Applied Medical Image Processing Lecture Notes

1 Image Registration:

Image registration involves aligning different sets of data into one coordinate system. This is essential for various applications in medical imaging.

1.1 Applicatioins

1. **Identify Correspondence Between Two Different Images**
    - *Same Subject, Different Times*: The same subject is imaged at different times to observe changes over time.
    - *Same Subject, Same Time, Different Modalities*: The same subject is imaged simultaneously using different imaging modalities to gather diverse information.
    - *Different Subjects*: Images of different subjects are compared or aligned for various analyses.
2. **Overlay Images**
    - Overlaying images allows for direct comparison of information from different sources. This helps in integrating data for better analysis and interpretation.
3. **Match a Reference Image (Atlas) to Individual Subject**
    - Propagate labels for image segmentation.
    - Use deformation to study shape characteristics.

1.2 Factors Influencing Image Registration

For effective image mapping, several factors drive the registration process:

1. **Feature Space**

- *Extrinsic Landmarks*: External markers that are visible in the images.
- *Intrinsic Features*: Landmarks on anatomy such as curves, surfaces, and voxel intensities that are inherent to the subject.

2. **Similarity Criterion**

   - The similarity criterion depends on the feature space being used. It measures how similar the images are after transformation. As an example, we can consider:

$$S(x, I_0, I_1, T) = \sum_x \|I_1(x) - T[I_0(x)]\|^2$$

   Here, $S(x, I_0, I_1, T)$ represents the similarity measure between images I_0 and I_1 after applying the transformation T. The summation is over all pixels x, and $\|\cdot\|$ denotes the norm, typically the Euclidean norm.

3. **Transformation**

   - Defines how we want to establish correspondence between the images. The transformation could be rigid, affine, or non-rigid depending on the application.

   Transformations are crucial in image registration, allowing us to map points from one space to another, often to align images for analysis.

   3.1. **Types of Transformations**

   Two common types of transformations are mapping 2D image to another 2D image or 3D image to another 3D image.

   - $T : \mathbb{R}^2 \to \mathbb{R}^2$
   - $T : \mathbb{R}^3 \to \mathbb{R}^3$

   Given points (x, y, z) located on a discrete lattice $\mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$:

$$(x', y', z') = T(x, y, z)$$

   The transformed points (x', y', z') may not fall into a discrete raster point, requiring interpolation.

3.2. Transformation Components

For 3D transformations, the new coordinates (x', y', z') are given by:

$$\begin{aligned} x' &= T_x(x, y, z) \\ y' &= T_y(x, y, z) \\ z' &= T_z(x, y, z) \end{aligned}$$

In matrix form, this can be written as:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} dx \\ dy \\ dz \end{pmatrix}$$

3.3. 2D Rotation

For 2D rotation, the transformation equations are:

$$\begin{aligned} T_x : x' &= x \cos\alpha - y \sin\alpha \\ T_y : y' &= x \sin\alpha + y \cos\alpha \end{aligned}$$

3.4. 3D Rotation Matrices

Rotation matrices for 3D transformations are defined as follows:
- **Rotation about the Z-axis**:

$$R_z = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- **Rotation about the X-axis**:

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta \\ 0 & \sin\beta & \cos\beta \end{pmatrix}$$

- **Rotation about the Y-axis**:

$$R_y = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}$$

3.5. Composite Rotation

The composite rotation $\mathbf{X}'$ is obtained by applying the rotation matrices in sequence:

$$\mathbf{X}' = R_x R_y R_z \mathbf{X}$$

Note that the order in sequence is important, therefore previous sequence results in a different transformation from this one:

$$\mathbf{X}' = R_z R_y R_x \mathbf{X}$$

3.6. Rigid Transformation

Rigid transformations involve rotation and translation but preserve the length of lines, ensuring that the shape and size of objects do not change.

3.7. Affine Transformations in Image Registration

Affine transformations are used in image registration to map points from one space to another while allowing the length and orientation of vectors to change.

3.7.1. Scaling Transformation

The scaling transformation matrix T_s is defined as:

$$T_s = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{pmatrix}$$

3.7.2. Shearing Transformation

Shearing transformation parallel to the x-axis T_{sh}:

$$T_{sh} = \begin{pmatrix} 1 & b_x & 0 \\ 0 & 1 & 0 \\ 0 & d_1 & 1 \end{pmatrix}$$

Shearing transformation parallel to the y-axis T_{sh}:

$$T_{sh} = \begin{pmatrix} 1 & 0 & 0 \\ b_y & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

General shearing transformation:

$$\begin{pmatrix} 1 & a & b \\ c & 1 & d \\ e & f & 1 \end{pmatrix}$$

3.7.3. **Combined Affine Transformation**

An affine transformation can include translation, rotation, shearing, and scaling. The general form of an affine transformation matrix T is:

$$T = \begin{pmatrix} a_{11} & a_{12} & a_{13} & dx \\ a_{21} & a_{22} & a_{23} & dy \\ a_{31} & a_{32} & a_{33} & dz \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The transformation applied to a point (x, y, z) results in new coordinates (x', y', z'):

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & dx \\ a_{21} & a_{22} & a_{23} & dy \\ a_{31} & a_{32} & a_{33} & dz \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

3.8. **Determinant and Inverse of the Transformation Matrix**

For affine transformations, the determinant of the matrix must be non-zero to ensure the transformation is invertible. This ensures that any point mapped from the source image to the target image can be uniquely mapped back, maintaining consistency in the registration process (see Fig. 1).The inverse of the transformation matrix A^{-1} can be calculated as long as $\det(A) \neq 0$:

$$A^{-1} \propto \frac{1}{\det(A)}$$

Affine and rigid transformations are global, meaning the same mapping is applied to all pixels in the image. However, in practice, local variations often require local mappings.

3.9. **Global vs. Local Transformations**

- **Global Transformations**: These apply the same mapping to all pixels.

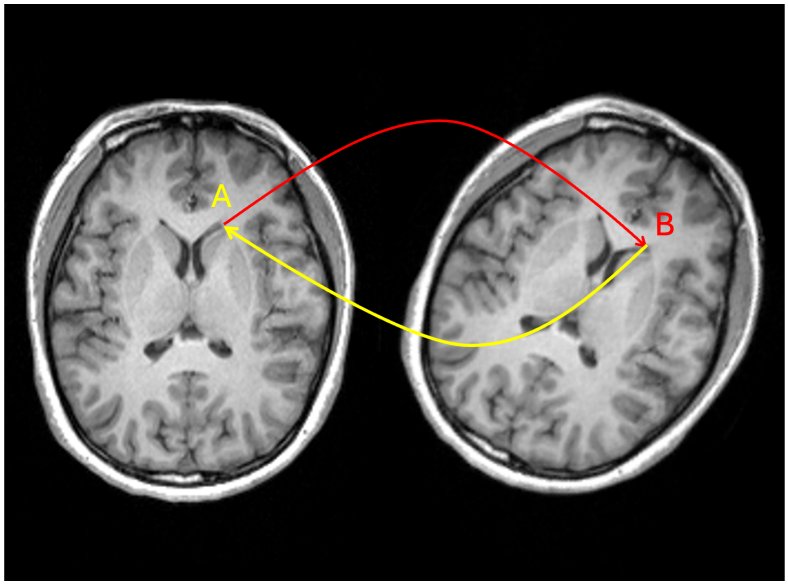

Figure 1: Illustration of the transformation and its inverse between two images of brain MRI

- **Local Transformations**: These account for local variations and apply different mappings to different parts of the image.

For a point $\mathbf{x}$:

$$T(\mathbf{x}) \rightarrow \begin{cases} x \rightarrow x + dx \\ y \rightarrow y + dy \\ z \rightarrow z + dz \end{cases}$$

3.10. Image Resampling

Image resampling involves interpolating image data from one coordinate system to another.

Let I_0 be the original image sampled on a discrete lattice. We want a new image I_1 which also has discretized coordinates. This can be represented as:

$$I_0(u, v) \rightarrow I_1(u', v')$$

where u, v and u', v' belong to the set of integer coordinates $\mathbb{Z}$.

3.10.1. Mapping Techniques

A. **Forward Mapping**: Maps points from the source image to the target image (Fig. 2).

$$\text{Source} \rightarrow \text{Target}$$

B. **Backward Mapping**: Maps points from the target image to the source image (Fig. 2).

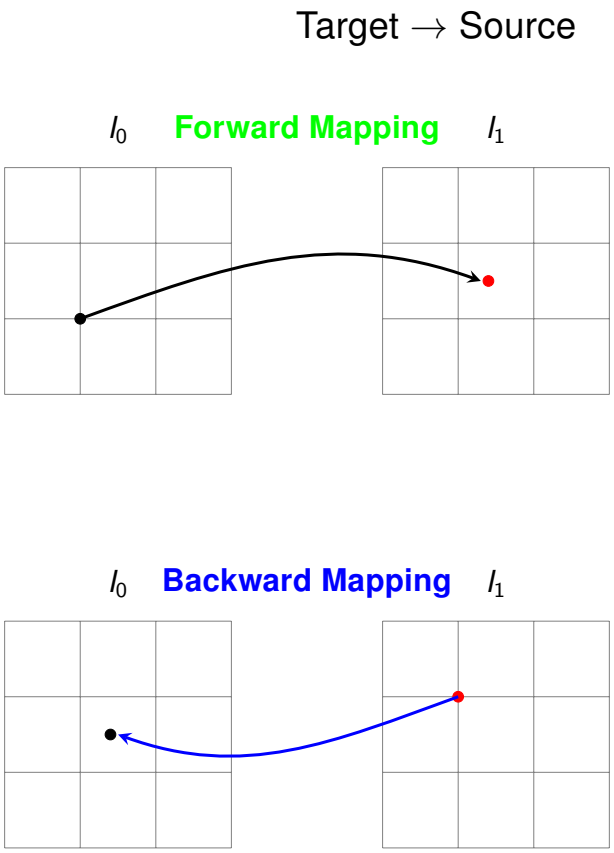

Figure 2: Illustration of Forward and Backward Mapping

In figure 2, forward mapping is shown on the top, where points from the source image I_0 are mapped to the target image I_1. Forward mapping may result in holes or gaps in the target image because some pixels in the target image might not be filled if the transformation function skips certain areas. Backward mapping is shown on the bottom, where points from the target image I_1 are mapped back to the source image I_0. Interpolation is necessary when mapped points do not fall on discrete grid points.

2 Intensity-based Image Registration

2.1 Description

Intensity-based image registration is a process used to align two images, I_0 and I_1, by optimizing a transformation T that minimizes the difference between the intensity of images. The goal is to find the optimal transformation T^* that minimizes the energy function E.

2.2 Energy Function

The energy function E is defined as the sum of the squared differences between the pixel intensities of the two images after applying the transformation T:

$$E = \sum_{\vec{x} \in \Omega} \left(I_0(\vec{x}) - I_1(T(\vec{x})) \right)^2$$

The optimal transformation T^* is found by minimizing the energy function:

$$T^* = \arg\min_T E$$

2.3 Affine and Polynomial Transformations

We can define different transformation that maps pixels in one image to another one. One example is Affine transformation. Affine transformation for a 3D point (x, y, z) can be expressed as:

$$\begin{cases} x' = K_{x0} + K_{x1}x + K_{x2}y + K_{x3}z \\ y' = K_{y0} + K_{y1}x + K_{y2}y + K_{y3}z \\ z' = K_{z0} + K_{z1}x + K_{z2}y + K_{z3}z \end{cases}$$

A more complex polynomial transformation for x', y', z' can be expressed as:

$$x' = K_{x0} + K_{x1}x + K_{x2}y + K_{x3}z + K_{x4}x^2 + K_{x5}xy + K_{x6}y^2 + K_{x7}xz + K_{x8}yz + K_{x9}z^2$$
$$y' = K_{y0} + K_{y1}x + K_{y2}y + K_{y3}z + K_{y4}x^2 + K_{y5}xy + K_{y6}y^2 + K_{y7}xz + K_{y8}yz + K_{y9}z^2$$
$$z' = K_{z0} + K_{z1}x + K_{z2}y + K_{z3}z + K_{z4}x^2 + K_{z5}xy + K_{z6}y^2 + K_{z7}xz + K_{z8}yz + K_{z9}z^2$$

In this case we have *30 parameters* ($K_{x/y/z,n}$)to describe the transformation that need to be optimized. Typically, we don't have an analytical solution and instead we need to use an iterative approach.

3 Optimization

3.1 Chi-Squared Function

The energy function to be optimized can be expressed generally as chi-squared function. The chi-squared function $\chi^2(\vec{a})$ measures the difference between observed data points y_i and model predictions $y(x_i; \vec{a})$:

$$\chi^2(\vec{a}) = \sum_{i=1}^{N} \frac{[y_i - y(x_i; \vec{a})]^2}{\sigma_i^2}$$

Here $\vec{a}$ represents the set of parameters (transformation coefficients in our model) to be optimizes in order to minimize the energy function (Chi-Squared Function).

3.2 Gradient Descent Approach

Gradient descent is used to iteratively update the parameters $\vec{a}$ to minimize the chi-squared function:

$$\vec{a}_{\text{next}} = \vec{a}_{\text{current}} - \lambda \nabla \chi^2(\vec{a}_{\text{current}})$$

This algorithm iteratively moves towards the minimum of energy function by taking steps proportional to the negative of the gradient (first derivative) of the

function at the current point. Here, λ is the learning rate that needs to be predefined for the optimization process. Here, λ is the learning rate that needs to be predefined for the optimization process. Small values of λ prolong the optimization process, while large values increase the chance of overshooting the optimum solution. Finally, $\nabla\chi^2(\vec{a}_{\text{current}})$ is the gradient of the function at the current point.

3.3 Gauss-Newton Optimization

In Gradient Descent algorithm convergence can be slow, especially if the learning rate is not chosen properly. Alternatively, we can use Gauss-Newton algorithm which is a specialized optimization technique used for solving non-linear least squares problems (Chi-squared function described above). It approximates the Hessian matrix (second derivative) of the function. This leads to faster convergence for problems that fit the assumptions.

Using Taylor expansion and ignoring higher-order terms:

$$\nabla\chi^2(\vec{a}) \approx \underbrace{\nabla\chi^2(\vec{a}_0)}_{\text{Jacobian}} + \delta\vec{a}^T \underbrace{\nabla^2\chi^2(\vec{a}_0)}_{\text{Hessian}} = 0$$

Solving for $\delta\vec{a}$ (updates steps for the parameters):

$$\begin{aligned} \delta\vec{a}^T\nabla^2\chi^2(\vec{a}_0) &= -\nabla\chi^2(\vec{a}_0) \\ \delta\vec{a} &= \nabla^2\chi^2(\vec{a}_0)^{-1}(-\nabla\chi^2(\vec{a}_0)) \end{aligned}$$

This indicates that to calculate the update step, Jacobian (first derivative) and Hessian (second derivative) of Chi-squared function needs to be calculated.

The derivative of χ^2 with respect to parameter a_k is:

$$\frac{\partial\chi^2}{\partial a_k} = -2\sum_{i=1}^{N}\frac{[y_i - y(x_i;\vec{a})]}{\sigma_i^2}\frac{\partial y(x_i;\vec{a})}{\partial a_k}$$

for $k = 1, 2, \ldots, M$.

The second derivative of the chi-squared function with respect to parameters a_k and a_l is given by:

$$\frac{\partial^2 \chi^2(\vec{a})}{\partial a_k \partial a_l} = 2 \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left[\frac{\partial y(x_i; \vec{a})}{\partial a_k} \frac{\partial y(x_i; \vec{a})}{\partial a_l} - (y_i - y(x_i; \vec{a})) \frac{\partial^2 y(x_i; \vec{a})}{\partial a_k \partial a_l} \right]$$

We denote:

$$\beta_k = -\frac{1}{2} \frac{\partial \chi^2}{\partial a_k}$$

and

$$\alpha_{kl} = \frac{1}{2} \frac{\partial^2 \chi^2(\vec{a})}{\partial a_k \partial a_l}$$

Then for the gradient descent algorithm, we could write $\delta a_k =$ constant β_k. For the Gauss Newton approach, we should solve:

$$\sum_{l=1}^{M} \alpha_{kl} \delta a_l = \beta_k$$

3.4 Marquardt-Levenberg Optimization Algorithm

The Marquardt-Levenberg (ML) algorithm is an iterative optimization technique that combines aspects of gradient descent and the Gauss-Newton method. It is particularly useful for nonlinear least squares problems.

Steps of the ML algorithm:

1. Compute $\chi^2(\vec{a})$
2. Pick a value for λ (e.g., $\lambda = 0.001$)

3. Solve for $\delta\vec{a}$ and evaluate $\chi^2(\vec{a}+\delta\vec{a})$

$$\delta\vec{a} : \sum_{l=1}^{M} \alpha'_{kl}\delta a_l = \beta_k$$

The update for the matrix α in the Marquardt-Levenberg algorithm is defined as follows:

$$\begin{cases} \alpha'_{kk} = \alpha_{kk}(1+\lambda) \\ \alpha'_{kl} = \alpha_{kl} \quad k \neq l \end{cases}$$

4. Conditional Update of Parameters

   (a) If $\chi^2(\vec{a}+\delta\vec{a}) > \chi^2(\vec{a})$, increase λ by a factor of 10 and go back to step 3.

   (b) If $\chi^2(\vec{a}+\delta\vec{a}) < \chi^2(\vec{a})$, decrease λ by a factor of 10, update $\vec{a} \rightarrow \vec{a}+\delta\vec{a}$, and go back to step 3.

Note: Hessian matrices are difficult to calculate and can be unstable.

3.5 Application to Image Registration

In the context of image registration the Chi-squared or cost function E is defined as:

$$E = \chi^2 = \sum_{x\in\Omega} \left(I_0(\vec{x}) - I_1(T(\vec{x}))\right)^2$$

The gradient of χ^2 with respect to parameter a_k is:

$$\frac{\partial\chi^2}{\partial a_k} = \frac{\partial E}{\partial a_k}$$

$$= 2\sum_{x\in\Omega} \left(I_0(\vec{x}) - I_1(T(\vec{x}))\right) \cdot \frac{\partial I_1(T(\vec{x}))}{\partial T(\vec{x})} \cdot \frac{\partial T(\vec{x})}{\partial a_k}$$

where $T(\vec{x}) = x' = [X', Y', Z']$.

The transformation equation for X' is given by:

$$X' = k_{x0} + k_{x1}x + k_{x2}y + k_{x3}z + k_{x4}x^2 + k_{x5}xy + k_{x6}y^2 + k_{x7}xz + k_{x8}yz + k_{x9}z^2$$

The partial derivatives of the transformation function relative to the parameters K_{xn} are:

$$\frac{\partial T(\vec{x})}{\partial a_k} = \frac{\partial X'}{\partial a_k} = \begin{cases} \frac{\partial x'}{\partial a_0} = 1 \\ \frac{\partial x'}{\partial a_1} = x \\ \frac{\partial x'}{\partial a_2} = y \\ \frac{\partial x'}{\partial a_3} = z \\ \vdots \\ \frac{\partial x'}{\partial a_9} = z^2 \end{cases}$$

To calculate $\frac{\partial I_1(\vec{x'})}{\partial \vec{x'}}$, identify the 8 nearest voxels to the point (X', Y', Z') using rounding down ($\swarrow$) or up ($\nearrow$) to the nearest voxel. This is called trilinear interpolation. The interpolated intensity I_{interp} at point (X', Y', Z') is given by:

$$\begin{aligned} I_{\text{interp}(X',Y',Z'} = {} & I_{X'_{\swarrow} Y'_{\swarrow} Z_{\swarrow}} \cdot (X'_{\nearrow} - X') \cdot (Y'_{\nearrow} - Y') \cdot (Z'_{\nearrow} - Z') \\ & + I_{X'_{\nearrow} Y'_{\swarrow} Z_{\swarrow}} \cdot (X' - X'_{\swarrow}) \cdot (Y'_{\nearrow} - Y') \cdot (Z'_{\nearrow} - Z') \\ & + I_{X'_{\swarrow} Y'_{\nearrow} Z_{\swarrow}} \cdot (X'_{\nearrow} - X') \cdot (Y' - Y'_{\swarrow}) \cdot (Z'_{\nearrow} - Z') \\ & + I_{X'_{\nearrow} Y'_{\nearrow} Z_{\swarrow}} \cdot (X' - X'_{\swarrow}) \cdot (Y' - Y'_{\swarrow}) \cdot (Z'_{\nearrow} - Z') \\ & + I_{X'_{\swarrow} Y'_{\swarrow} Z_{\nearrow}} \cdot (X'_{\nearrow} - X') \cdot (Y'_{\nearrow} - Y') \cdot (Z' - Z'_{\swarrow}) \\ & + I_{X'_{\nearrow} Y'_{\swarrow} Z_{\nearrow}} \cdot (X' - X'_{\swarrow}) \cdot (Y'_{\nearrow} - Y') \cdot (Z' - Z'_{\swarrow}) \\ & + I_{X'_{\swarrow} Y'_{\nearrow} Z_{\nearrow}} \cdot (X'_{\nearrow} - X') \cdot (Y' - Y'_{\swarrow}) \cdot (Z' - Z'_{\swarrow}) \\ & + I_{X'_{\nearrow} Y'_{\nearrow} Z_{\nearrow}} \cdot (X' - X'_{\swarrow}) \cdot (Y' - Y'_{\swarrow}) \cdot (Z' - Z'_{\swarrow}) \end{aligned}$$

where $X'_{\nearrow \text{ or } \swarrow}$, $Y'_{\nearrow \text{ or } \swarrow}$, and $Z'_{\nearrow \text{ or } \swarrow}$ are the coordinates of the nearest voxels.

Based on this definition, the partial derivative of the interpolated intensity I_{interp} with respect to X' is:

$$
\begin{aligned}
\frac{\partial I_{\text{interp}}}{\partial X'} = & \\
& - I_{X'_{\swarrow} Y'_{\swarrow} Z_{\swarrow}} \cdot (Y'_{\nearrow} - y') \cdot (Z'_{\nearrow} - Z') \\
& + I_{X'_{\nearrow} Y'_{\swarrow} Z_{\swarrow}} \cdot (Y'_{\nearrow} - Y') \cdot (Z'_{\nearrow} - Z') \\
& - I_{X'_{\swarrow} Y'_{\nearrow} Z_{\swarrow}} \cdot (y' - y'_{\swarrow}) \cdot (z'_{\nearrow} - Z') \\
& \vdots \\
& + I_{X'_{\nearrow} Y'_{\nearrow} Z_{\nearrow}} \cdot (Y' - Y'_{\swarrow}) \cdot (Z' - Z'_{\swarrow})
\end{aligned}
$$

Similarly, we can compute the partial derivatives with respect to y' and z':

$$\frac{\partial I_{\text{interp}}}{\partial y'} = \dots$$

$$\frac{\partial I_{\text{interp}}}{\partial z'} = \dots$$

The second-order partial derivatives can be calculated as well using this approach:

$$\frac{\partial^2 I_{\text{interp}}}{\partial x' \partial y'} = \dots$$

$$\frac{\partial^2 I_{\text{interp}}}{\partial x' \partial z'} = \dots$$

$$\frac{\partial^2 I_{\text{interp}}}{\partial y' \partial z'} = \dots$$

Once first and second derivatives were calculated, Marquardt-Levenberg optimization Algorithm will be used to find the optimum transformation parameters to match one image to another.

Bibliography

[1] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed., Cambridge University Press, 1992.

[2] Roger P. Woods, Subhadip G. Mazziotta, and John C. Mazziotta, *Automated Image Registration: I. General Methods and Intrasubject, Intramodality Validation*, Journal of Computer Assisted Tomography, vol. 16, no. 4, pp. 620-633, 1992.

[3] Woods, R. P., Grafton, S. T., Holmes, C. J., Cherry, S. R., and Mazziotta, J. C. (1993). Automated Image Registration: II. Intersubject Validation of Linear and Nonlinear Models. Journal of Computer Assisted Tomography, 17(4), 536-546.