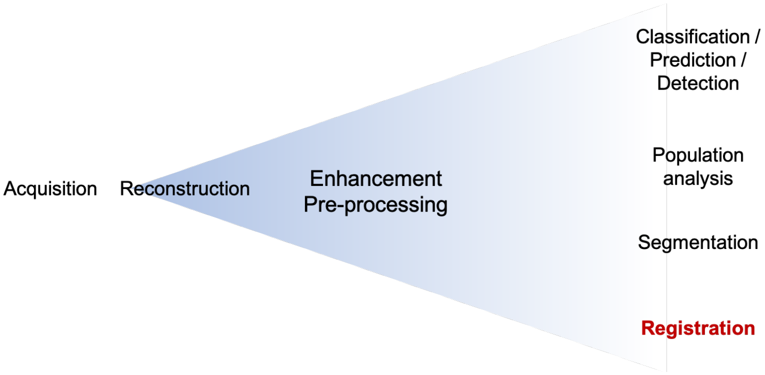
Transformation Models

Ender Konukoglu

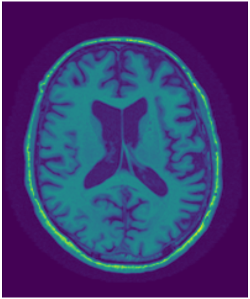
ETH Zürich

March 10, 2020

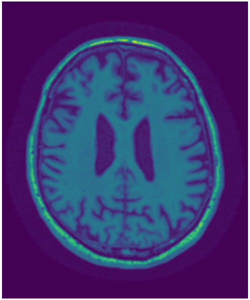
Registration is an essential task in medical image analysis



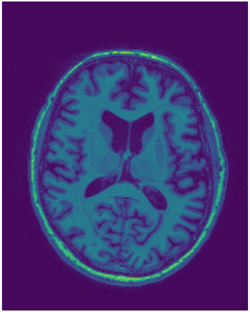
What is image registration and why do we need it?



Time point 1



Time point 2



Time point 3

A common problem

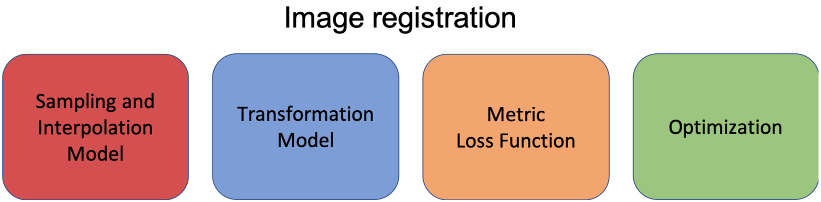
These are three images taken from the same individual 6 months apart. Images show the same cross-section (90th slide) from three different volumetric images. Notice that they do not show the same anatomy. That is because they are not aligned. Image registration aligns these images through spatial transformations and allows comparisons.

Useful for many different applications

Image Registration
Spatial normalization

- Aligning images of different modalities
- Longitudinal analysis
- Atlas-based segmentation
- Population Analysis
- Image analysis for interventions: alignment of pre- and intra-intervention images
- Any other analysis that requires aligning different images

Four main components

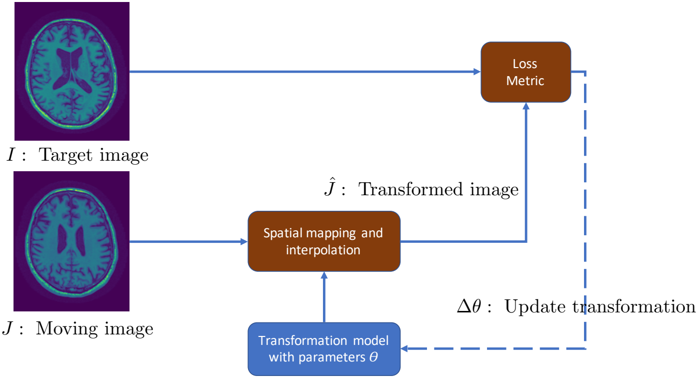


Any registration algorithm is composed of four components. Today we will study the "Transformation Models".

Outline

- Sampling model - How to apply a transformation
- Interpolation models
- Linear transformation models
- Non-linear deformable transformation models

Overview on a registration algorithm



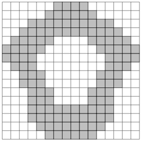
$$\theta^* = \arg_\theta \min \mathcal{L}(I, T_\theta \circ J)$$

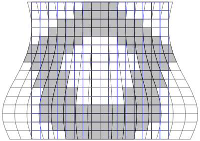
Main question of today

How do we parameterize T_θ with $T_\theta \circ J \triangleq J(T(x))$?

Previously we have seen an example

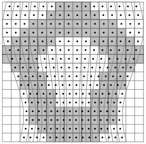
We already used a parameterized transformation model but a very specific one.

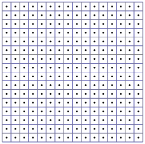

Input grid and Moving image


Output grid and Transformed grid

Forward transformation

$$T(x) = \{x_1 / [0.15 \sin(2.5 x_2) + 0.85], x_2\} = \{x_1', x_2'\} = x'$$


Input grid / Moving image


Output grid / Output image

Backward transformation

$$T^{-1}(x') = \{x_1' [0.15 \sin(2.5 x_2) + 0.85], x_2'\} = \{x_1, x_2\} = x$$

Section 2

Transformation models

Outline

- Sampling model - How to apply a transformation
- Interpolation models
- Linear transformation models
  - Rigid transformations
  - Similarity transformations
  - Affine transformations
- Non-linear deformable transformation models
  - Pixel/Voxel-wise physical models
  - Kernel-based interpolation models

Subsection 1

Linear Transformations

Linear mappings as transformations

The general form of the transformation is a matrix-vector product:

$$x' = T(x) = \mathbf{A}x + t,$$

Linear mappings as transformations

The general form of the transformation is a matrix-vector product:

$$x' = T(x) = \mathbf{A}x + t,$$

where the transformation is defined by the matrix $\mathbf{A}$ and the translation vector t. The dimension of this matrix and the translation vector depend on the dimension of the input and output spaces. Transformations between two two-dimensional spaces

$$x \in \mathbb{R}^2,\ x' \in \mathbb{R}^2 \rightarrow \mathbf{A} \in \mathbb{R}^{2\times 2},\ t \in \mathbb{R}^2$$

Linear mappings as transformations

The general form of the transformation is a matrix-vector product:

$$x' = T(x) = \mathbf{A}x + t,$$

where the transformation is defined by the matrix $\mathbf{A}$ and the translation vector t. The dimension of this matrix and the translation vector depend on the dimension of the input and output spaces. Transformations between two two-dimensional spaces

$$x \in \mathbb{R}^2,\ x' \in \mathbb{R}^2 \rightarrow \mathbf{A} \in \mathbb{R}^{2\times 2},\ t \in \mathbb{R}^2$$

Transformations between two three-dimensional spaces

$$x \in \mathbb{R}^3,\ x' \in \mathbb{R}^3 \rightarrow \mathbf{A} \in \mathbb{R}^{3\times 3},\ t \in \mathbb{R}^3$$

Linear mappings as transformations

The general form of the transformation is a matrix-vector product:

$$x' = T(x) = \mathbf{A}x + t,$$

where the transformation is defined by the matrix $\mathbf{A}$ and the translation vector t. The dimension of this matrix and the translation vector depend on the dimension of the input and output spaces. Transformations between two two-dimensional spaces

$$x \in \mathbb{R}^2,\ x' \in \mathbb{R}^2 \to \mathbf{A} \in \mathbb{R}^{2\times 2},\ t \in \mathbb{R}^2$$

Transformations between two three-dimensional spaces

$$x \in \mathbb{R}^3,\ x' \in \mathbb{R}^3 \to \mathbf{A} \in \mathbb{R}^{3\times 3},\ t \in \mathbb{R}^3$$

- These are the most commonly used cases.
- One can also consider transformation of a 3D space to 2D and a 2D space to 3D.
- Transformations between 2D images and 3D volumes, e.g. 2D ultrasound and 3D CT for intervention, are also studied.
- Our focus will be on the commonly used cases.

Parameterization in the most generic case

In two-dimensions

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \; t = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

Four + Two = Six free parameters

In three-dimensions

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, \; t = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

Nine + Three = 12 free parameters

Parameterization in the most generic case

In two-dimensions

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \; t = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

Four + Two = Six free parameters

In three-dimensions

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, \; t = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

Nine + Three = 12 free parameters

- Linear transformations has small number of parameters compared to non-linear transformations.

Parameterization in the most generic case

In two-dimensions

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \ t = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

Four + Two = Six free parameters

In three-dimensions

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, \ t = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

Nine + Three = 12 free parameters

- Linear transformations has small number of parameters compared to non-linear transformations.
- The same parameters apply to the entire image.

Parameterization in the most generic case

In two-dimensions

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \; t = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

Four + Two = Six free parameters

In three-dimensions

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, \; t = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

Nine + Three = 12 free parameters

- Linear transformations has small number of parameters compared to non-linear transformations.
- The same parameters apply to the entire image.
- Therefore, the number of parameters is *independent* of the number of pixels / voxels.

$$T(x) = \mathbf{A}x + t, \; \forall x$$

Parameterization in the most generic case

In two-dimensions

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \ t = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

Four + Two = Six free parameters

In three-dimensions

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, \ t = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

Nine + Three = 12 free parameters

- Linear transformations has small number of parameters compared to non-linear transformations.
- The same parameters apply to the entire image.
- Therefore, the number of parameters is *independent* of the number of pixels / voxels.

$$T(x) = \mathbf{A}x + t, \ \forall x$$

- If the matrix **A** is invertible then the inverse transformation is given as

$$T^{-1}(x') = \mathbf{A}^{-1}x' + t', \ t' = -\mathbf{A}^{-1}t, \ \forall x'$$

Homogeneous coordinates

Alternative representation of the linear transformation can be written with the more compact homogeneous coordinates. Instead of

$$T(x) = \mathbf{A}x + t, \ \forall x$$

Homogeneous coordinates

Alternative representation of the linear transformation can be written with the more compact homogeneous coordinates. Instead of

$$T(x) = \mathbf{A}x + t, \ \forall x$$

One can use the homogeneous coordinates

$$\tilde{T}(x) = \tilde{\mathbf{A}}\tilde{x}, \ \tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & t \\ \mathbf{0} & 1 \end{bmatrix}, \ \tilde{x} = \begin{bmatrix} x \\ 1 \end{bmatrix}$$

where **0** is a matrix of appropriate size.

Homogeneous coordinates

Alternative representation of the linear transformation can be written with the more compact homogeneous coordinates. Instead of

$$T(x) = \mathbf{A}x + t, \ \forall x$$

One can use the homogeneous coordinates

$$\tilde{T}(x) = \tilde{\mathbf{A}}\tilde{x}, \ \tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & t \\ \mathbf{0} & 1 \end{bmatrix}, \ \tilde{x} = \begin{bmatrix} x \\ 1 \end{bmatrix}$$

where **0** is a matrix of appropriate size. For example in 2D, transformation matrix with the homogeneous coordinates can be written as

$$\tilde{A} = \begin{bmatrix} a_{11} & a_{12} & t_1 \\ a_{21} & a_{22} & t_2 \\ 0 & 0 & 1 \end{bmatrix}$$

Homogeneous coordinates

Alternative representation of the linear transformation can be written with the more compact homogeneous coordinates. Instead of

$$T(x) = \mathbf{A}x + t, \ \forall x$$

One can use the homogeneous coordinates

$$\tilde{T}(x) = \tilde{\mathbf{A}}\tilde{x}, \ \tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & t \\ \mathbf{0} & 1 \end{bmatrix}, \ \tilde{x} = \begin{bmatrix} x \\ 1 \end{bmatrix}$$

where $\mathbf{0}$ is a matrix of appropriate size. For example in 2D, transformation matrix with the homogeneous coordinates can be written as

$$\tilde{A} = \begin{bmatrix} a_{11} & a_{12} & t_1 \\ a_{21} & a_{22} & t_2 \\ 0 & 0 & 1 \end{bmatrix}$$

Question

Can you write the transformation in 3D in the homogeneous coordinates?

Transformations of interest

$$x' = \mathbf{A}x + t$$

- While all **A** matrices define a mapping, not all such mappings are interesting geometric transformations for our purposes.

Transformations of interest

$$x' = \mathbf{A}x + t$$

- While all **A** matrices define a mapping, not all such mappings are interesting geometric transformations for our purposes.
- There are three important transformation classes that we will study:

Transformations of interest

$$x' = \mathbf{A}x + t$$

- While all **A** matrices define a mapping, not all such mappings are interesting geometric transformations for our purposes.
- There are three important transformation classes that we will study:
    - Rigid body transformations

Transformations of interest

$$x' = \mathbf{A}x + t$$

- While all **A** matrices define a mapping, not all such mappings are interesting geometric transformations for our purposes.
- There are three important transformation classes that we will study:
  - Rigid body transformations
  - Similarity transformations

Transformations of interest

$$x' = \mathbf{A}x + t$$

- While all **A** matrices define a mapping, not all such mappings are interesting geometric transformations for our purposes.
- There are three important transformation classes that we will study:
  - Rigid body transformations
  - Similarity transformations
  - Affine transformations

Transformations of interest

$$x' = \mathbf{A}x + t$$

- While all **A** matrices define a mapping, not all such mappings are interesting geometric transformations for our purposes.
- There are three important transformation classes that we will study:
  - Rigid body transformations
  - Similarity transformations
  - Affine transformations
- Additional transformations, such as perspective transformation, are also interesting in computer vision and graphics applications.

Transformations of interest

$$x' = \mathbf{A}x + t$$

- While all **A** matrices define a mapping, not all such mappings are interesting geometric transformations for our purposes.
- There are three important transformation classes that we will study:
  - Rigid body transformations
  - Similarity transformations
  - Affine transformations
- Additional transformations, such as perspective transformation, are also interesting in computer vision and graphics applications.
- We only focus on the three classes that are most widely used in medical image analysis.

Rigid body transformation in 2D

- Rigid body transformations is one of the most commonly used transformation models in medical image analysis.
- The transformation is defined through a rotation and translation.

Rigid body transformation in 2D

- Rigid body transformations is one of the most commonly used transformation models in medical image analysis.
- The transformation is defined through a rotation and translation.
- In 2D, it is parameterized with one angle θ and one translation vector t

$$x' = T(x) = \mathbf{R}x + t, \ \mathbf{R} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

- The transformation rotates the xy-plane about the origin counterclockwise by θ.
- Three parameters in total.

Rigid body transformation in 3D

- Rigid body transformations is one of the most commonly used transformation models in medical image analysis.
- The transformation is defined through a rotation and translation.

Rigid body transformation in 3D

- Rigid body transformations is one of the most commonly used transformation models in medical image analysis.
- The transformation is defined through a rotation and translation.
- In 3D, it is parameterized with three angles θ_{xy}, θ_{xz} and θ_{yz} and a translation vector t.

$$\mathbf{R} = \mathbf{R}(\theta_{xy})\mathbf{R}(\theta_{xz})\mathbf{R}(\theta_{yz})$$

- Each angle defines rotation about another axis:
  - θ_{xy}: rotation about the z-axis (rotation of the xy plane)
  - θ_{xz}: rotation about the y-axis (rotation of the xz plane)
  - θ_{yz}: rotation about the x-axis (rotation of the yz plane)

Rigid body transformation in 3D

- Rigid body transformations is one of the most commonly used transformation models in medical image analysis.
- The transformation is defined through a rotation and translation.
- In 3D, it is parameterized with three angles θ_{xy}, θ_{xz} and θ_{yz} and a translation vector t.

$$\mathbf{R} = \mathbf{R}(\theta_{xy})\mathbf{R}(\theta_{xz})\mathbf{R}(\theta_{yz})$$

- Each angle defines rotation about another axis:
  - θ_{xy}: rotation about the z-axis (rotation of the xy plane)
  - θ_{xz}: rotation about the y-axis (rotation of the xz plane)
  - θ_{yz}: rotation about the x-axis (rotation of the yz plane)

$$R(\theta_{xy}) = \begin{bmatrix} \cos(\theta_{xy}) & -\sin(\theta_{xy}) & 0 \\ \sin(\theta_{xy}) & \cos(\theta_{xy}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad R(\theta_{xz}) = \begin{bmatrix} \cos(\theta_{xz}) & 0 & \sin(\theta_{xz}) \\ 0 & 1 & 0 \\ -\sin(\theta_{xz}) & 0 & \cos(\theta_{xz}) \end{bmatrix}$$

$$R(\theta_{yz}) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & \cos(\theta_{yz}) & -\sin(\theta_{yz}) \\ 0 & \sin(\theta_{yz}) & \cos(\theta_{yz}) \end{bmatrix}$$

Rigid body transformation in 3D

- Rigid body transformations is one of the most commonly used transformation models in medical image analysis.
- The transformation is defined through a rotation and translation.
- In 3D, it is parameterized with three angles θ_{xy}, θ_{xz} and θ_{yz} and a translation vector t.

$$\mathbf{R} = \mathbf{R}(\theta_{xy})\mathbf{R}(\theta_{xz})\mathbf{R}(\theta_{yz})$$

- Each angle defines rotation about another axis:
  - θ_{xy}: rotation about the z-axis (rotation of the xy plane)
  - θ_{xz}: rotation about the y-axis (rotation of the xz plane)
  - θ_{yz}: rotation about the x-axis (rotation of the yz plane)

$$R(\theta_{xy}) = \begin{bmatrix} \cos(\theta_{xy}) & -\sin(\theta_{xy}) & 0 \\ \sin(\theta_{xy}) & \cos(\theta_{xy}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad R(\theta_{xz}) = \begin{bmatrix} \cos(\theta_{xz}) & 0 & \sin(\theta_{xz}) \\ 0 & 1 & 0 \\ -\sin(\theta_{xz}) & 0 & \cos(\theta_{xz}) \end{bmatrix}$$

$$R(\theta_{yz}) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & \cos(\theta_{yz}) & -\sin(\theta_{yz}) \\ 0 & \sin(\theta_{yz}) & \cos(\theta_{yz}) \end{bmatrix}$$

Question

Is $R(\theta_{xy})R(\theta_{yz})R(\theta_{xz}) = R(\theta_{xy})R(\theta_{xz})R(\theta_{yz})$?

Rigid body transformation in 3D

- Rigid body transformations is one of the most commonly used transformation models in medical image analysis.
- The transformation is defined through a rotation and translation.
- In 3D, it is parameterized with three angles θ_{xy}, θ_{xz} and θ_{yz} and a translation vector t.

$$\mathbf{R} = \mathbf{R}(\theta_{xy})\mathbf{R}(\theta_{xz})\mathbf{R}(\theta_{yz})$$

- Each angle defines rotation about another axis:
  - θ_{xy}: rotation about the z-axis (rotation of the xy plane)
  - θ_{xz}: rotation about the y-axis (rotation of the xz plane)
  - θ_{yz}: rotation about the x-axis (rotation of the yz plane)

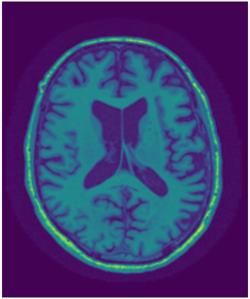
$$R(\theta_{xy}) = \begin{bmatrix} \cos(\theta_{xy}) & -\sin(\theta_{xy}) & 0 \\ \sin(\theta_{xy}) & \cos(\theta_{xy}) & 0 \\ 0 & 0 & 1 \end{bmatrix} R(\theta_{xz}) = \begin{bmatrix} \cos(\theta_{xz}) & 0 & \sin(\theta_{xz}) \\ 0 & 1 & 0 \\ -\sin(\theta_{xz}) & 0 & \cos(\theta_{xz}) \end{bmatrix}$$

$$R(\theta_{yz}) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & \cos(\theta_{yz}) & -\sin(\theta_{yz}) \\ 0 & \sin(\theta_{yz}) & \cos(\theta_{yz}) \end{bmatrix}$$

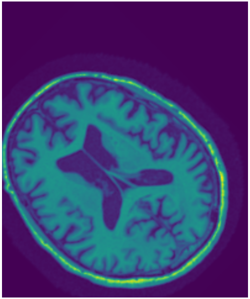
Question

Is $R(\theta_{xy})R(\theta_{yz})R(\theta_{xz}) = R(\theta_{xy})R(\theta_{xz})R(\theta_{yz})$? No.

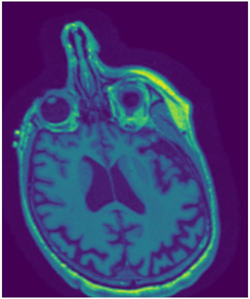
Examples: Rigid transformations



Original



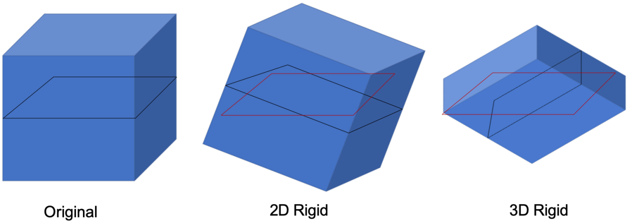
2D Rigid



3D Rigid

- The transformation can push the object out of the FOV.
- 3D rigid transformation can change the visible anatomy in the same slice. Essentially a different cross-section occupies the same slice after transformation.
- Interpolation is performed using bilinear and trilinear methods.

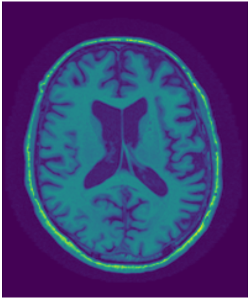
Examples: Rigid transformations



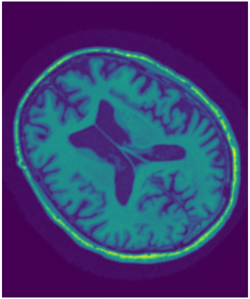
- The transformation can push the object out of the FOV.
- 3D rigid transformation can change the visible anatomy in the same slice. Essentially a different cross-section occupies the same slice after transformation.
- Interpolation is performed using bilinear and trilinear methods.

Center of rotation is important

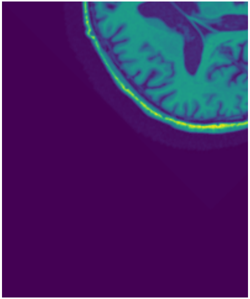
A tricky piece of information that needs to be defined is the centers of coordinate systems in both domains. In the previous examples, the centers of transformation was always taken as the center of the image. However, this can be changed and the resulting transformations also change accordingly.



Original



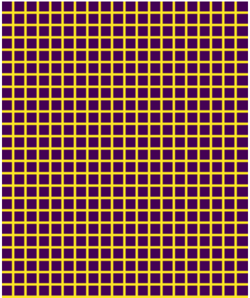
Centers at image centers



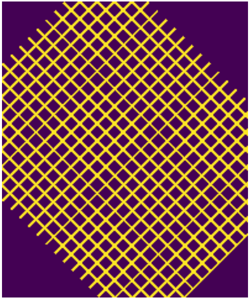
Centers at image origins

Center of rotation is important

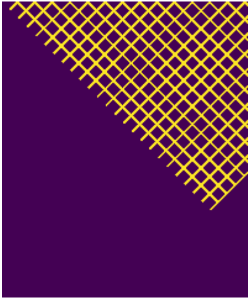
A tricky piece of information that needs to be defined is the centers of coordinate systems in both domains. In the previous examples, the centers of transformation was always taken as the center of the image. However, this can be changed and the resulting transformations also change accordingly.



Original



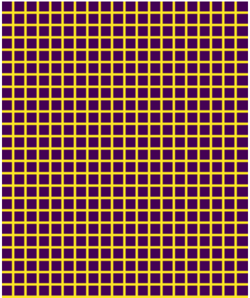
Centers at image centers



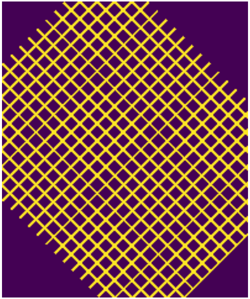
Centers at image origins

Center of rotation is important

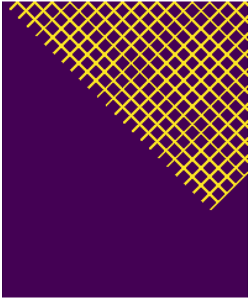
A tricky piece of information that needs to be defined is the centers of coordinate systems in both domains. In the previous examples, the centers of transformation was always taken as the center of the image. However, this can be changed and the resulting transformations also change accordingly.



Original



Centers at image centers



Centers at image origins

Center of transformation applies to all transformations and is especially important for linear transformations where the same matrix applies to the entire image.

Transformations of interest

$$x' = \mathbf{A}x + t$$

- While all **A** matrices define a mapping, not all such mappings are interesting geometric transformations for our purposes.
- There are three important transformation classes that we will study:
  - Rigid body transformations
  - Similarity transformations
  - Affine transformations
- Additional transformations, such as perspective transformation, are also interesting in computer vision and graphics applications.
- We only focus on the three classes that are most widely used in medical image analysis.

Similarity transformation

- Similarity transformation builds on top of rigid body motion by integrating scaling.
- It is also used quite extensively in medical image analysis.
- The transformation is defined through a rotation, translation and scaling.

Similarity transformation

- Similarity transformation builds on top of rigid body motion by integrating scaling.
- It is also used quite extensively in medical image analysis.
- The transformation is defined through a rotation, translation and scaling.
- It is parameterized with a rotation matrix, a translation vector t and a scaling factor

$$x' = T(x) = \mathbf{RS}x + t, \ \mathbf{S} = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} \text{ or } \mathbf{S} = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & s \end{bmatrix}$$

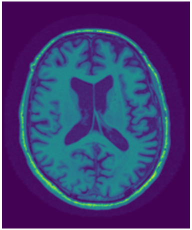
Similarity transformation

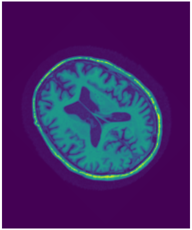
- Similarity transformation builds on top of rigid body motion by integrating scaling.
- It is also used quite extensively in medical image analysis.
- The transformation is defined through a rotation, translation and scaling.
- It is parameterized with a rotation matrix, a translation vector t and a scaling factor

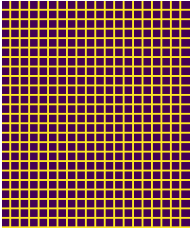
$$x' = T(x) = \mathbf{RS}x + t, \ \mathbf{S} = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} \text{ or } \mathbf{S} = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & s \end{bmatrix}$$

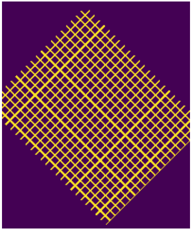
- The scaling "zooms" in or out of the image.
- The transformation rotates the xy-plane about the origin counterclockwise by θ.
- The basic similarity transformation adds one parameter to rigid body motion: Four parameters in 2D and Seven parameters in 3D.

Examples: Similarity transformation

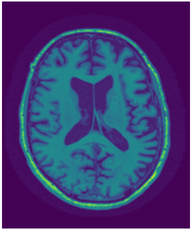

Original


2D Similarity

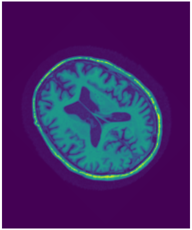

Original grid


Transformed grid

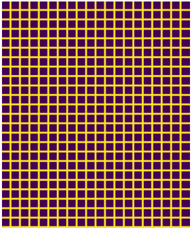
Examples: Similarity transformation



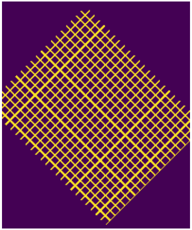
Original



2D Similarity



Original grid

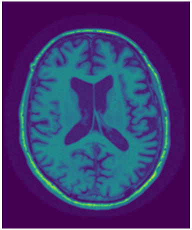


Transformed grid

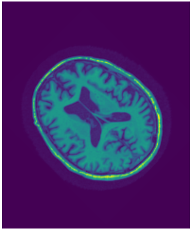
Question

Is $x' = \mathbf{SR}x$ the same as $x' = \mathbf{RS}x$?

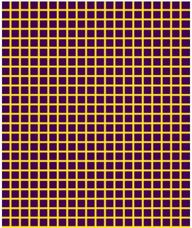
Examples: Similarity transformation



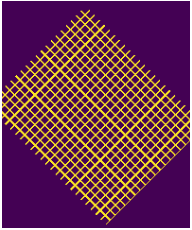
Original



2D Similarity



Original grid



Transformed grid

Question

Is $x' = \mathbf{SR}x$ the same as $x' = \mathbf{RS}x$? Yes. The order of rotation and scaling is not important in similarity transformation.

Transformations of interest

$$x' = \mathbf{A}x + t$$

- While all **A** matrices define a mapping, not all such mappings are interesting geometric transformations for our purposes.
- There are three important transformation classes that we will study:
  - Rigid body transformations
  - Similarity transformations
  - Affine transformations
- Additional transformations, such as perspective transformation, are also interesting in computer vision and graphics applications.
- We only focus on the three classes that are most widely used in medical image analysis.

Extending similarity transformations with Anisotropic Scaling

- Previously we defined similarity transformation that used only one scaling factor.
- One can also think about scaling with different factors in different dimensions.
- In this case, the transformation is defined through a rotation matrix, translation and a scaling matrix.

$$x' = T(x) = \mathbf{RS}x + t, \ \mathbf{S} = \begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix} \text{ or } \mathbf{S} = \begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & s_3 \end{bmatrix}$$

Extending similarity transformations with Anisotropic Scaling

- Previously we defined similarity transformation that used only one scaling factor.
- One can also think about scaling with different factors in different dimensions.
- In this case, the transformation is defined through a rotation matrix, translation and a scaling matrix.

$$x' = T(x) = \mathbf{RS}x + t, \ \mathbf{S} = \begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix} \text{ or } \mathbf{S} = \begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & s_3 \end{bmatrix}$$

- The scaling "zooms" in or out of the image in different amounts in different dimensions.

Extending similarity transformations with Anisotropic Scaling

- Previously we defined similarity transformation that used only one scaling factor.
- One can also think about scaling with different factors in different dimensions.
- In this case, the transformation is defined through a rotation matrix, translation and a scaling matrix.

$$x' = T(x) = \mathbf{RS}x + t, \ \mathbf{S} = \begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix} \text{ or } \mathbf{S} = \begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & s_3 \end{bmatrix}$$

- The scaling "zooms" in or out of the image in different amounts in different dimensions.
- The anisotropic form is no longer a similarity transformation, i.e. shapes are not preserved.

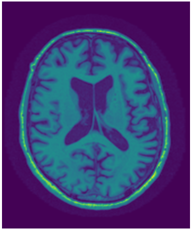
Extending similarity transformations with Anisotropic Scaling

- Previously we defined similarity transformation that used only one scaling factor.
- One can also think about scaling with different factors in different dimensions.
- In this case, the transformation is defined through a rotation matrix, translation and a scaling matrix.

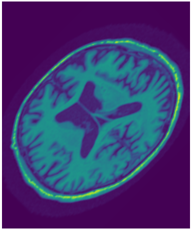
$$x' = T(x) = \mathbf{RS}x + t,\ \mathbf{S} = \begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix} \text{ or } \mathbf{S} = \begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & s_3 \end{bmatrix}$$

- The scaling "zooms" in or out of the image in different amounts in different dimensions.
- The anisotropic form is no longer a similarity transformation, i.e. shapes are not preserved.
- Number of parameters increases:
  - 2D: 1 (rotation) + 2 (translation) + 2 (scaling) = 5 parameters
  - 3D: 3 (rotation) + 3 (translation) + 3 (scaling) = 9 parameters

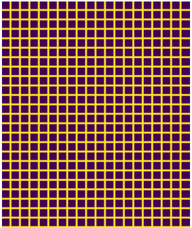
Examples: Anisotropic scaling



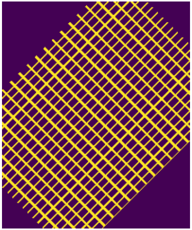
Original



2D Aniso Scaling

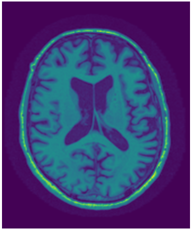


Original grid

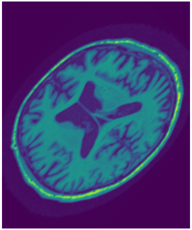


Transformed grid

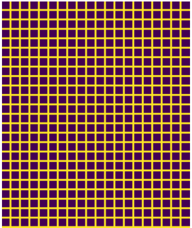
Examples: Anisotropic scaling



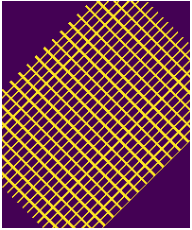
Original



2D Aniso Scaling



Original grid

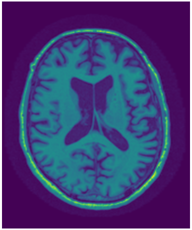


Transformed grid

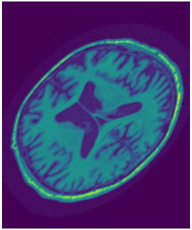
Question

Is $x' = \mathbf{SR}x$ the same as $x' = \mathbf{RS}x$?

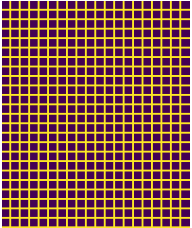
Examples: Anisotropic scaling



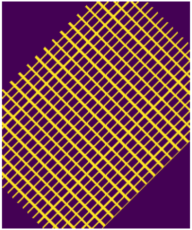
Original



2D Aniso Scaling



Original grid

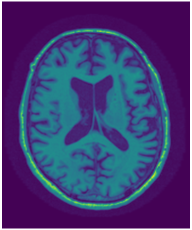


Transformed grid

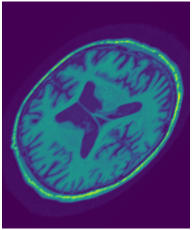
Question

Is $x' = \mathbf{SR}x$ the same as $x' = \mathbf{RS}x$? No. The order of rotation and scaling is very important when applying anisotropic scaling.

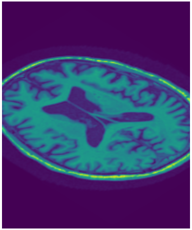
Wrong order gives shearing



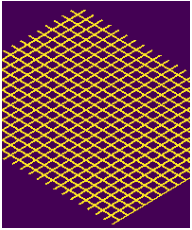
Original



2D Aniso Scaling



Wrong Order



Transformed grid

Applying the transformation with $T(x) = \mathbf{SR}x + t$ yields shearing as seen in the third column. The transformation parameters are the same for the images on the second and third columns.

Affine transformation

- Extends the mapping with anisotropic scaling with *shearing*.
- The transformation is defined through

$$x' = T(x) = \mathbf{WRS}x + t, \ \mathbf{W} = \begin{bmatrix} 1 & w \\ 0 & 1 \end{bmatrix} \text{ or } \mathbf{W} = \begin{bmatrix} 1 & w_1 & w_2 \\ 0 & 1 & w_3 \\ 0 & 0 & 1 \end{bmatrix}$$

with the aditional shearing matrix $\mathbf{W}$.

Affine transformation

- Extends the mapping with anisotropic scaling with *shearing*.
- The transformation is defined through

$$x' = T(x) = \mathbf{WRS}x + t,\ \mathbf{W} = \begin{bmatrix} 1 & w \\ 0 & 1 \end{bmatrix} \text{ or } \mathbf{W} = \begin{bmatrix} 1 & w_1 & w_2 \\ 0 & 1 & w_3 \\ 0 & 0 & 1 \end{bmatrix}$$

  with the aditional shearing matrix **W**.
- Shapes are no longer preserved.

Affine transformation

- Extends the mapping with anisotropic scaling with *shearing*.
- The transformation is defined through

$$x' = T(x) = \mathbf{WRS}x + t, \ \mathbf{W} = \begin{bmatrix} 1 & w \\ 0 & 1 \end{bmatrix} \text{ or } \mathbf{W} = \begin{bmatrix} 1 & w_1 & w_2 \\ 0 & 1 & w_3 \\ 0 & 0 & 1 \end{bmatrix}$$

  with the aditional shearing matrix **W**.
- Shapes are no longer preserved.
- Number of parameters increases:
  - 2D: 1 (rotation) + 2 (translation) + 2 (scaling) + 1 (shearing) = 6 parameters
  - 3D: 3 (rotation) + 3 (translation) + 3 (scaling) + 3 (shearing) = 12 parameters
- Alternative parameterizations are possible.

Affine transformation

- Extends the mapping with anisotropic scaling with *shearing*.
- The transformation is defined through

$$x' = T(x) = \mathbf{WRS}x + t, \; \mathbf{W} = \begin{bmatrix} 1 & w \\ 0 & 1 \end{bmatrix} \text{ or } \mathbf{W} = \begin{bmatrix} 1 & w_1 & w_2 \\ 0 & 1 & w_3 \\ 0 & 0 & 1 \end{bmatrix}$$

  with the aditional shearing matrix $\mathbf{W}$.
- Shapes are no longer preserved.
- Number of parameters increases:
  - 2D: 1 (rotation) + 2 (translation) + 2 (scaling) + 1 (shearing) = 6 parameters
  - 3D: 3 (rotation) + 3 (translation) + 3 (scaling) + 3 (shearing) = 12 parameters
- Alternative parameterizations are possible.
- A full affine transformation would also have reflection. This can be achieved with negative scaling factor.

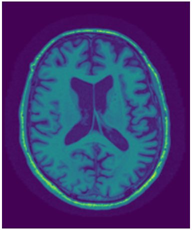
Affine transformation

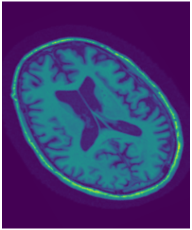
- Extends the mapping with anisotropic scaling with *shearing*.
- The transformation is defined through

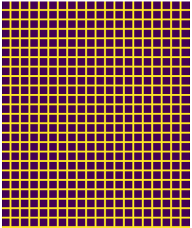
$$x' = T(x) = \mathbf{WRS}x + t, \;\; \mathbf{W} = \begin{bmatrix} 1 & w \\ 0 & 1 \end{bmatrix} \text{ or } \mathbf{W} = \begin{bmatrix} 1 & w_1 & w_2 \\ 0 & 1 & w_3 \\ 0 & 0 & 1 \end{bmatrix}$$

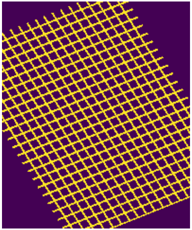
  with the aditional shearing matrix **W**.
- Shapes are no longer preserved.
- Number of parameters increases:
  - 2D: 1 (rotation) + 2 (translation) + 2 (scaling) + 1 (shearing) = 6 parameters
  - 3D: 3 (rotation) + 3 (translation) + 3 (scaling) + 3 (shearing) = 12 parameters
- Alternative parameterizations are possible.
- A full affine transformation would also have reflection. This can be achieved with negative scaling factor.
- Affine transformation with shearing is rarely used in medical image analysis. Shearing model is not used often.

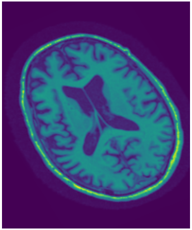
Examples: Affine transformation

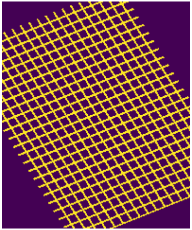

Original


2D Affine


Original grid


Transformed grid


with reflection


Transformed grid

Applications areas of linear transformations - not an exhaustive list

- Rigid transformations - rotation and translation

Applications areas of linear transformations - not an exhaustive list

- Rigid transformations - rotation and translation
    - Longitudinal analysis - aligning temporal sequences.
    - Motion correction - correcting rigid motion between slices in a volumetric acquisition.
    - Multi-modal registration - aligning different images of different modality acquired at the same time, e.g. PET/MRI/CT.

Applications areas of linear transformations - not an exhaustive list

- Rigid transformations - rotation and translation
  - Longitudinal analysis - aligning temporal sequences.
  - Motion correction - correcting rigid motion between slices in a volumetric acquisition.
  - Multi-modal registration - aligning different images of different modality acquired at the same time, e.g. PET/MRI/CT.
- Similarity transformations - rotation, translation and global scaling.

Applications areas of linear transformations - not an exhaustive list

- Rigid transformations - rotation and translation
  - Longitudinal analysis - aligning temporal sequences.
  - Motion correction - correcting rigid motion between slices in a volumetric acquisition.
  - Multi-modal registration - aligning different images of different modality acquired at the same time, e.g. PET/MRI/CT.
- Similarity transformations - rotation, translation and global scaling.
  - Spatial normalization for machine learning applications - removing variations easy to eliminate.
  - Alignment for shape analysis - eliminating global variation in shape.

Applications areas of linear transformations - not an exhaustive list

- Rigid transformations - rotation and translation
  - Longitudinal analysis - aligning temporal sequences.
  - Motion correction - correcting rigid motion between slices in a volumetric acquisition.
  - Multi-modal registration - aligning different images of different modality acquired at the same time, e.g. PET/MRI/CT.
- Similarity transformations - rotation, translation and global scaling.
  - Spatial normalization for machine learning applications - removing variations easy to eliminate.
  - Alignment for shape analysis - eliminating global variation in shape.
- Similarity with anisotropic scaling - rotation, translation and scaling in each dimension.

Applications areas of linear transformations - not an exhaustive list

- Rigid transformations - rotation and translation
  - Longitudinal analysis - aligning temporal sequences.
  - Motion correction - correcting rigid motion between slices in a volumetric acquisition.
  - Multi-modal registration - aligning different images of different modality acquired at the same time, e.g. PET/MRI/CT.
- Similarity transformations - rotation, translation and global scaling.
  - Spatial normalization for machine learning applications - removing variations easy to eliminate.
  - Alignment for shape analysis - eliminating global variation in shape.
- Similarity with anisotropic scaling - rotation, translation and scaling in each dimension.
  - Initialization for non-linear registration.
  - Spatial normalization for machine learning applications.

Applications areas of linear transformations - not an exhaustive list

- Rigid transformations - rotation and translation
  - Longitudinal analysis - aligning temporal sequences.
  - Motion correction - correcting rigid motion between slices in a volumetric acquisition.
  - Multi-modal registration - aligning different images of different modality acquired at the same time, e.g. PET/MRI/CT.
- Similarity transformations - rotation, translation and global scaling.
  - Spatial normalization for machine learning applications - removing variations easy to eliminate.
  - Alignment for shape analysis - eliminating global variation in shape.
- Similarity with anisotropic scaling - rotation, translation and scaling in each dimension.
  - Initialization for non-linear registration.
  - Spatial normalization for machine learning applications.
- Affine transformation - rotation, translation, scaling in each dimension and shear.
  - Initialization for non-linear registration.
  - Studying mechanical tissue properties.
  - Mechanical modeling of intervention.
  - Used to define more complicated non-linear transformations.

Summary

- Linear transformations are used extensively.
- Rigid, similarity and similarity+anisotropic scaling are common in medical image computing.
- Affine transformation is common in modeling interventions.
- Easy to implement

Exercise

Implement linear transformation and resampling classes in your favorite language. Play with random transformations of each type to better understand their effects.

Outline

- Sampling model - How to apply a transformation
- Interpolation models
- Linear transformation models
  - Rigid transformations
  - Similarity transformations
  - Affine transformations
- Non-linear deformable transformation models
  - Kernel-based interpolation models
  - Pixel/Voxel-wise physical models

Subsection 2

Non-linear deformable transformation models

Non-linear functions for displacement fields

The general form of the transformation is:

$$x = T^{-1}(x') = x' + u(x'),$$

where the transformation is defined by the function $u(x')$. Note that

- This function can be non-linear.
- We directly model the inverse transformation from output (target) domain to the input (moving) domain.
- This is due to the difficulty in establishing the inverse transformation for a given arbitrary forward transformation.
- Remember that it is more beneficial from the sampling point of view to work with the inverse transformation.
- The question is "How do we parameterize $u(x')$?"
- We will follow Sotiras, Davatzikos, Paragios. IEEE TMI 2013

Naive parameterization

$$x = T^{-1}(x') = x' + u(x'),$$

where the transformation is defined by the function $u(x')$.

- In the naive parameterization, one defines a displacement vector for each pixel / voxel independent from each other.

Naive parameterization

$$x = T^{-1}(x') = x' + u(x'),$$

where the transformation is defined by the function $u(x')$.

- In the naive parameterization, one defines a displacement vector for each pixel / voxel independent from each other.
- Number of parameters in 2D: (Number of pixels) $\times$ 2, two components for each displacement vector.

Naive parameterization

$$x = T^{-1}(x') = x' + u(x'),$$

where the transformation is defined by the function $u(x')$.

- In the naive parameterization, one defines a displacement vector for each pixel / voxel independent from each other.
- Number of parameters in 2D: (Number of pixels) $\times$ 2, two components for each displacement vector.
- Number of parameters in 3D: (Number of voxels) $\times$ 3, three components for each displacement vector.

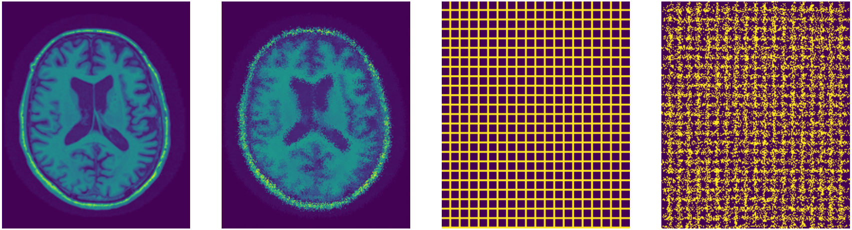
Naive parameterization

$$x = T^{-1}(x') = x' + u(x'),$$

where the transformation is defined by the function $u(x')$.

- In the naive parameterization, one defines a displacement vector for each pixel / voxel independent from each other.
- Number of parameters in 2D: (Number of pixels) $\times$ 2, two components for each displacement vector.
- Number of parameters in 3D: (Number of voxels) $\times$ 3, three components for each displacement vector.
- There are two important issues to consider:
  - Very large number of parameters, very large degrees of freedom. It would be difficult to estimate.
  - This transformation model can generate very "noisy" transformations. Such transformations may not be realistic and even change the topology of the underlying image.

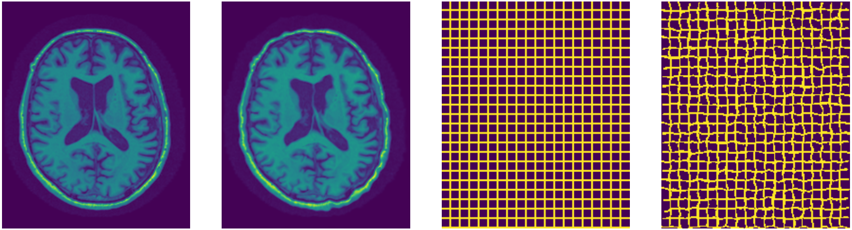
Example of a non-linear transformation field with the naive parameterization



$$x = T^{-1}(x') = x' + u(x), \;\; u(x) \sim \mathcal{N}(0, 1.5\mathbf{I}_2)$$

Random transformations look noisy.

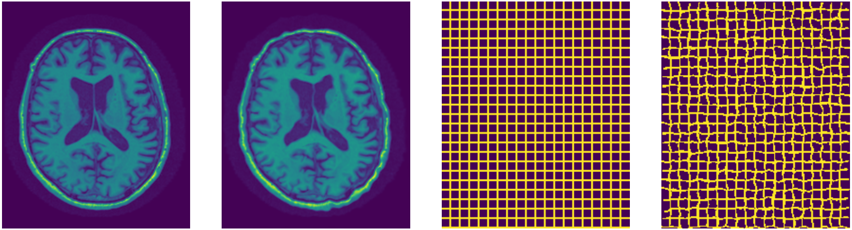
Example of a smooth non-linear transformation field



$$x = T^{-1}(x') = x' + u(x),\;\; u(x) = G_{3.0} * \tilde{u}(x),\;\; \tilde{u}(x) \sim \mathcal{N}(0, 10.0\mathbf{I}_2)$$

where G_κ is a Gaussian kernel of standard deviation κ and $G_\kappa * \tilde{u}(x)$ denotes Gaussian smoothing.

Example of a smooth non-linear transformation field

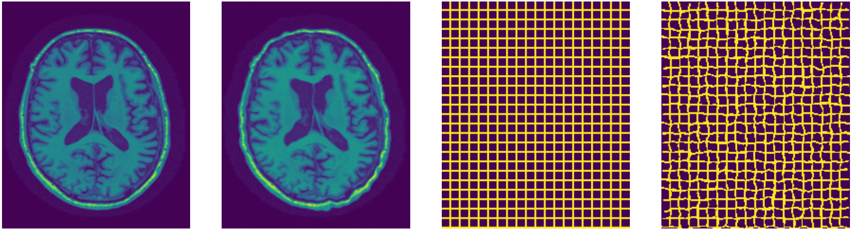


$$x = T^{-1}(x') = x' + u(x), \;\; u(x) = G_{3.0} * \tilde{u}(x), \;\; \tilde{u}(x) \sim \mathcal{N}(0, 10.0\mathbf{I}_2)$$

where G_κ is a Gaussian kernel of standard deviation κ and $G_\kappa * \tilde{u}(x)$ denotes Gaussian smoothing.

- Even when smoothed, displacement fields and resulting transformations can be noisy.

Example of a smooth non-linear transformation field

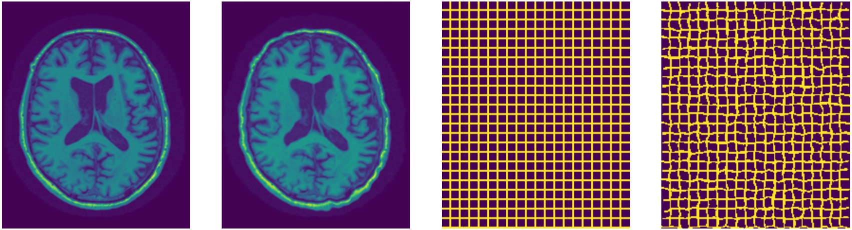


$$x = T^{-1}(x') = x' + u(x),\ \ u(x) = G_{3.0} * \tilde{u}(x),\ \ \tilde{u}(x) \sim \mathcal{N}(0, 10.0\mathbf{I}_2)$$

where G_κ is a Gaussian kernel of standard deviation κ and $G_\kappa * \tilde{u}(x)$ denotes Gaussian smoothing.

- Even when smoothed, displacement fields and resulting transformations can be noisy.
- It would be nice to be able to generate smooth transformations all the time.

Example of a smooth non-linear transformation field

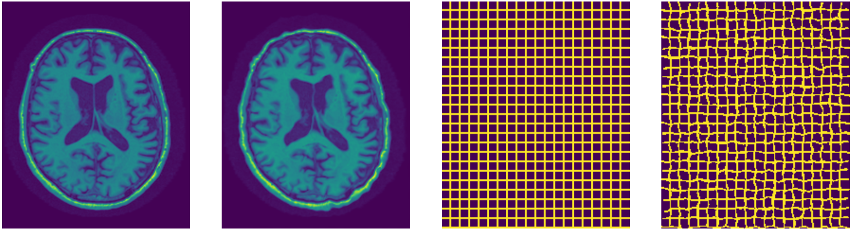


$$x = T^{-1}(x') = x' + u(x), \;\; u(x) = G_{3.0} * \tilde{u}(x), \;\; \tilde{u}(x) \sim \mathcal{N}(0, 10.0\mathbf{I}_2)$$

where G_κ is a Gaussian kernel of standard deviation κ and $G_\kappa * \tilde{u}(x)$ denotes Gaussian smoothing.

- Even when smoothed, displacement fields and resulting transformations can be noisy.
- It would be nice to be able to generate smooth transformations all the time.
- This can be achieved via the parameterization.

Example of a smooth non-linear transformation field

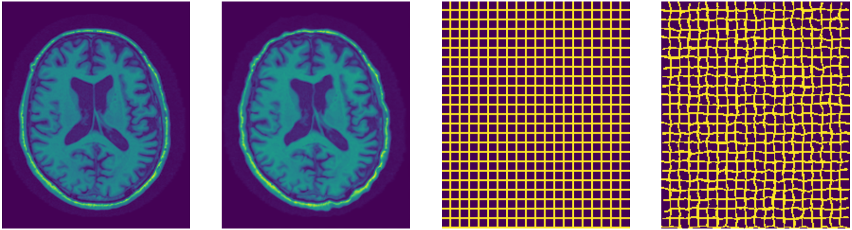


$$x = T^{-1}(x') = x' + u(x),\ \ u(x) = G_{3.0} * \tilde{u}(x),\ \ \tilde{u}(x) \sim \mathcal{N}(0, 10.0\mathbf{I}_2)$$

where G_κ is a Gaussian kernel of standard deviation κ and $G_\kappa * \tilde{u}(x)$ denotes Gaussian smoothing.

- Even when smoothed, displacement fields and resulting transformations can be noisy.
- It would be nice to be able to generate smooth transformations all the time.
- This can be achieved via the parameterization.
- We need models that can generate smooth displacement fields, effectively reducing the number of free parameters.

Example of a smooth non-linear transformation field



$$x = T^{-1}(x') = x' + u(x),\ \ u(x) = G_{3.0} * \tilde{u}(x),\ \ \tilde{u}(x) \sim \mathcal{N}(0, 10.0\mathbf{I}_2)$$

where G_κ is a Gaussian kernel of standard deviation κ and $G_\kappa * \tilde{u}(x)$ denotes Gaussian smoothing.

- Even when smoothed, displacement fields and resulting transformations can be noisy.
- It would be nice to be able to generate smooth transformations all the time.
- This can be achieved via the parameterization.
- We need models that can generate smooth displacement fields, effectively reducing the number of free parameters.
- There are two strategies to this end:
  - Interpolation based
  - Physical model based

Interpolation based methods

Main idea

Define displacement vectors for sparse set of “control” points and interpolate in between with a smooth model.

Interpolation based methods

Main idea

Define displacement vectors for sparse set of “control” points and interpolate in between with a smooth model.

- Radial basis functions:

$$u(x') = \sum_{n=1}^{N} \phi(|x' - x'_n|)d_n, \ \phi : \mathbb{R}^+ \to \mathbb{R}, \ d_n \in \mathbb{R}^{2 \text{ or } 3}$$

where x_n are the control points, ϕ are the basis functions and d_n are the non-linear coefficients, defining the transformation.

Interpolation based methods

Main idea

Define displacement vectors for sparse set of "control" points and interpolate in between with a smooth model.

- Radial basis functions:

$$u(x') = \sum_{n=1}^{N} \phi(|x' - x'_n|) d_n, \;\; \phi : \mathbb{R}^+ \to \mathbb{R}, \;\; d_n \in \mathbb{R}^{2 \text{ or } 3}$$

  where x_n are the control points, ϕ are the basis functions and d_n are the non-linear coefficients, defining the transformation.

- Two commonly used forms:
  - Thin Plate Splines (TPS)
  - Free Form Deformations (FFD) with B-spline.

Interpolation based methods

Main idea

Define displacement vectors for sparse set of "control" points and interpolate in between with a smooth model.

- Radial basis functions:

$$u(x') = \sum_{n=1}^{N} \phi(|x' - x'_n|)d_n, \;\; \phi : \mathbb{R}^+ \to \mathbb{R}, \;\; d_n \in \mathbb{R}^{2 \text{ or } 3}$$

  where x_n are the control points, ϕ are the basis functions and d_n are the non-linear coefficients, defining the transformation.

- Two commonly used forms:
  - Thin Plate Splines (TPS)
  - Free Form Deformations (FFD) with B-spline.
- Let us analyze these two forms.

Thin plate splines (TPS)

Radial basis functions model:

$$u(x') = \sum_{n=1}^{N} \phi(|x' - x'_n|)d_n, \ \phi : \mathbb{R}^+ \to \mathbb{R}, \ d_n \in \mathbb{R}^{2 \text{ or } 3}$$

where x_n are the control points, ϕ are the basis functions and d_n are the non-linear coefficients.

Thin plate splines (TPS)

Radial basis functions model:

$$u(x') = \sum_{n=1}^{N} \phi(|x' - x_n'|) d_n, \ \phi : \mathbb{R}^+ \to \mathbb{R}, \ d_n \in \mathbb{R}^{2 \text{ or } 3}$$

where x_n are the control points, ϕ are the basis functions and d_n are the non-linear coefficients.

In TPS, the basis functions are defined as

$$\phi(|x' - x_n'|) = \phi(r) = r^2 \log r$$

Thin plate splines (TPS)

Radial basis functions model:

$$u(x') = \sum_{n=1}^{N} \phi(|x' - x'_n|) d_n, \ \ \phi : \mathbb{R}^+ \to \mathbb{R}, \ \ d_n \in \mathbb{R}^{2 \text{ or } 3}$$

where x_n are the control points, ϕ are the basis functions and d_n are the non-linear coefficients.

In TPS, the basis functions are defined as

$$\phi(|x' - x'_n|) = \phi(r) = r^2 \log r$$

- Sometimes defined together with an affine transformation as follows

$$x = \mathbf{A}x' + t + \sum_{n=1}^{N} \phi(|x' - x'_n|) d_n,$$

where $\mathbf{A}$ and t define a global linear transformation and TPS define local refinement.

Thin plate splines (TPS)

Radial basis functions model:

$$u(x') = \sum_{n=1}^{N} \phi(|x' - x_n'|) d_n, \ \phi : \mathbb{R}^+ \to \mathbb{R}, \ d_n \in \mathbb{R}^{2 \text{ or } 3}$$

where x_n are the control points, ϕ are the basis functions and d_n are the non-linear coefficients.

In TPS, the basis functions are defined as

$$\phi(|x' - x_n'|) = \phi(r) = r^2 \log r$$

- Sometimes defined together with an affine transformation as follows

$$x = \mathbf{A}x' + t + \sum_{n=1}^{N} \phi(|x' - x_n'|) d_n,$$

  where $\mathbf{A}$ and t define a global linear transformation and TPS define local refinement.
- Control points do not have to be uniformly spaced, they can be randomly dispersed.

Thin plate splines (TPS)

Radial basis functions model:

$$u(x') = \sum_{n=1}^{N} \phi(|x' - x'_n|) d_n, \;\; \phi : \mathbb{R}^+ \to \mathbb{R}, \;\; d_n \in \mathbb{R}^{2 \text{ or } 3}$$

where x_n are the control points, ϕ are the basis functions and d_n are the non-linear coefficients.

In TPS, the basis functions are defined as

$$\phi(|x' - x'_n|) = \phi(r) = r^2 \log r$$

- Sometimes defined together with an affine transformation as follows

  $$x = \mathbf{A}x' + t + \sum_{n=1}^{N} \phi(|x' - x'_n|) d_n,$$

  where $\mathbf{A}$ and t define a global linear transformation and TPS define local refinement.
- Control points do not have to be uniformly spaced, they can be randomly dispersed.
- TPS is extensively used, especially for landmark-based registration.

Parametrization in TPS

$$u(x') = \sum_{n=1}^{N} \phi(|x' - x'_n|) d_n, \ \phi(r) = r^2 \log r$$

Parametrization in TPS

$$u(x') = \sum_{n=1}^{N} \phi(|x' - x'_n|)d_n, \ \phi(r) = r^2 \log r$$

- In landmark-based registration, where TPS is extensively used, control points are given by the user.

Parametrization in TPS

$$u(x') = \sum_{n=1}^{N} \phi(|x' - x'_n|)d_n, \ \phi(r) = r^2 \log r$$

- In landmark-based registration, where TPS is extensively used, control points are given by the user.
- The parameters of the nonlinear transformation model are the coefficients $\{d_n\}_{n=1}^{N}$.
  - In 2D: (Number of control points) $\times$ 2
  - In 3D: (Number of control points) $\times$ 3

Parametrization in TPS

$$u(x') = \sum_{n=1}^{N} \phi(|x' - x'_n|)d_n, \ \phi(r) = r^2 \log r$$

- In landmark-based registration, where TPS is extensively used, control points are given by the user.
- The parameters of the nonlinear transformation model are the coefficients $\{d_n\}_{n=1}^{N}$.
  - In 2D: (Number of control points) $\times$ 2
  - In 3D: (Number of control points) $\times$ 3
- If affine transformation is included then there are the parameters of the linear transformation $\mathbf{A}x' + t$.

$$x = \mathbf{A}x' + t + \sum_{n=1}^{N} \phi(|x' - x'_n|)d_n,$$

Parametrization in TPS

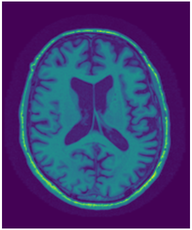
$$u(x') = \sum_{n=1}^{N} \phi(|x' - x'_n|) d_n, \ \phi(r) = r^2 \log r$$

- In landmark-based registration, where TPS is extensively used, control points are given by the user.
- The parameters of the nonlinear transformation model are the coefficients $\{d_n\}_{n=1}^{N}$.
  - In 2D: (Number of control points) $\times$ 2
  - In 3D: (Number of control points) $\times$ 3
- If affine transformation is included then there are the parameters of the linear transformation $\mathbf{A}x' + t$.

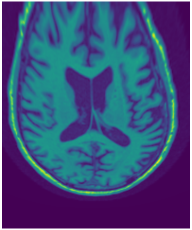
$$x = \mathbf{A}x' + t + \sum_{n=1}^{N} \phi(|x' - x'_n|) d_n,$$

- TPS is a nonlinear deformable transformation model with very few number of parameters.

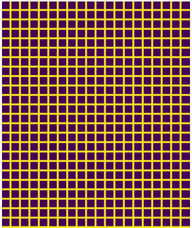
Examples: TPS with 50 control points



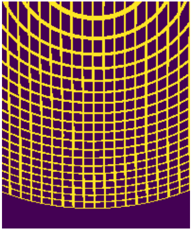
Original



2D Random TPS



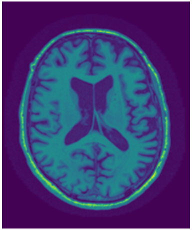
Original grid

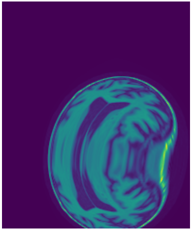


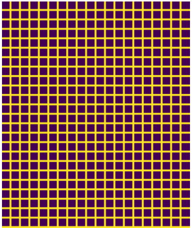
Transformed grid

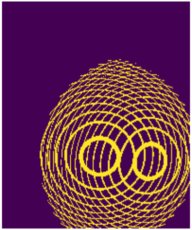
Leads to smooth transformations that can also apply non-linear deformations to the object in the image.

Examples: TPS with 50 control points


Original

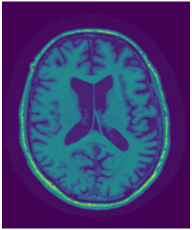

2D Random TPS


Original grid

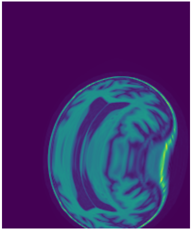

Transformed grid

Leads to smooth transformations that can also apply non-linear deformations to the object in the image.

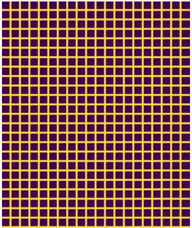
Examples: TPS with 50 control points



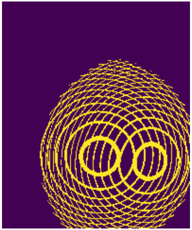
Original



2D Random TPS



Original grid



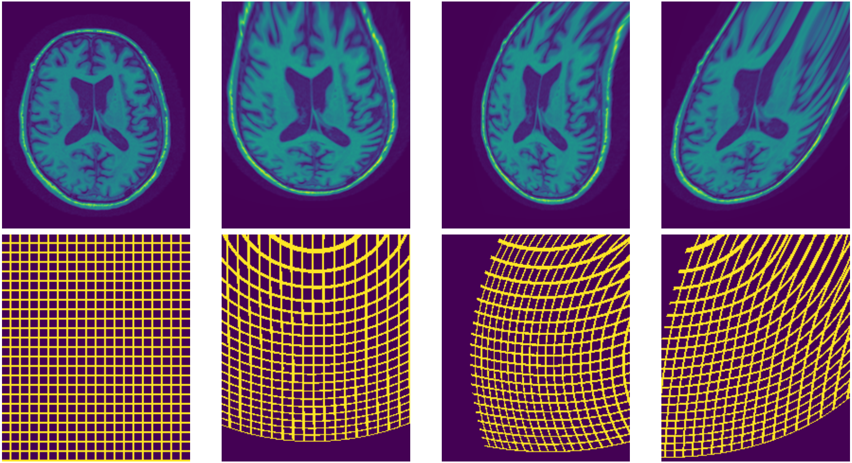
Transformed grid

Leads to smooth transformations that can also apply non-linear deformations to the object in the image.

Question

For 50 control points, how many parameters are there in the TPS model?

Increasing the number of control points in TPS



Original 25 CP 100 CP 400CP

Increasing the number of control points can give more complicated transformations.

TPS analysis

$$u(x') = \sum_{n=1}^{N} \phi(|x' - x'_n|) d_n, \ \phi(r) = r^2 \log r$$

- Transformations are bound to be smooth due to the interpolating kernel ϕ.
- Effects of each control point and the displacement field on it are global.
- Increasing the control points can lead to more complicated transformations.
- Mostly used for landmark-based registration not intensity-based registration.

Free Form Deformation with B-splines

Radial basis functions model:

$$u(x') = \sum_{n=1}^{N} \phi(|x' - x'_n|) d_n, \ \phi : \mathbb{R}^+ \to \mathbb{R}, \ d_n \in \mathbb{R}^{2 \text{ or } 3}$$

Free Form Deformation with B-splines

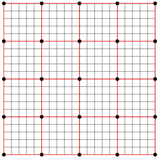
Radial basis functions model:

$$u(x') = \sum_{n=1}^{N} \phi(|x' - x'_n|) d_n,\ \ \phi : \mathbb{R}^+ \to \mathbb{R},\ \ d_n \in \mathbb{R}^{2 \text{ or } 3}$$

Free Form Deformations (FFD) with B-spline is based on (in 3D)

$$u(x') = \sum_{l=0}^{3} \sum_{m=0}^{3} \sum_{n=0}^{3} B_l(\mu_1) B_m(\mu_2) B_n(\mu_3) d_{i+l,j+m,k+n}$$

There are $N_x \times N_y \times N_z$ control points placed in a uniform grid with spacing δ.



Gray is the image grid and red is the grid for the FFD control points. Displacement vectors d are given on the control points only. $u(x')$ is interpolated for the image grid points.

FFD with B-splines formulation

Free Form Deformations (FFD) with B-spline is based on (in 3D)

$$u(x') = \sum_{l=0}^{3} \sum_{m=0}^{3} \sum_{n=0}^{3} B_l(\mu_1) B_m(\mu_2) B_n(\mu_3) d_{i+l,j+m,k+n}$$

- There are $N_x \times N_y \times N_z$ control points placed in a uniform grid with spacing δ.

FFD with B-splines formulation

Free Form Deformations (FFD) with B-spline is based on (in 3D)

$$u(x') = \sum_{l=0}^{3}\sum_{m=0}^{3}\sum_{n=0}^{3} B_l(\mu_1)B_m(\mu_2)B_n(\mu_3)d_{i+l,j+m,k+n}$$

- There are $N_x \times N_y \times N_z$ control points placed in a uniform grid with spacing δ.
- Parameters are defined as (in 3D):
  - Indices for the control points

$$i = \lfloor \frac{x_1'}{\delta} \rfloor - 1,\ j = \lfloor \frac{x_2'}{\delta} \rfloor - 1,\ k = \lfloor \frac{x_3'}{\delta} \rfloor - 1$$

  - Distances to control

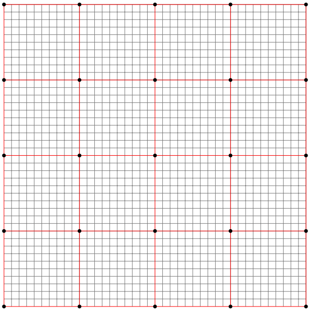
$$\mu_1 = \frac{x_1}{\delta} - \lfloor \frac{x_1}{\delta} \rfloor,\ \mu_2 = \frac{x_2}{\delta} - \lfloor \frac{x_2}{\delta} \rfloor,\ \mu_3 = \frac{x_3}{\delta} - \lfloor \frac{x_3}{\delta} \rfloor$$

  - Basis functions are

$$\begin{aligned} B_0(s) &= (1-s)^3/6 \\ B_1(s) &= (3s^3 - 6s^2 + 4)/6 \\ B_2(s) &= (-3s^3 + 3s^2 + 3s + 1)/6 \\ B_3(s) &= s^3/6 \end{aligned}$$

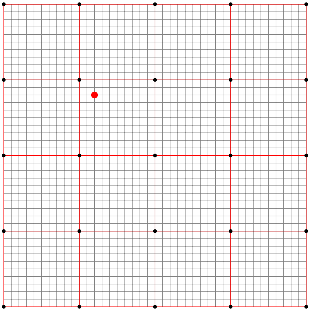
FFD with B-splines - graphical explanation

$$
\begin{aligned}
u(x') &= \sum_{l=0}^{3}\sum_{m=0}^{3}\sum_{n=0}^{3} B_l(\mu_1)B_m(\mu_2)B_n(\mu_3)d_{i+l,j+m,k+n} \\
i &= \lfloor \frac{x_1'}{\delta} \rfloor - 1,\ j = \lfloor \frac{x_2'}{\delta} \rfloor - 1,\ k = \lfloor \frac{x_3'}{\delta} \rfloor - 1 \\
\mu_1 &= \frac{x_1}{\delta} - \lfloor \frac{x_1}{\delta} \rfloor,\ \mu_2 = \frac{x_2}{\delta} - \lfloor \frac{x_2}{\delta} \rfloor,\ \mu_3 = \frac{x_3}{\delta} - \lfloor \frac{x_3}{\delta} \rfloor
\end{aligned}
$$



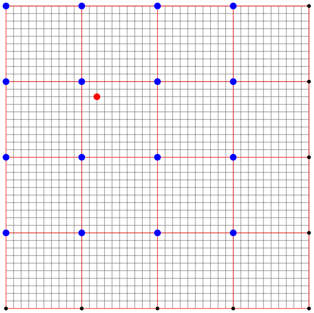
FFD with B-splines - graphical explanation

$$
\begin{aligned}
u(x') &= \sum_{l=0}^{3}\sum_{m=0}^{3}\sum_{n=0}^{3} B_l(\mu_1)B_m(\mu_2)B_n(\mu_3)d_{i+l,j+m,k+n} \\
i &= \lfloor \frac{x'_1}{\delta} \rfloor - 1,\ j = \lfloor \frac{x'_2}{\delta} \rfloor - 1,\ k = \lfloor \frac{x'_3}{\delta} \rfloor - 1 \\
\mu_1 &= \frac{x_1}{\delta} - \lfloor \frac{x_1}{\delta} \rfloor,\ \mu_2 = \frac{x_2}{\delta} - \lfloor \frac{x_2}{\delta} \rfloor,\ \mu_3 = \frac{x_3}{\delta} - \lfloor \frac{x_3}{\delta} \rfloor
\end{aligned}
$$

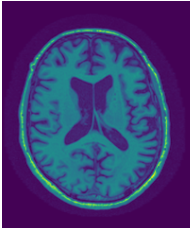


FFD with B-splines - graphical explanation

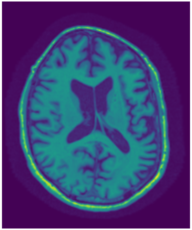
$$
\begin{aligned}
u(x') &= \sum_{l=0}^{3}\sum_{m=0}^{3}\sum_{n=0}^{3} B_l(\mu_1)B_m(\mu_2)B_n(\mu_3)d_{i+l,j+m,k+n} \\
i &= \lfloor \frac{x_1'}{\delta} \rfloor - 1,\ j = \lfloor \frac{x_2'}{\delta} \rfloor - 1,\ k = \lfloor \frac{x_3'}{\delta} \rfloor - 1 \\
\mu_1 &= \frac{x_1}{\delta} - \lfloor \frac{x_1}{\delta} \rfloor,\ \mu_2 = \frac{x_2}{\delta} - \lfloor \frac{x_2}{\delta} \rfloor,\ \mu_3 = \frac{x_3}{\delta} - \lfloor \frac{x_3}{\delta} \rfloor
\end{aligned}
$$



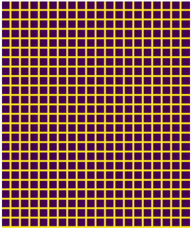
Examples: FFD with BSplines with 25 control points



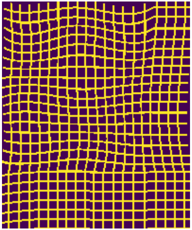
Original



2D FFD w BSplines



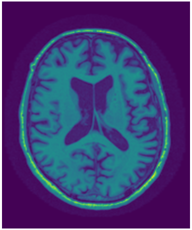
Original grid



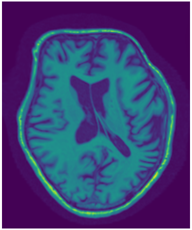
Transformed grid

Leads to smooth transformations that can also apply non-linear deformations to the object in the image.

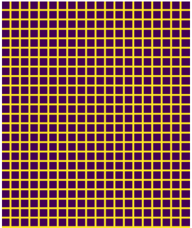
Examples: FFD with BSplines with 25 control points



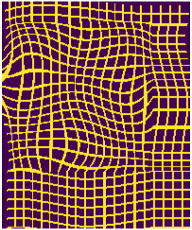
Original



2D FFD w BSpliens



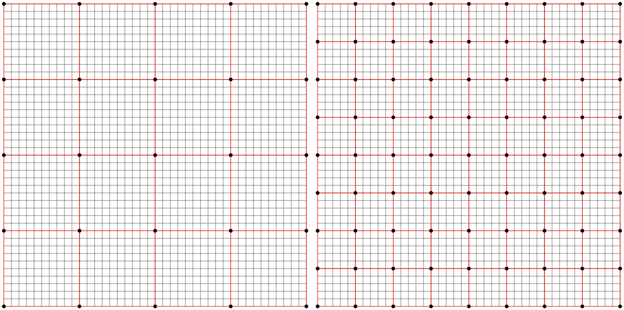
Original grid



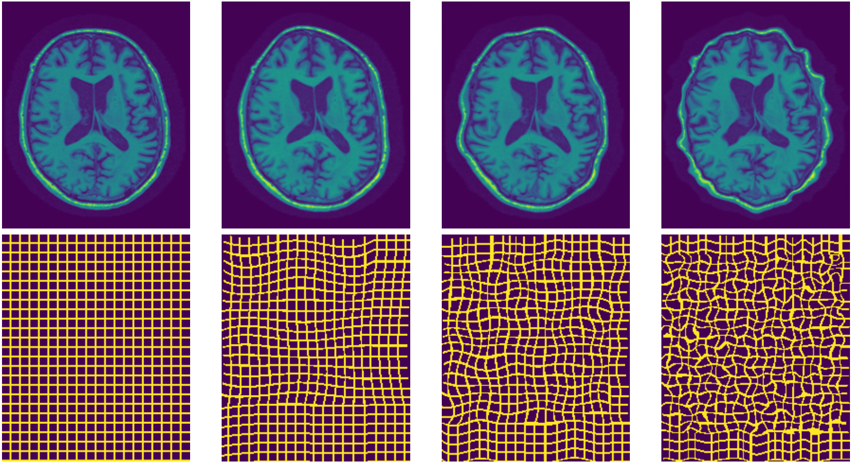
Transformed grid

Leads to smooth transformations that can also apply non-linear deformations to the object in the image.
Even when the displacements at the control points are large.

Increasing the number of control points



Increasing the number of control points in FFD with BSplines



Original	25 CP	100 CP	400CP

Increasing the number of control points gives us less smooth transformations.

Parameterization in FFD with B-splines

$$
\begin{aligned}
u(x') &= \sum_{l=0}^{3}\sum_{m=0}^{3}\sum_{n=0}^{3} B_l(\mu_1)B_m(\mu_2)B_n(\mu_3)d_{i+l,j+m,k+n} \\
i &= \lfloor \frac{x'_1}{\delta} \rfloor - 1,\ j = \lfloor \frac{x'_2}{\delta} \rfloor - 1,\ k = \lfloor \frac{x'_3}{\delta} \rfloor - 1 \\
\mu_1 &= \frac{x_1}{\delta} - \lfloor \frac{x_1}{\delta} \rfloor,\ \mu_2 = \frac{x_2}{\delta} - \lfloor \frac{x_2}{\delta} \rfloor,\ \mu_3 = \frac{x_3}{\delta} - \lfloor \frac{x_3}{\delta} \rfloor
\end{aligned}
$$

Parameterization in FFD with B-splines

$$
\begin{aligned}
u(x') &= \sum_{l=0}^{3}\sum_{m=0}^{3}\sum_{n=0}^{3} B_l(\mu_1)B_m(\mu_2)B_n(\mu_3)d_{i+l,j+m,k+n} \\
i &= \lfloor \frac{x'_1}{\delta} \rfloor - 1,\ j = \lfloor \frac{x'_2}{\delta} \rfloor - 1,\ k = \lfloor \frac{x'_3}{\delta} \rfloor - 1 \\
\mu_1 &= \frac{x_1}{\delta} - \lfloor \frac{x_1}{\delta} \rfloor,\ \mu_2 = \frac{x_2}{\delta} - \lfloor \frac{x_2}{\delta} \rfloor,\ \mu_3 = \frac{x_3}{\delta} - \lfloor \frac{x_3}{\delta} \rfloor
\end{aligned}
$$

- Grid of control points is often subsampled from the original image grid.

Parameterization in FFD with B-splines

$$
\begin{aligned}
u(x') &= \sum_{l=0}^{3}\sum_{m=0}^{3}\sum_{n=0}^{3} B_l(\mu_1)B_m(\mu_2)B_n(\mu_3)d_{i+l,j+m,k+n} \\
i &= \lfloor \frac{x_1'}{\delta} \rfloor - 1,\ j = \lfloor \frac{x_2'}{\delta} \rfloor - 1,\ k = \lfloor \frac{x_3'}{\delta} \rfloor - 1 \\
\mu_1 &= \frac{x_1}{\delta} - \lfloor \frac{x_1}{\delta} \rfloor,\ \mu_2 = \frac{x_2}{\delta} - \lfloor \frac{x_2}{\delta} \rfloor,\ \mu_3 = \frac{x_3}{\delta} - \lfloor \frac{x_3}{\delta} \rfloor
\end{aligned}
$$

- Grid of control points is often subsampled from the original image grid.
- The parameters of the nonlinear transformation model are the coefficients $\{d_n\}_{n=1}^{N}$.
    - In 2D: (Number of control points) $\times$ 2
    - In 3D: (Number of control points) $\times$ 3

  (This is the same as the TPS model.)

Parameterization in FFD with B-splines

$$
\begin{aligned}
u(x') &= \sum_{l=0}^{3}\sum_{m=0}^{3}\sum_{n=0}^{3} B_l(\mu_1)B_m(\mu_2)B_n(\mu_3)d_{i+l,j+m,k+n} \\
i &= \lfloor \frac{x_1'}{\delta} \rfloor - 1,\; j = \lfloor \frac{x_2'}{\delta} \rfloor - 1,\; k = \lfloor \frac{x_3'}{\delta} \rfloor - 1 \\
\mu_1 &= \frac{x_1}{\delta} - \lfloor \frac{x_1}{\delta} \rfloor,\; \mu_2 = \frac{x_2}{\delta} - \lfloor \frac{x_2}{\delta} \rfloor,\; \mu_3 = \frac{x_3}{\delta} - \lfloor \frac{x_3}{\delta} \rfloor
\end{aligned}
$$

- Grid of control points is often subsampled from the original image grid.
- The parameters of the nonlinear transformation model are the coefficients $\{d_n\}_{n=1}^{N}$.
  - In 2D: (Number of control points) $\times$ 2
  - In 3D: (Number of control points) $\times$ 3

  (This is the same as the TPS model.)
- One can again include an affine transformation similar to TPS model.

Parameterization in FFD with B-splines

$$
\begin{aligned}
u(x') &= \sum_{l=0}^{3}\sum_{m=0}^{3}\sum_{n=0}^{3} B_l(\mu_1)B_m(\mu_2)B_n(\mu_3)d_{i+l,j+m,k+n} \\
i &= \lfloor \frac{x_1'}{\delta} \rfloor - 1,\ j = \lfloor \frac{x_2'}{\delta} \rfloor - 1,\ \ k = \lfloor \frac{x_3'}{\delta} \rfloor - 1 \\
\mu_1 &= \frac{x_1}{\delta} - \lfloor \frac{x_1}{\delta} \rfloor,\ \mu_2 = \frac{x_2}{\delta} - \lfloor \frac{x_2}{\delta} \rfloor,\ \mu_3 = \frac{x_3}{\delta} - \lfloor \frac{x_3}{\delta} \rfloor
\end{aligned}
$$

- Grid of control points is often subsampled from the original image grid.
- The parameters of the nonlinear transformation model are the coefficients $\{d_n\}_{n=1}^{N}$.
  - In 2D: (Number of control points) $\times$ 2
  - In 3D: (Number of control points) $\times$ 3

  (This is the same as the TPS model.)
- One can again include an affine transformation similar to TPS model.
- One can also consider optimizing locations of the control points - increasing the number of parameters.

FFD with BSplines analysis

$$
\begin{aligned}
u(x') &= \sum_{l=0}^{3}\sum_{m=0}^{3}\sum_{n=0}^{3} B_l(\mu_1)B_m(\mu_2)B_n(\mu_3)d_{i+l,j+m,k+n} \\
i &= \lfloor \frac{x'_1}{\delta} \rfloor - 1,\ j = \lfloor \frac{x'_2}{\delta} \rfloor - 1,\ k = \lfloor \frac{x'_3}{\delta} \rfloor - 1 \\
\mu_1 &= \frac{x_1}{\delta} - \lfloor \frac{x_1}{\delta} \rfloor,\ \mu_2 = \frac{x_2}{\delta} - \lfloor \frac{x_2}{\delta} \rfloor,\ \mu_3 = \frac{x_3}{\delta} - \lfloor \frac{x_3}{\delta} \rfloor
\end{aligned}
$$

- Transformations are bound to be smooth due to the interpolating kernel.
- Effects of each control point and the displacement field on it are local.
- Increasing the number of control points, i.e. using a finer grid, can lead to less smooth transformations.
- This provides the ability to perform multi-resolutional transformations. Start with few and increase the number of control points.
- FFD with BSplines is mostly used for intensity-based registration not necessarily for landmark-based.

Physical models

- Pixel/Voxel-wise modeling - a displacement field for each grid point.
- Assume a physical model of transformation
- Often motivated from physical phenomenon, such as fluids and elastic body.
- Very large number of parameters but restricted transformations based on the underlying model.

Coarse classification:

1. Elastic body models
2. Fluid flow models
3. Curvature registration
4. Diffusion models
5. Flows of diffeomorphisms

PDE-based models - examples

- Elastic body models (linear):

$$\mu\Delta u(x') + (\mu+\lambda)\nabla(\nabla \cdot u(x')) + F(x') = 0$$

  with Δ being the Laplace operator, μ the rigidity, λ Lame's first coefficient, ∇ gradient, $\nabla\cdot$ divergence and $F(x')$ the user defined force field.

- Diffusion models:

$$\Delta u(x') + F(x') = 0$$

  The displacement field satisfies the Possion equation.

- Curvature registration:

$$\Delta^2 u(x') + F(x') = 0$$

- In all the models one looks for $u(x')$ with a displacement vector for each pixel/voxel.

PDE-based models - examples

- Elastic body models (linear):

$$\mu\Delta u(x') + (\mu+\lambda)\nabla(\nabla\cdot u(x')) + F(x') = 0$$

with Δ being the Laplace operator, μ the rigidity, λ Lame's first coefficient, ∇ gradient, $\nabla\cdot$ divergence and $F(x')$ the user defined force field.

- Diffusion models:

$$\Delta u(x') + F(x') = 0$$

The displacement field satisfies the Possion equation.

- Curvature registration:

$$\Delta^2 u(x') + F(x') = 0$$

- In all the models one looks for $u(x')$ with a displacement vector for each pixel/voxel.
- However, the transformation $u(x')$ must satisfy the model equation, hence it has lower effective degrees of freedom.

Flow models

The transformation of each point is defined as an Ordinary Differential Equation (ODE)

$$\frac{dx}{dt} = v(x, t),\ x(0) = x',$$

where the transformation is defined through the time-dependent "velocity" field $v(x, t)$.

Flow models

The transformation of each point is defined as an Ordinary Differential Equation (ODE)

$$\frac{dx}{dt} = v(x,t), \;\; x(0) = x',$$

where the transformation is defined through the time-dependent "velocity" field $v(x,t)$.

The ODE can be solved through integration in time

$$x = \int_0^1 v(x,t) + x' = u(x') + x'$$

Flow models

The transformation of each point is defined as an Ordinary Differential Equation (ODE)

$$\frac{dx}{dt} = v(x,t), \;\; x(0) = x',$$

where the transformation is defined through the time-dependent "velocity" field $v(x,t)$.
The ODE can be solved through integration in time

$$x = \int_0^1 v(x,t) + x' = u(x') + x'$$

- Theory of ODE says that if $v(x,t)$ is smooth then the resulting transformation is a diffeomorphism.

Flow models

The transformation of each point is defined as an Ordinary Differential Equation (ODE)

$$\frac{dx}{dt} = v(x,t), \;\; x(0) = x',$$

where the transformation is defined through the time-dependent "velocity" field $v(x,t)$.
The ODE can be solved through integration in time

$$x = \int_0^1 v(x,t) + x' = u(x') + x'$$

- Theory of ODE says that if $v(x,t)$ is smooth then the resulting transformation is a diffeomorphism.
- This framework leads to the Large Deformation Diffeomorphic Metric Mapping (LDDMM) registration algorithm.

Flow models

The transformation of each point is defined as an Ordinary Differential Equation (ODE)

$$\frac{dx}{dt} = v(x,t),\ \ x(0) = x',$$

where the transformation is defined through the time-dependent "velocity" field $v(x,t)$.
The ODE can be solved through integration in time

$$x = \int_0^1 v(x,t) + x' = u(x') + x'$$

- Theory of ODE says that if $v(x,t)$ is smooth then the resulting transformation is a diffeomorphism.
- This framework leads to the Large Deformation Diffeomorphic Metric Mapping (LDDMM) registration algorithm.
- Number of parameters are even higher - one for each pixel/voxel and time.

Flow with stationary vector fields

A variation of the flow model is formulated with stationary vector field (SVF) (v does not depend on time):

$$x = \int_0^1 v(x) + x' = u(x') + x'$$

Flow with stationary vector fields

A variation of the flow model is formulated with stationary vector field (SVF) (v does not depend on time):

$$x = \int_0^1 v(x) + x' = u(x') + x'$$

- This model has fewer parameters - one velocity vector per pixel/voxel.

Flow with stationary vector fields

A variation of the flow model is formulated with stationary vector field (SVF) (v does not depend on time):

$$x = \int_0^1 v(x) + x' = u(x') + x'$$

- This model has fewer parameters - one velocity vector per pixel/voxel.
- When the vector field is stationary, efficient integration schemes exist.

Outline

- Spatial mapping and transformation models
  - Sampling model - How to apply a transformation
  - Interpolation models
  - Linear transformation models
  - Non-linear transformation models
- Loss functions and registration algorithms
  - Landmark based registration
  - Intensity based registration