection field. Right: X-space reconstructed MPI image for the case of selection field with 9mm and 12mm gradient offsets in -x and +y directions respectively.

A. PSF Resolution

In order to be able to assess the resolution, Full-Width-Half-Maximum (FWHM) of the Point Spread Functions (PSF) of SPIOs that are placed at the center and near the edge are measured for ideal and non-ideal cases.

Table I shows that the FWHM values for the non-ideal cases (offset and inhomogeneous gradients) are almost identical to the ideal case. For example, at the center, the FWHM$_x$ is

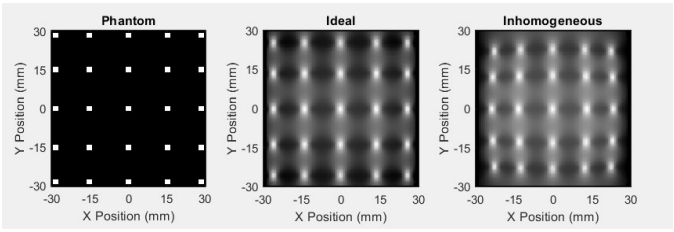

Fig. 4. Left: Phantom with point source SPIOs placed at 15 mm separations. Middle: X-space reconstructed MPI image for the ideal selection field. Right: X-space reconstructed MPI image for the case of selection field with inhomogeneous gradient when k = -3.5.

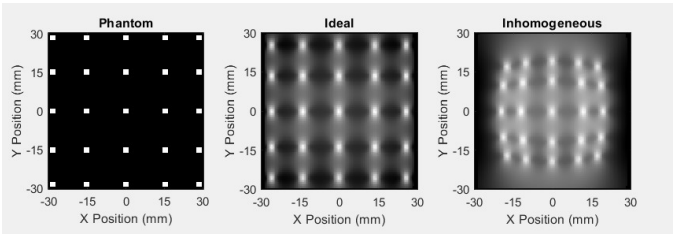

Fig. 5. Left: Phantom with point source SPIOs placed at 15 mm separations. Middle: X-space reconstructed MPI image for the ideal selection field. Right: X-space reconstructed MPI image for the case of selection field with inhomogeneous gradient when k = -10.

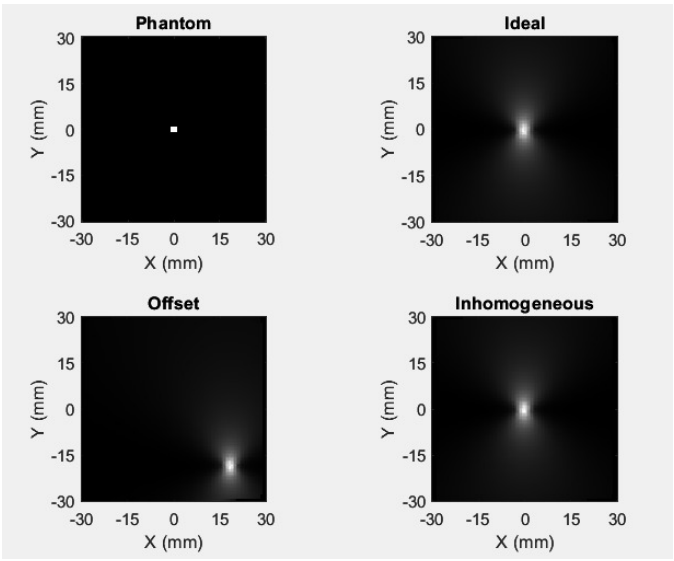

Fig. 6. Top Left: Point source SPIO in the center of the FOV. Top Right: PDF for the ideal selection field. Bottom Left: PSF for the case of selection field with offset gradient. Bottom Right: PSF for the case of selection field with inhomogeneous gradient

TABLE I
FWHM VALUES FOR SPIOS NEAR THE EDGE AND ON THE CENTER

Location	Condition	FWHM$_x$ (mm)	FWHM$_y$ (mm)
Near Edge	Ideal	2.25	6.00
	Offset	3.00	6.50
	Inhomogeneous	2.25	6.45
Center	Ideal	3.00	6.00
	Offset	2.62	5.50
	Inhomogeneous	3.00	6.00

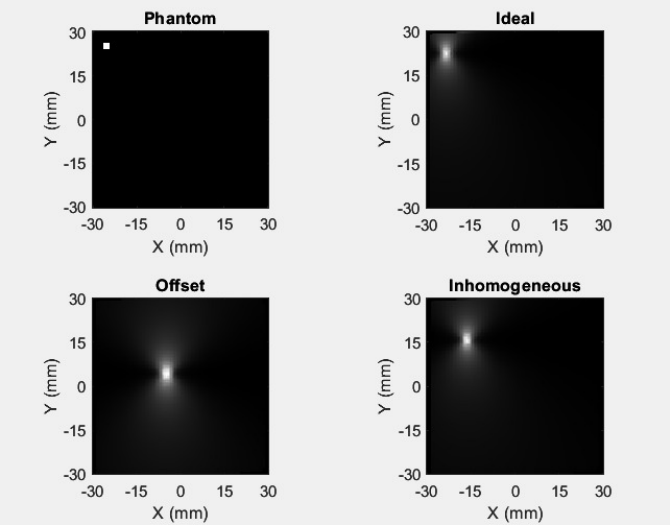

Fig. 7. Top Left: Point source SPIO 5mm apart from the up and left edges of the FOV. Top Right: PSF for the ideal selection field. Bottom Left: PSF for the case of selection field with offset gradient. Bottom Right: PSF for the case of selection field with inhomogeneous gradient

3.00 mm for both the ideal and inhomogeneous cases, while it is slightly lower (2.62 mm) in the offset case. Similarly, near the edge, the FWHM$_x$ is 2.25 mm for both the ideal and inhomogeneous cases, while it increases slightly to 3.00 mm in the offset case. It is evident from Table I that the FWHM values in the y-direction are observed generally larger than those in the x-direction, as expected based on the theoretical results. For instance, near the edge, FWHM$_y$ is 6.00 mm in the ideal case and increases to 6.50 mm in the offset case, while it only slightly increases to 6.45 mm in the inhomogeneous case. Similarly, at the center, FWHM$_y$ is consistently 6.00 mm across the ideal and inhomogeneous cases, with a slight reduction to 5.50 mm in the offset case. It is observed that the inhomogeneous gradient results have slightly smaller FWHM values near the edge compared to the center, which may be attributed to the increased compression closer to the phantom's boundary.The reconstructed images, along with the phantom images, provided in Fig. 6 and Fig. 7, visually confirm these observations.

B. Displacement Resolution

Error calculation was made between the reconstructed ideal image and reconstructed images with offset gradient and inhomogeneous gradient using Euclidean distances to calculate displacement resolution. The reconstructed ideal gradient, 120x120 pixels, was divided into 25 squares of 24x24 pixels. The x and y coordinates of the highest value pixel of each square were recorded. Thus, 25 (x, y) pairs were obtained. Then, this process was repeated for offset gradient and inhomogeneous gradient.

First, Euclidean distances were calculated for three different areas for the (x, y) values of the ideal gradient and offset gradient. The first of the regions is all 25 squares framed in red, the second is the middle nine squares framed in blue, and

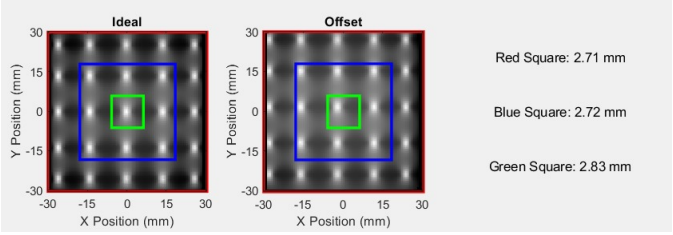

Fig. 8. Left: X-space reconstructed MPI image for the ideal selection field with framed areas. Right: X-space reconstructed MPI image for the case of selection field with 2mm gradient offsets in both -x and +y directions with framed areas.

the last is the middle square framed in green. As can be seen in Fig. 8 the calculated error values are close to each other since the offset process is applied equally to all frames on the image. This value obtained is approximately 2.8 mm since the offset value is 2 mm in both axes.

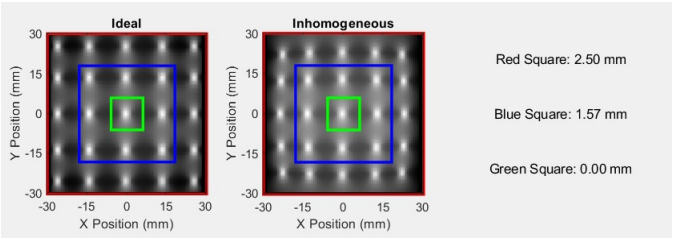

Fig. 9. Left: X-space reconstructed MPI image for the ideal selection field with framed areas. Right: X-space reconstructed MPI image for the case of selection field with inhomogeneous gradient when k = -3.5 with framed areas.

Afterward, Euclidean distance calculations were made the same way between the ideal and inhomogeneous gradients for three different areas. When the results in Fig. 9 are examined, the error for the middle square framed in green is calculated as 0mm. This is because inhomogeneity depends on distance, and there is no change at the (0, 0) point. As we move away from the origin, the distortion increases, and therefore, the error increases. While the error was 1.57mm for the middle nine squares surrounded by blue, the error increased to 2.50mm for the 25 squares surrounded by red.

C. Demonstration on a Vasculature Phantom

To illustrate the impact of artifacts caused by a non-ideal selection field in a more realistic scenario, imaging simulations were conducted using the vasculature phantom depicted in Fig. 10.

In Fig. 10 the top left panel shows the vasculature phantom model, representing the structure of blood vessels used for the simulations. The top right panel illustrates the resulting image under an ideal selection field, which accurately reconstructs the vasculature without artifacts. In contrast, the bottom left panel highlights the impact of an offset gradient in the selection field, introducing a shift in +x and -y directions in the reconstructed image. Similarly, the bottom right panel depicts the effects of an inhomogeneous gradient, resulting in worse resolution for the points that are further away from the center as expected.

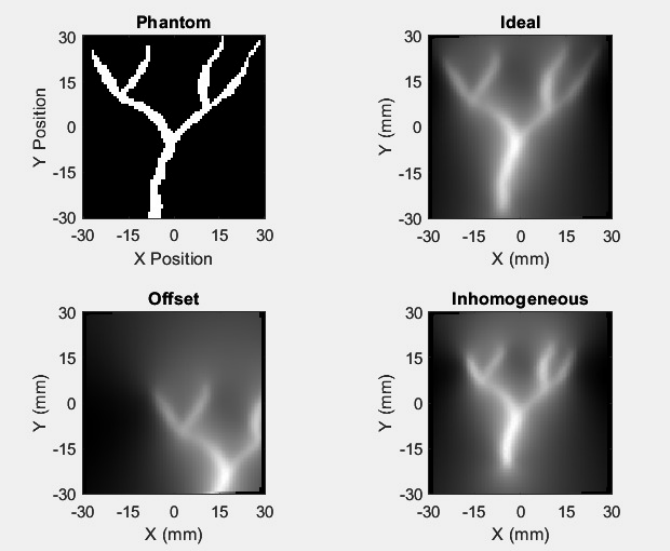

Fig. 10. Top Left: Vasculature Phantom. Top Right: X-space reconstructed MPI for the ideal selection field. Bottom Left: X-space reconstructed MPI image for the case of selection field with offset gradient. Bottom Right: X-space reconstructed MPI image for the case of selection field with inhomogeneous gradient

IV. Discussion

This project successfully showed that when the intended field of view goes beyond the linear region of the selection field, geometric warping artifacts appear in the x-space reconstructed images. These are caused by the non-linearity of the selection field and the focus and drive fields' inability to take this imperfection into account.

Comparing the PSFs to the ideal case, Table I shows that their FWHM values are nearly the same. This is due to the fact that resolution was measured using FWHM. Despite the fact that the image has both warping and resolution loss, the two effects balance each other out so that the PSFs' shapes remain mostly unaltered in the warped coordinate frame. However, the resolution loss is evident in the distorted image if the resolution is determined by the separability of two point sources which is realized by the Euclidean distance in this work. While going further away from the scanner's center, the warping has brought the point sources closer together, making it more difficult to distinguish them, even though the FWHM hasn't changed.

The unwarping process outlined in a prior study can be used to rectify the warping artifact enabling successful reconstruction of the image [3]. By varying the amplitudes of the focus and drive fields, the selection field's non-linearity may also be partially offset [3]. While this removes the warping artifact, it does not stop the resolution degradation that occurs farther out from the scanner's center. Another method for staying inside the linear region of the selection field without the use of a focus field is to move the phantom or subject through the bore of the scanner in a sliding-table fashion which was also previously proposed in another study for enlarging the FOV [8]. Resolution loss would also be lessened by that technique. However, in reality, it will only eliminate warping that is directed towards the bore. It would be possible to calculate the actual FFP trajectory and perform x-space reconstruction directly and simply if the selection field could be known [3]. However, it is inconvenient because the FFP trajectory would need to be recalculated each time the drive field's parameters such as frequency, amplitude, or even trajectory type change.

References

[1] B. Gleich and J. Weizenecker, "Tomographic imaging using the nonlinear response of magnetic particles," Nature, vol. 435, no. 7046, pp. 1214–1217, 2005. [Online]. Available: https://doi.org/10.1038/nature03808

[2] L. Zhang, J. Li, J. Du, G. Fang, D. Zhang, and Z. Tang, "Current reconstruction approaches of magnetic particle imaging: A review," Journal of Magnetism and Magnetic Materials, vol. 569, pp. 170440, 2024. [Online]. Available: https://doi.org/10.1016/j.jmmm.2023.170440

[3] E. Yagiz, A. R. Cagil, and E. U. Saritas, "Non-ideal Selection Field Induced Artifacts in X-Space MPI," International Journal on Magnetic Particle Imaging, vol. 6, no. 2, p. 2006001, 2020, doi: 10.18416/I-JMPI.2020.2006001.

[4] P. W. Goodwill and S. M. Conolly, "The X-Space Formulation of the Magnetic Particle Imaging Process: 1-D Signal, Resolution, Bandwidth, SNR, SAR, and Magnetostimulation," IEEE Transactions on Medical Imaging, vol. 29, no. 11, pp. 1851–1859, Nov. 2010. [Online]. Available: https://doi.org/10.1109/TMI.2010.2052284

[5] J. Rahmer, J. Weizenecker, B. Gleich, and J. Borgert, "Analysis of a 3-D System Function Measured for Magnetic Particle Imaging," IEEE Transactions on Medical Imaging, vol. 31, no. 6, pp. 1289–1299, June 2012. [Online]. Available: https://doi.org/10.1109/TMI.2012.2188639

[6] OS-MPI, Educational Simulations for Magnetic Particle Imaging. GitHub repository. [Online]. Available: https://github.com/OS-MPI/Educational_Simulations. [Accessed: Jan. 2, 2025].

[7] P. W. Goodwill and S. M. Conolly. Multidimensional X-Space Mag netic Particle Imaging. IEEE Transactions on Medical Imaging, 30(9):1581–1590, 2011, doi:10.1109/TMI.2011.2125982.

[8] P. Szwargulski, N. Gdaniec, M. Graeser, M. Möddel, F. Griese, K. M. Krishnan, T. M. Buzug, and T. Knopp. Moving table magnetic particle imaging: a stepwise approach preserving high spatio temporal resolution. Journal of Medical Imaging, 5(4):1, 2018, doi:10.1117/1.JMI.5.4.046002.

V. Appendix

A. MPI Main Code

```
ImageIn = zeros(60, 60);
ImageIn(2:3,   2:3)   = 1;ImageIn(2:3,   15:16) = 1;ImageIn
     (2:3,   30:31) = 1;ImageIn(2:3,   45:46) = 1;ImageIn
     (2:3,   58:59) = 1;
ImageIn(15:16, 2:3)   = 1;ImageIn(15:16, 15:16) = 1;ImageIn
     (15:16, 30:31) = 1;ImageIn(15:16, 45:46) = 1;ImageIn
     (15:16, 58:59) = 1;
ImageIn(30:31, 2:3)   = 1;ImageIn(30:31, 15:16) = 1;ImageIn
     (30:31, 30:31) = 1;ImageIn(30:31, 45:46) = 1;ImageIn
     (30:31, 58:59) = 1;
ImageIn(45:46, 2:3)   = 1;ImageIn(45:46, 15:16) = 1;ImageIn
     (45:46, 30:31) = 1;ImageIn(45:46, 45:46) = 1;ImageIn
     (45:46, 58:59) = 1;
ImageIn(58:59, 2:3)   = 1;ImageIn(58:59, 15:16) = 1;ImageIn
     (58:59, 30:31) = 1;ImageIn(58:59, 45:46) = 1;ImageIn
     (58:59, 58:59) = 1;

ImageOut_1 = X_Space(ImageIn, 1);
ImageOut_2 = X_Space(ImageIn, 2);
ImageOut_3 = X_Space(ImageIn, 3);

[error1, error2, error3] = CalculateError(ImageOut_1,
     ImageOut_2);

error1 = error1 / 50;
error2 = error2 / 18;
error3 = error3 / 2;
```

```
[error4, error5, error6] = CalculateError(ImageOut_1,
    ImageOut_3);

error4 = error4 / 50;
error5 = error5 / 18;

GenerateFigures(ImageIn, ImageOut_1, ImageOut_2, ImageOut_3
    , error1, error2, error3, error4, error5, error6);

GenerateFieldFigures()

MPI_Vessel()
```

B. X-Space Reconstruction Function

```
function [ImageOut] = X_Space(ImageIn, s)
    fDriveX = 25.5e3;
    Fs= 2e6;
    step = 0.0005;
    x = -.03:step:.03-step;

    DrivePeriods = length(x)*1;
    y = flipud(x(:));
    [X,Y] = meshgrid(x,y);
    ConcentrationMap = imresize(ImageIn,size(X));
    ConcentrationMap = cast(ConcentrationMap,'double');

    if s == 1
        Grad = 3;
        BfflFunc = @(X) Grad.*X;
        B_FFP_X_TMP = BfflFunc(X);
        B_FFP_Y_TMP = BfflFunc(Y);
    elseif s == 2
        Grad = 3;
        BfflFunc = @(X) Grad.*X;
        B_FFP_X_TMP = BfflFunc(X - 0.009);
        B_FFP_Y_TMP = BfflFunc(Y + 0.012);
    elseif s == 3
        Grad = 3;
        BfflFunc = @(X) Grad.*X;
        Inhomogeneity = exp(-10 .* sqrt(X .^ 2 + Y .^ 2));
        B_FFP_X_TMP = Inhomogeneity .* BfflFunc(X);
        B_FFP_Y_TMP = Inhomogeneity .* BfflFunc(Y);
    end


    t = 0:1/Fs:DrivePeriods*1/fDriveX;
    dt = t(2)-t(1);

    BDriveX(1,1,:) = sin(2*pi*fDriveX*t);
    BDriveX = repmat(BDriveX,length(y),length(x));

    BDriveY(1,1,:) = sawtooth(2*pi*1/(t(end)+dt)*t);
    BDriveY = repmat(BDriveY,length(y),length(x));

    B_FFP_X = repmat(B_FFP_X_TMP,1,1,length(t));
    BX = B_FFP_X+(1.1*Grad*x(end)*BDriveX); %This line
        defines the B field for all points in time in the X
         direction,
    %it is the summation of the FFP field plus the drive
        field. the 1.1 makes sure the FFP covers the full
        field of view
    B_FFP_Y = repmat(B_FFP_Y_TMP,1,1,length(t));
    BY = B_FFP_Y+(1.1*Grad*y(end)*BDriveY);

    clear B_FFP_Y_TMP B_FFP_X_TMP

    BMag = sqrt(BX.^2+BY.^2);
    FFP_Loc_Orig_X = X(BMag(:,:,1)==min(min(BMag(:,:,1))));
    FFP_Loc_Orig_Y = Y(BMag(:,:,1)==min(min(BMag(:,:,1))));

    Kb = 1.38e-23; %J/k
    Ms = 446e3; %A/m
    T = 300; %K
    Vol = 4/3*pi*(12.5e-9)^3;%m^3 (12.5nm core radius)
    BMag(BMag==0)=eps;

    M = ConcentrationMap(:,:,1).*(coth(Vol*Ms*BMag./(Kb*T))
          - (Kb*T)./(Vol*Ms*BMag));

    Mx_Demo = squeeze(sum(sum(M.*BX./(BMag),2),1)); %The
        imrotate is necessary to make sure the coordinates
        are aligned

    Vx_Demo = diff(BX(1,1,:)); %Taking the velocity at an
        arbitrary location
    Vx_Demo=Vx_Demo(:); %making a column

    Pos_X = FFP_Loc_Orig_X-cumtrapz(Vx_Demo/Grad);

    Vy_Demo = diff(BY(1,1,:));
    Vy_Demo = Vy_Demo(:);
    Pos_Y = FFP_Loc_Orig_Y-cumtrapz(Vy_Demo/Grad);

    SigX_Demo = diff(Mx_Demo(:));
    CroppingFactor = 0.3; %The cropping factor controls how
         much data is deleted due to being at low
        velocities. 0.3= 30% of data is deleted.
    HighVelocityIndex = abs(Vx_Demo)>(CroppingFactor*max(
        Vx_Demo(:))); %This is just a vector with ones
        corresponding to the times where the velocity is
        above 0.3* the max

    SigX_Demo(not(HighVelocityIndex)) = NaN;

    Pos_X = Pos_X(:);
    Pos_Y = Pos_Y(:);
    Vx_Demo = Vx_Demo(:);
    SigX_DemoDisp = SigX_Demo(HighVelocityIndex);
    SigX_DemoDisp = SigX_DemoDisp(:);

    Pos_X = Pos_X - min(Pos_X(:));
    Pos_X = Pos_X / max(Pos_X(:)) * (max(x) - min(x)) + min
        (x);

    Pos_Y = Pos_Y - min(Pos_Y(:));
    Pos_Y = Pos_Y / max(Pos_Y(:)) * (max(y) - min(y)) + min
        (y);

    HighVelocityIndex = abs(Vx_Demo)>(.3*max(Vx_Demo(:)));
    F = scatteredInterpolant(Pos_X(HighVelocityIndex),Pos_Y
        (HighVelocityIndex),SigX_DemoDisp./Vx_Demo(
        HighVelocityIndex), 'linear', 'none');

    Interp_Data = F(X,Y);
    ImageOut = Interp_Data;
end
```

C. FWHM Measurement Code

```
ImageIn = zeros(60, 60);
ImageIn(5:6, 5:6) = 1;    % near edge
% ImageIn(30:31, 30:31) = 1;  % center
ideal_img = X_Space2D_Demo(ImageIn, 1);
offset_img = X_Space2D_Demo(ImageIn, 2);
inhomo_img = X_Space2D_Demo(ImageIn, 3);

FWHM_ideal_x = findFWHM(ideal_img, 'x');
fprintf("FWHMx of ideal :\t %.2f mm \n",FWHM_ideal_x);
FWHM_ideal_y = findFWHM(ideal_img, 'y');
fprintf("FWHMy of ideal :\t %.2f mm \n",FWHM_ideal_y)

FWHM_offset_x = findFWHM(offset_img, 'x');
fprintf("FWHMx of offset :\t %.2f mm \n",FWHM_offset_x);
FWHM_offset_y = findFWHM(offset_img, 'y');
fprintf("FWHMy of offset :\t %.2f mm \n",FWHM_offset_y);

FWHM_inhomo_x = findFWHM(inhomo_img, 'x');
fprintf("FWHMx of Inhomogeneous :\t %.2f mm \n",
    FWHM_inhomo_x);
FWHM_inhomo_y = findFWHM(inhomo_img, 'y');
fprintf("FWHMy of Inhomogeneous :\t %.2f mm \n",
    FWHM_inhomo_y);

figure;
tiledlayout(2, 2);

nexttile;
imagesc(ImageIn);
title('Phantom');
xlabel('X␣(mm)');
ylabel('Y␣(mm)');
```

```
axis image;
colormap(gray);

xticks(linspace(1, size(ImageIn, 2), 5));
xticklabels(linspace(-30, 30, 5));
yticks(linspace(1, size(ImageIn, 1), 5));
yticklabels(linspace(30, -30, 5));

nexttile;

imagesc(ideal_img);
title('Ideal');
xlabel('X (mm)');
ylabel('Y (mm)');
axis image;
colormap(gray);

xticks(linspace(1, size(ideal_img, 2), 5));
xticklabels(linspace(-30, 30, 5));
yticks(linspace(1, size(ideal_img, 1), 5));
yticklabels(linspace(30, -30, 5));

nexttile;

imagesc(offset_img);
title('Offset');
xlabel('X (mm)');
ylabel('Y (mm)');
axis image;
colormap(gray);

xticks(linspace(1, size(offset_img, 2), 5));
xticklabels(linspace(-30, 30, 5));
yticks(linspace(1, size(offset_img, 1), 5));
yticklabels(linspace(30, -30, 5));


nexttile;

imagesc(inhomo_img);
title('Inhomogeneous');
xlabel('X (mm)');
ylabel('Y (mm)');
axis image;
colormap(gray);

xticks(linspace(1, size(inhomo_img, 2), 5));
xticklabels(linspace(-30, 30, 5));
yticks(linspace(1, size(inhomo_img, 1), 5));
yticklabels(linspace(30, -30, 5));



function FWHM = findFWHM(A, direction)
    % A: Input matrix (2D) or vector (1D) of field values
    % direction: Specify 'x' or 'y' to calculate FWHM in
        the respective direction

    % Find maximum value and its location
    [max_value, linear_index] = max(A(:));
    [row, col] = ind2sub(size(A), linear_index); % Get row
        and column indices of max value
    target_value = max_value / 2; % Half Maximum
    tolerance = 500; % Adjust tolerance based on
        requirements

    % Handle separate directions
    if strcmp(direction, 'x')
        % Extract the row corresponding to the max value
        row_data = A(row, :);
        indices = find(abs(row_data - target_value) <=
            tolerance);
        if ~isempty(indices)
            FWHM = abs((indices(end) - indices(1)) * (60 /
                240)); % FWHM in 'x' direction
        else
            fprintf('No values close to %.2f were found in
                the x direction.\n', target_value);
            FWHM = NaN;
        end
    elseif strcmp(direction, 'y')
        % Extract the column corresponding to the max value
        column_data = A(:, col);
        indices = find(abs(column_data - target_value) <=
            tolerance);
        if ~isempty(indices)
            FWHM = abs((indices(end) - indices(1)) * (60 /
                240)); % FWHM in 'y' direction
        else
            fprintf('No values close to %.2f were found in
                the y direction.\n', target_value);
            FWHM = NaN;
        end
    else
        error('Invalid direction specified. Use "x" or "y".
            ');
    end
end
```

D. Displacement Error Calculation Function

```
function [errorTotal, errorCenter9, errorCenter1] =
    CalculateError(ImageOut_1, ImageOut_2)
    blockSize = 24;
    numBlocks = 120 / blockSize;

    results_1 = [];
    results_2 = [];

    for rowBlock = 1:numBlocks
        for colBlock = 1:numBlocks
            rowStart = (rowBlock - 1) * blockSize + 1;
            rowEnd = rowBlock * blockSize;
            colStart = (colBlock - 1) * blockSize + 1;
            colEnd = colBlock * blockSize;

            subMatrix_1 = ImageOut_1(rowStart:rowEnd,
                colStart:colEnd);
            [~, linearIndex1] = max(subMatrix_1(:));
            [y1, x1] = ind2sub(size(subMatrix_1),
                linearIndex1);
            globalX1 = colStart + x1 - 1;
            globalY1 = rowStart + y1 - 1;
            results_1 = [results_1; globalX1, globalY1];

            subMatrix_2 = ImageOut_2(rowStart:rowEnd,
                colStart:colEnd);
            [~, linearIndex2] = max(subMatrix_2(:));
            [y2, x2] = ind2sub(size(subMatrix_2),
                linearIndex2);
            globalX2 = colStart + x2 - 1;
            globalY2 = rowStart + y2 - 1;
            results_2 = [results_2; globalX2, globalY2];
        end
    end


    errorTotal = 0;
    for i = 1:size(results_1, 1)
        dx = results_1(i, 1) - results_2(i, 1);
        dy = results_1(i, 2) - results_2(i, 2);
        errorTotal = errorTotal + sqrt(dx^2 + dy^2);
    end


    centerBlocks = [8, 9, 10, 13, 14, 15, 18, 19, 20];
    errorCenter9 = 0;
    for i = centerBlocks
        dx = results_1(i, 1) - results_2(i, 1);
        dy = results_1(i, 2) - results_2(i, 2);
        errorCenter9 = errorCenter9 + sqrt(dx^2 + dy^2);
    end

    centerBlockIndex = 13;
    dx = results_1(centerBlockIndex, 1) - results_2(
        centerBlockIndex, 1);
    dy = results_1(centerBlockIndex, 2) - results_2(
        centerBlockIndex, 2);
    errorCenter1 = sqrt(dx^2 + dy^2);
end
```

E. MPI Figure Generation Function

```
function GenerateFigures(ImageIn, ImageOut_1, ImageOut_2,
     ImageOut_3, error1, error2, error3, error4, error5,
     error6)
    figure;
    tiledlayout(1, 3);

    nexttile;
    imagesc(ImageIn);
    title('Phantom');
    xlabel('X Position (mm)');
    ylabel('Y Position (mm)');
    axis image;
    colormap(gray);
    xticks(linspace(1, size(ImageIn, 2), 5));
    xticklabels(linspace(-30, 30, 5));
    yticks(linspace(1, size(ImageIn, 1), 5));
    yticklabels(linspace(30, -30, 5));

    nexttile;
    imagesc(ImageOut_1);
    title('Ideal');
    xlabel('X Position (mm)');
    ylabel('Y Position (mm)');
    axis image;
    colormap(gray);
    xticks(linspace(1, size(X_Space(ImageIn, 1), 2), 5));
    xticklabels(linspace(-30, 30, 5));
    yticks(linspace(1, size(X_Space(ImageIn, 1), 1), 5));
    yticklabels(linspace(30, -30, 5));

    nexttile;
    imagesc(ImageOut_2);
    title('Offset');
    xlabel('X Position (mm)');
    ylabel('Y Position (mm)');
    axis image;
    colormap(gray);
    xticks(linspace(1, size(X_Space(ImageIn, 2), 2), 5));
    xticklabels(linspace(-30, 30, 5));
    yticks(linspace(1, size(X_Space(ImageIn, 2), 1), 5));
    yticklabels(linspace(30, -30, 5));


    figure;
    tiledlayout(1, 3);

    nexttile;
    imagesc(ImageIn);
    title('Phantom');
    xlabel('X Position (mm)');
    ylabel('Y Position (mm)');
    axis image;
    colormap(gray);
    xticks(linspace(1, size(ImageIn, 2), 5));
    xticklabels(linspace(-30, 30, 5));
    yticks(linspace(1, size(ImageIn, 1), 5));
    yticklabels(linspace(30, -30, 5));

    nexttile;
    imagesc(ImageOut_1);
    title('Ideal');
    xlabel('X Position (mm)');
    ylabel('Y Position (mm)');
    axis image;
    colormap(gray);
    xticks(linspace(1, size(X_Space(ImageIn, 1), 2), 5));
    xticklabels(linspace(-30, 30, 5));
    yticks(linspace(1, size(X_Space(ImageIn, 1), 1), 5));
    yticklabels(linspace(30, -30, 5));

    nexttile;
    imagesc(ImageOut_3);
    title('Inhomogeneous');
    xlabel('X Position (mm)');
    ylabel('Y Position (mm)');
    axis image;
    colormap(gray);
    xticks(linspace(1, size(X_Space(ImageIn, 3), 2), 5));
    xticklabels(linspace(-30, 30, 5));
    yticks(linspace(1, size(X_Space(ImageIn, 3), 1), 5));
    yticklabels(linspace(30, -30, 5));


    figure;
    tiledlayout(1, 3);

    blockSize = size(ImageOut_1, 1) / 5;
    centerBlock = 13;

    center3x3StartX = blockSize + 1;
    center3x3StartY = blockSize + 1;
    center3x3Width = 3 * blockSize;

    fullStartX = 1;
    fullStartY = 1;
    fullWidth = 5 * blockSize;

    nexttile;
    imagesc(ImageOut_1);
    title('Ideal');
    xlabel('X Position (mm)');
    ylabel('Y Position (mm)');
    axis image;
    colormap(gray);
    xticks(linspace(1, size(ImageOut_1, 2), 5));
    xticklabels(linspace(-30, 30, 5));
    yticks(linspace(1, size(ImageOut_1, 1), 5));
    yticklabels(linspace(30, -30, 5));

    hold on;
    rectangle('Position', [fullStartX, fullStartY,
         fullWidth, fullWidth], ...
              'EdgeColor', 'r', 'LineWidth', 2);

    rectangle('Position', [center3x3StartX, center3x3StartY
         , center3x3Width, center3x3Width], ...
              'EdgeColor', 'b', 'LineWidth', 2);

    [row, col] = ind2sub([5, 5], centerBlock);
    rectangle('Position', [(col-1)*blockSize + 1, (row-1)*
         blockSize + 1, blockSize, blockSize], ...
              'EdgeColor', 'g', 'LineWidth', 2);

    nexttile;
    imagesc(ImageOut_2);
    title('Offset');
    xlabel('X Position (mm)');
    ylabel('Y Position (mm)');
    axis image;
    colormap(gray);
    xticks(linspace(1, size(ImageOut_2, 2), 5));
    xticklabels(linspace(-30, 30, 5));
    yticks(linspace(1, size(ImageOut_2, 1), 5));
    yticklabels(linspace(30, -30, 5));

    hold on;
    rectangle('Position', [fullStartX, fullStartY,
         fullWidth, fullWidth], ...
              'EdgeColor', 'r', 'LineWidth', 2);

    rectangle('Position', [center3x3StartX, center3x3StartY
         , center3x3Width, center3x3Width], ...
              'EdgeColor', 'b', 'LineWidth', 2);

    rectangle('Position', [(col-1)*blockSize + 1, (row-1)*
         blockSize + 1, blockSize, blockSize], ...
              'EdgeColor', 'g', 'LineWidth', 2);

    nexttile;
    text(0.5, 0.7, sprintf('Red Square: %.2f mm', error1),
         'FontSize', 10, 'HorizontalAlignment', 'center');
    text(0.5, 0.5, sprintf('Blue Square: %.2f mm', error2),
          'FontSize', 10, 'HorizontalAlignment', 'center');
    text(0.5, 0.3, sprintf('Green Square: %.2f mm', error3)
         , 'FontSize', 10, 'HorizontalAlignment', 'center');
    axis off;
    title('Average Errors');



    figure;
    tiledlayout(1, 3);

    blockSize = size(ImageOut_1, 1) / 5;
```

```
    centerBlock = 13;

    center3x3StartX = blockSize + 1;
    center3x3StartY = blockSize + 1;
    center3x3Width = 3 * blockSize;

    fullStartX = 1;
    fullStartY = 1;
    fullWidth = 5 * blockSize;

    nexttile;
    imagesc(ImageOut_1);
    title('Ideal');
    xlabel('X Position (mm)');
    ylabel('Y Position (mm)');
    axis image;
    colormap(gray);
    xticks(linspace(1, size(ImageOut_1, 2), 5));
    xticklabels(linspace(-30, 30, 5));
    yticks(linspace(1, size(ImageOut_1, 1), 5));
    yticklabels(linspace(30, -30, 5));

    hold on;
    rectangle('Position', [fullStartX, fullStartY,
        fullWidth, fullWidth], ...
            'EdgeColor', 'r', 'LineWidth', 2);

    rectangle('Position', [center3x3StartX, center3x3StartY
        , center3x3Width, center3x3Width], ...
            'EdgeColor', 'b', 'LineWidth', 2);

    [row, col] = ind2sub([5, 5], centerBlock);
    rectangle('Position', [(col-1)*blockSize + 1, (row-1)*
        blockSize + 1, blockSize, blockSize], ...
            'EdgeColor', 'g', 'LineWidth', 2);

    nexttile;
    imagesc(ImageOut_3);
    title('Inhomogeneous');
    xlabel('X Position (mm)');
    ylabel('Y Position (mm)');
    axis image;
    colormap(gray);
    xticks(linspace(1, size(ImageOut_3, 2), 5));
    xticklabels(linspace(-30, 30, 5));
    yticks(linspace(1, size(ImageOut_3, 1), 5));
    yticklabels(linspace(30, -30, 5));

    hold on;
    rectangle('Position', [fullStartX, fullStartY,
        fullWidth, fullWidth], ...
            'EdgeColor', 'r', 'LineWidth', 2);

    rectangle('Position', [center3x3StartX, center3x3StartY
        , center3x3Width, center3x3Width], ...
            'EdgeColor', 'b', 'LineWidth', 2);

    rectangle('Position', [(col-1)*blockSize + 1, (row-1)*
        blockSize + 1, blockSize, blockSize], ...
            'EdgeColor', 'g', 'LineWidth', 2);

    nexttile;
    text(0.5, 0.7, sprintf('Red Square: %.2f mm', error4),
        'FontSize', 10, 'HorizontalAlignment', 'center');
    text(0.5, 0.5, sprintf('Blue Square: %.2f mm', error5),
         'FontSize', 10, 'HorizontalAlignment', 'center');
    text(0.5, 0.3, sprintf('Green Square: %.2f mm', error6)
        , 'FontSize', 10, 'HorizontalAlignment', 'center');
    axis off;
    title('Errors');
end
```

F. Selection Fields Figure Generation Function

```
function GenerateFieldFigures()
    step = 0.0005;
    x = -.03:step:.03-step;
    y = flipud(x(:));
    [X, Y] = meshgrid(x, y);

    Grad = 3;
    BfflFunc = @(X) Grad .* X;

    B1_FFP_X_TMP = BfflFunc(X);
    B1_FFP_Y_TMP = BfflFunc(Y);

    B2_FFP_X_TMP = BfflFunc(X - 0.009);
    B2_FFP_Y_TMP = BfflFunc(Y + 0.012);

    Inhomogeneity = exp(-10 .* sqrt(X .^ 2 + Y .^ 2));
    B3_FFP_X_TMP = Inhomogeneity .* BfflFunc(X);
    B3_FFP_Y_TMP = Inhomogeneity .* BfflFunc(Y);


    figure;

    subplot(3, 3, 1);
    imagesc(x * 1e3, y * 1e3, B1_FFP_X_TMP * 1e3);
    title('Bx, Ideal');
    xlabel('x (mm)');
    ylabel('y (mm)');
    colormap(jet);
    colorbar;
    clim([-100 100]);

    subplot(3, 3, 2);
    imagesc(x * 1e3, y * 1e3, B1_FFP_Y_TMP * 1e3);
    title('By, Ideal');
    xlabel('x (mm)');
    ylabel('y (mm)');
    colormap(jet);
    colorbar;
    clim([-100 100]);

    subplot(3, 3, 3);
    imagesc(x * 1e3, y * 1e3, 0);
    title('Bz, Ideal');
    xlabel('x (mm)');
    ylabel('y (mm)');
    colormap(jet);
    colorbar;
    clim([-100 100]);


    subplot(3, 3, 4);
    imagesc(x * 1e3, y * 1e3, B2_FFP_X_TMP * 1e3);
    title('Bx, Offset');
    xlabel('x (mm)');
    ylabel('y (mm)');
    colormap(jet);
    colorbar;
    clim([-100 100]);

    subplot(3, 3, 5);
    imagesc(x * 1e3, y * 1e3, B2_FFP_Y_TMP * 1e3);
    title('By, Offset');
    xlabel('x (mm)');
    ylabel('y (mm)');
    colormap(jet);
    colorbar;
    clim([-100 100]);

    subplot(3, 3, 6);
    imagesc(x * 1e3, y * 1e3, 0);
    title('Bz, Offset');
    xlabel('x (mm)');
    ylabel('y (mm)');
    colormap(jet);
    colorbar;
    clim([-100 100]);


    subplot(3, 3, 7);
    imagesc(x * 1e3, y * 1e3, B3_FFP_X_TMP * 1e3);
    title('Bx, Inhomogeneous');
    xlabel('x (mm)');
    ylabel('y (mm)');
    colormap(jet);
    colorbar;
    clim([-100 100]);

    subplot(3, 3, 8);
    imagesc(x * 1e3, y * 1e3, B3_FFP_Y_TMP * 1e3);
    title('By, Inhomogeneous');
    xlabel('x (mm)');
```

```
    ylabel('y (mm)');
    colormap(jet);
    colorbar;
    clim([-100 100]);

    subplot(3, 3, 9);
    imagesc(x * 1e3, y * 1e3, 0);
    title('Bz, Inhomogeneous');
    xlabel('x (mm)');
    ylabel('y (mm)');
    colormap(jet);
    colorbar;
    clim([-100 100]);
end
```

G. Vessel Phantom MPI Image Generation Function

```
function MPI_Vessel()
    ImageIn = double(rgb2gray(imread("vesselphantom2.png"))
        );

    figure;
    tiledlayout(2, 2);

    nexttile;
    imagesc(ImageIn);
    title('Phantom');
    xlabel('X Position');
    ylabel('Y Position');
    axis image;
    colormap(gray);
    xticks(linspace(1, size(ImageIn, 2), 5));
    xticklabels(linspace(-30, 30, 5));
    yticks(linspace(1, size(ImageIn, 1), 5));
    yticklabels(linspace(30, -30, 5));

    nexttile;
    ideal_img = X_Space(ImageIn, 1);
    imagesc(ideal_img);
    title('Ideal');
    xlabel('X (mm)');
    ylabel('Y (mm)')
    axis image;
    colormap(gray);
    xticks(linspace(1, size(ideal_img, 2), 5));
    xticklabels(linspace(-30, 30, 5));
    yticks(linspace(1, size(ideal_img, 1), 5));
    yticklabels(linspace(30, -30, 5));

    nexttile;
    offset_img = X_Space(ImageIn, 2);
    imagesc(offset_img);
    title('Offset');
    xlabel('X (mm)');
    ylabel('Y (mm)');
    axis image;
    colormap(gray);
    xticks(linspace(1, size(offset_img, 2), 5));
    xticklabels(linspace(-30, 30, 5));
    yticks(linspace(1, size(offset_img, 1), 5));
    yticklabels(linspace(30, -30, 5));


    nexttile;
    inhomo_img = X_Space(ImageIn, 3);
    imagesc(inhomo_img);
    title('Inhomogeneous');
    xlabel('X (mm)');
    ylabel('Y (mm)');
    axis image;
    colormap(gray);
    xticks(linspace(1, size(inhomo_img, 2), 5));
    xticklabels(linspace(-30, 30, 5));
    yticks(linspace(1, size(inhomo_img, 1), 5));
    yticklabels(linspace(30, -30, 5));
end
```