TECHNICAL UNIVERSITY OF DENMARK

Exercise 2 - Landmark based registration

Date of submission: 30th of September, 2025

Submitted By:

Holger Max Fløe Lyng, S214776

Maja Rebien Johannesen, S204287

Dorthea Nørregaard, S204295

Othilia Wagner, S214780

Table of Contents

1 Introduction

The purpose of this exercise is to implement 3D affine and rigid image registration based on landmarks found in T1 and T2 MR-images of a patient, the positioning of the patient is not identical in the two images.In the exercise we use the concept of world versus voxel coordinate system. In the world coordinate system we can find the actual distances in the image. There are two major conventions for the world coordinates: RAS, which goes from right to left, anterior to posterior and superior to inferior, LPS which goes from left to right, posterior to anterior and superior to inferior. The convention of directions let us know the true directions in the patient. In the voxel coordinate system we know the voxels indexing but not the actual sizes of the structures depicted in the image. To go from voxel coordinates to world coordinates we use the following formula:

$$\mathbf{x{=}Av{+}t} \tag{1}$$

Where t is the translation vector, A is the transformation matrix generated by the scanner, v is voxel coordinates, and x is world coordinates. To map the voxel coordinates of one scan to the voxel coordinates of another scan the transformation can be rewritten as:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & t_1 \\ a_{2,1} & a_{2,2} & a_{2,3} & t_2 \\ a_{3,1} & a_{3,2} & a_{3,3} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{M} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ 1 \end{pmatrix} \tag{2}$$

The affine matrix M is used to map the voxel coordinates from T2 to the size of the T1 voxel coordinates as:

$$\begin{pmatrix} \mathbf{v_{T2}} \\ 1 \end{pmatrix} = \mathbf{M_{T2}^{-1}} \cdot \mathbf{M_{T1}} \cdot \begin{pmatrix} \mathbf{v_{T1}} \\ 1 \end{pmatrix} \tag{3}$$

In this exercise we have two images with different coordinates systems, to be able to compare the two images correctly we need to map one image into the image space of the second one. To map the from one image space to the other using landmarks we use the following formula:

$$\begin{pmatrix} \mathbf{v_{T2}} \\ 1 \end{pmatrix} = \mathbf{M_{T2}^{-1}} \cdot \mathbf{M} \cdot \mathbf{M_{T1}} \cdot \begin{pmatrix} \mathbf{v_{T1}} \\ 1 \end{pmatrix} \tag{4}$$

2 Task 1 - Coordinate systems

Familiarize yourself with the concept of coordinate systems. In your report, explain why we need to differentiate between "voxel coordinates/indices" $\mathbf{v}$ *and "world coordinates"* $\mathbf{x}$. *Why does the T2-weighted volume look so compressed in the viewer? For the enthusiastic student: calculate the voxel size in each dataset. The world coordinate system used in both the T1-weighted and the T2-weighted scan follows the RAS convention. Equipped with this information, determine the voxel index* $\mathbf{v}$ *of the center of the left eye of the patient in the T1-weighted scan. Do the same for the T2-weighted scan.*

When the image is acquired the voxel size can be set to an arbitrary size. Therefore, when looking at multiple images, it is important to distinguish between voxel coordinates and world coordinates.

- Voxel coordinates are integer indices that identify individual voxels within a 3D image grid.
- World coordinates represent the physical locations of those voxels in real space, typically measured in millimeters.

Let

$$v = (v_1, v_2, v_3)^T$$

denote the voxel coordinate of a voxel in an image of size $N_1 \times N_2 \times N_3$, where $v_d \in \{0, \ldots, N_d - 1\}$ for each dimension $d = 1, 2, 3$.
In addition to storing the $N_1 \times N_2 \times N_3$ array of intensity values, the scanner also provides a 3×3 matrix A and a 3×1 translation vector t, which together define the mapping from voxel coordinates to world coordinates, and are implemented using the eq.:

$$x = Av + t, \tag{5}$$

where

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix} \quad \text{and} \quad \mathbf{t} = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} \tag{6}$$

We can use this to determine which of the eyes is the left eye. We need to look in the world coordinates as we cannot be sure how the image has been acquired and what settings the machine has had.

For it to be the left eye their first coordinate should be negative, as the 0-line goes through the center of the image in the world space.

```
#Finding the affine matrices
A_T1 = T1.affine
A_T2 = T2.affine

#Finding the eye
eye_1_t1 = np.array([62,114,51,1])
eye_2_t1 = np.array([62,114,101,1])

eye_1_t2 = np.array([88,200,9,1])
eye_2_t2 = np.array([153,207,9,1])

#Finding the world coordinates of the eyes
eye_1_t1_world = A_T1 @ eye_1_t1
eye_2_t1_world = A_T1 @ eye_2_t1
eye_1_t2_world = A_T2 @ eye_1_t2
eye_2_t2_world = A_T2 @ eye_2_t2
print( f"Eye 1 T1 world coordinates: {eye_1_t1_world}" )
print( f"Eye 2 T1 world coordinates: {eye_2_t1_world}" )
print( f"Eye 1 T2 world coordinates: {eye_1_t2_world}" )
print( f"Eye 2 T2 world coordinates: {eye_2_t2_world}" )

Eye 1 T1 world coordinates:
[-29.57089194  75.33724889  -7.11575337   1.        ]
Eye 2 T1 world coordinates:
[30.41369669 76.41994342 -6.29240545  1.        ]
Eye 1 T2 world coordinates:
[ 35.61199212 118.5641368  -14.03259224   1.        ]
Eye 2 T2 world coordinates:
[-20.71020572 110.13705958 -28.41050775   1.        ]
```

Here we can see that Eye 1 for T1 and Eye 2 for T2 are the left eyes, as their first index is negative.

3 Task 2 - Resample the T2-weighted scan to the image grid of the T1-weighted scan

In this task you should resample the T2-weighted scan to the image grid of the T1-weighted scan, i.e., create a new 3D volume that has the same size as the T1-weighted volume, but that contains interpolated T2-weighted intensities instead. In particular, for each voxel index $\mathbf{v}_{T1}$ in the T1-weighted image grid, you should compute the corresponding voxel index $\mathbf{v}_{T2}$ in the T2-weighted volume as follows (see section 2.1 in the book):

$$\begin{pmatrix} \mathbf{v_{T2}} \\ 1 \end{pmatrix} = \mathbf{M}_{T2}^{-1} \cdot \mathbf{M}_{T1} \cdot \begin{pmatrix} \mathbf{v_{T1}} \\ 1 \end{pmatrix} .$$

At the location $\mathbf{v}_{T2}$, you should then use cubic B-spline interpolation to determine the intensity in the T2-weighted scan, and store it at index $\mathbf{v}_{T1}$ in the newly created image.Once you have created a new volume like this, visualize it overlaid on the T1-weighted volume as follows: $Viewer(T2_data_resampled/T2_data_resampled.max() + T1_data/T1_data.max())$

We want to resample the T2-weighted scan to the voxel grid of the T1-weighted scan. This required computing for every voxel index v_{T1} in the T1 grid, and the corresponding voxel index v_{T2} in the T2 volume using the relation:

$$\begin{pmatrix} \mathbf{v_{T2}} \\ 1 \end{pmatrix} = \mathbf{M}_{T2}^{-1} \cdot \mathbf{M}_{T1} \cdot \begin{pmatrix} \mathbf{v_{T1}} \\ 1 \end{pmatrix}$$

We then use the computed indices to interpolate the intensities from the T2-weighted scan, which are stored at the corresponding positions in the new volume. We then get a resampled T2 dataset with the same dimensions and voxel spacing as the T1-weighted scan.

```
#Creating the V_stack to
V1,V2,V3 = np.meshgrid(np.arange(T1_data.shape[0]),np.arange(T1_data.shape[1]),np.arange
V_stack = []
V_stack.append( V1.flatten() )
V_stack.append( V2.flatten() )
V_stack.append( V3.flatten() )
V_stack.append( np.ones( V1.size ) )
V_stack = np.array(V_stack)

V_T2_space = np.linalg.inv( A_T2 ) @ A_T1 @ V_stack

#Now we interpolate the T2 data to the T1 grid
V_T2_in_T1_space_intensity = scipy.ndimage.map_coordinates( T2_data, V_T2_space[:3,:])
V_T2_in_T1_space_intensity = V_T2_in_T1_space_intensity.reshape( T1_data.shape )
```

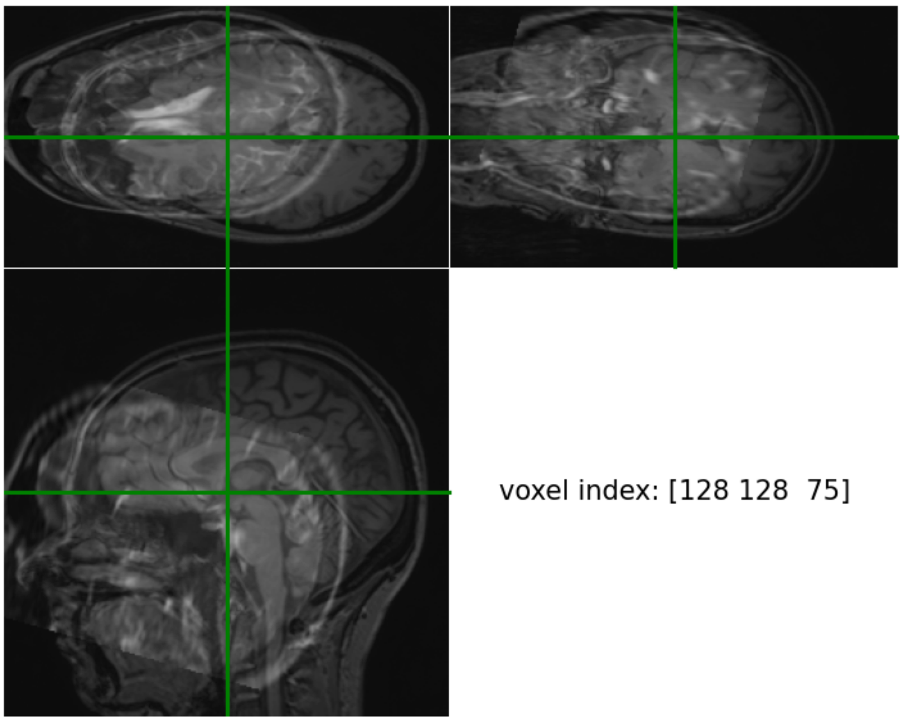

Figure 1: *Overlay of the T1-weighted scan and the resampled T2-weighted scan, both shown in the same voxel grid.*

The resampling produces a new T2-weighted volume alligned to the T1 grid, which makes direct voxel-wise comparison between the two modalities possible. When visualized in the viewer, the resampled T2 scan can be overlaid on the T1-weighted image, see Figure 1. This confirmed that the two data sets now share the same voxel grid. It is important to notice, that there are some anatomical misalignments due to the differences in acquisition parameters and patient positioning.

Task 2 highlights the importance of resampling as a pre-processing step in multimodal imaging. Without a common grid, any attempt at aligning them would be like comparing apples and oranges.

4 Task 3 - Collect corresponding landmarks

Using the viewer class, record the voxel coordinates $\mathbf{v_{T1}}$ and $\mathbf{v_{T2}}$ of at least five corresponding landmarks in the T1- and the T2-weighted volumes, respectively. List them in your report, and explain why you picked them.

We wanted to identify a set of anatomically corresponding landmarks in both the T1- and T2-weighted scans. The landmarks are anchor points that we will later use to estimate transformation matrices for registration.

```
    ## 5 pairwise landmarks
LM_t1 = np.vstack([
    [37, 110, 79, 1],     # Nose ridge
    [240, 110, 78, 1],    # Back skull
    [144, 110, 131, 1],   # Right side skull
    [144, 110, 20, 1],    # Left side skull
    [144, 198, 75, 1]    # Top of brain/skull
])

LM_t2 = np.vstack([
    [125, 231, 8, 1],     # Nose ridge
    [137, 20, 8, 1],      # Back skull
    [55, 116, 8, 1],      # Right side skull
    [206, 116, 8, 1],     # left side skull
    [128, 116, 25, 1]     # Top of brain/skull ])
```

Five pairs of landmarks were selected; the nose ridge, the back of the skull, and the lateral sides of the skull. The coordinates were made as voxel indices in both T1 and T2 arrays (LM_{T1} and LM_{T2}). When distributing the landmarks across different regions of the head, we ensured that they were not clustered in the same plane. The accuracy of the registration steps depends on the quality of these landmarks.

In figure 2 we can observe the chosen landmarks for the registration in the T1 image and in figure 3 the landmarks chosen for the T2 image.

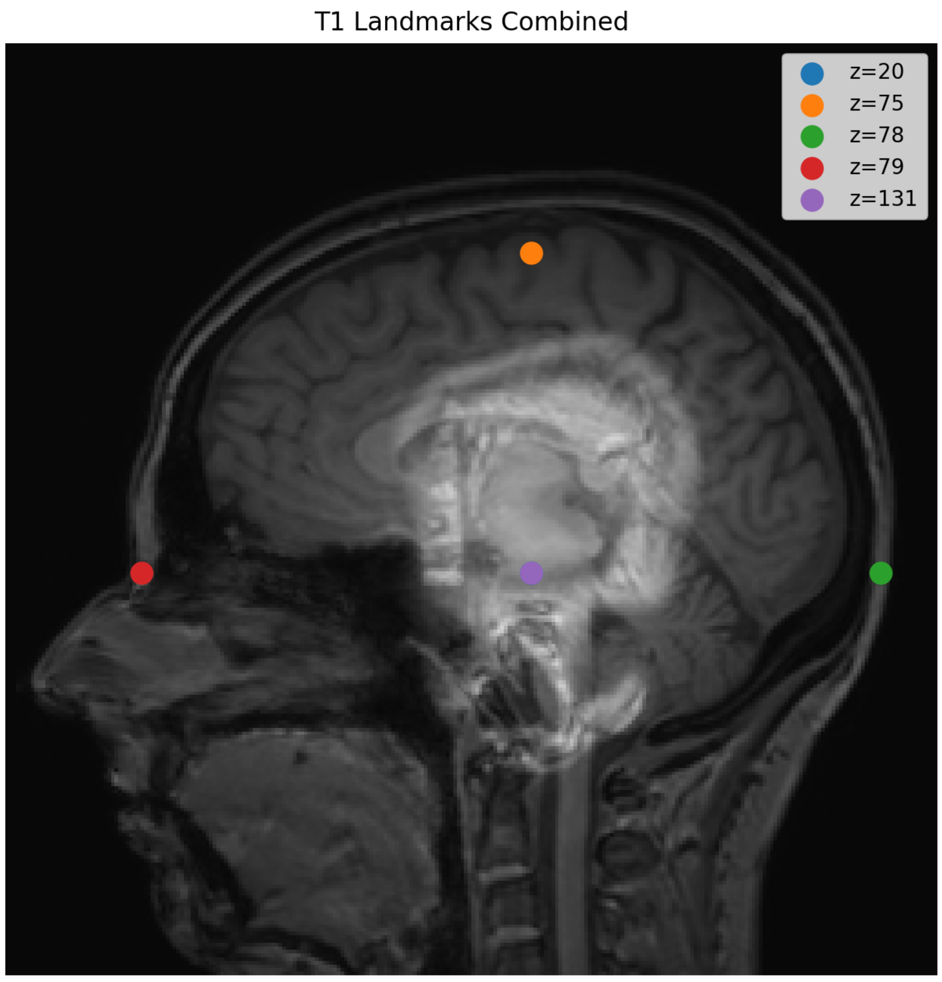

Figure 2: *Overview of the chosen T1 landmarks*

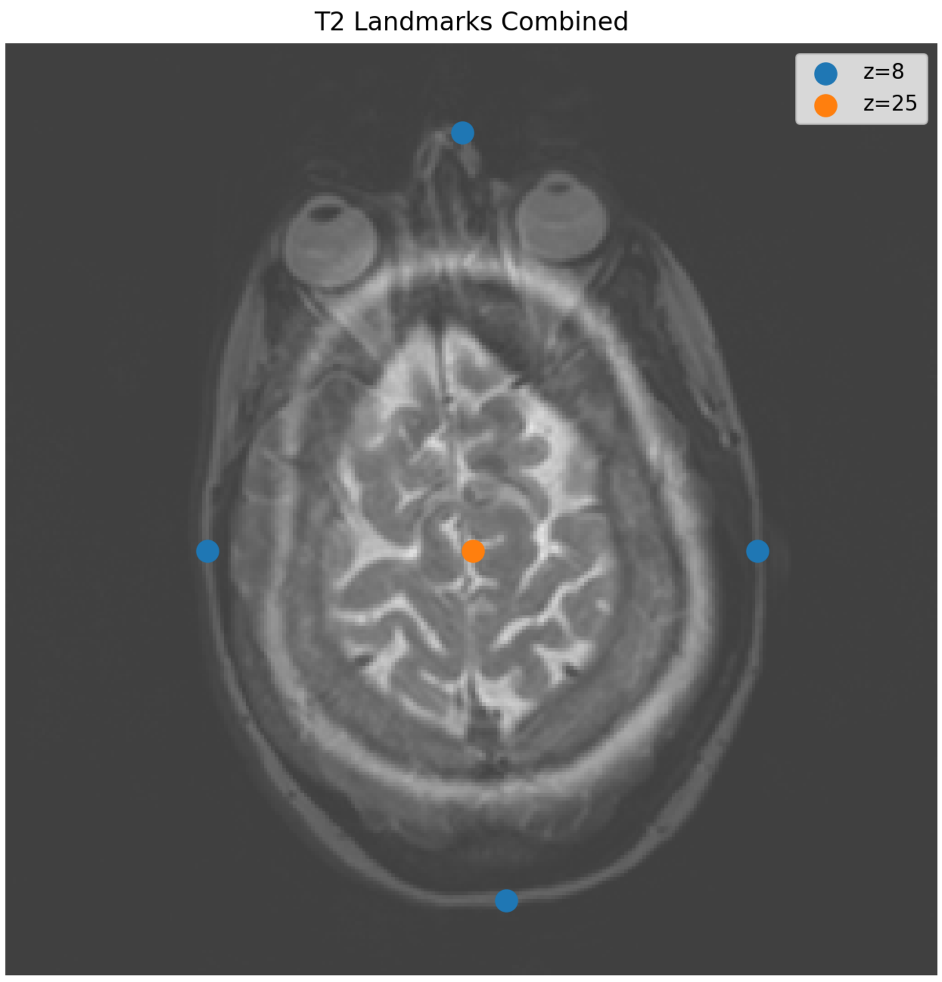

Figure 3: *Overview of the chosen T2 landmarks*

5 Task 4 - Perform affine landmark-based registration

Using the landmarks you recorded in the previous task, compute the parameters of the 3D affine transformation that brings the landmarks in the T1-weighted image closest to the corresponding ones in the T2-weighted image. For this purpose, use Equation (2.8) in the book.

Once you've determined the affine transformation, register to two images by resampling the T2-weighted image to the image grid of the T1-weighted image, and overlay the two images as in Task 2. To map voxel coordinates $\mathbf{v}_T1$ to $\mathbf{v}_T2$, you'll have to use

$$\begin{pmatrix} \mathbf{v_{T2}} \\ 1 \end{pmatrix} = \mathbf{M}_{T2}^{-1} \cdot \mathbf{M} \cdot \mathbf{M}_{T1} \cdot \begin{pmatrix} \mathbf{v_{T1}} \\ 1 \end{pmatrix},$$

where $\mathbf{M}$ is your 4×4 affine matrix (see book).

What happens when you increase/decrease the number of corresponding landmarks that are used in the computations? Comment.

After identifying the anatomically corresponding landmarks (**LM_t1/LM_t2**), the affine transformation from T2 to T1 space is calculated using equation (4). To do this, the positions of **LM_t1/LM_t2** in world coordinates are first obtained with equation (1).

```python
T1_world = (M_1 @ LM_t1.T).T[:, :3] # T1 LM converted to worlf coordinates
T2_world = (M_2 @ LM_t2.T).T[:, :3] # T2 LM converted to worlf coordinates
```

After computing the world coordinates of the T1 and T2 landmarks, the calculation continues with equation 2.8 from the lecture notes:

$$\begin{pmatrix} t_d \\ a_{d,1} \\ \vdots \\ a_{d,D} \end{pmatrix} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1} \mathbf{X}^{\mathrm{T}} \begin{pmatrix} y_{1,d} \\ \vdots \\ y_{N,d} \end{pmatrix}, \tag{7}$$

This equation gives the closed-form least-squares solution for finding the optimal parameters $\mathbf{w}$ of the affine transformation for landmark-based registration. The solution is computed with `np.linalg.lstsq(X, T2_world)`.

```python
# Calculate X from eq 2.8
X = np.hstack([T1_world, np.ones((T1_world.shape[0], 1))])
print(X.T.shape, T2_world.shape)
Calculating best affine matrix fit
A, _, _, _ = np.linalg.lstsq(X, T2_world, rcond=None)#
```

```python

# full affine matrix
M = np.eye(4)
M[:3, :] = A.T # # rotation and scale, translation
```

The affine transformation matrix, which maps the world coordinates of the T1 image (fixed) to those of the T2 image (moving), can then be used with equation (4) to register T2 to T1. Finally,

$$v = (v_1, v_2, v_3)^T$$

defines the T1 mesh grid onto which the T2 image is interpolated.

```python
# Find Big_M affine transformation matrix from T1 to T2
Big_M = np.linalg.solve(M_2, M @ M_1)
V1,V2,V3 = np.meshgrid( np.arange( T1_data.shape[0] ),
                        np.arange( T1_data.shape[1] ),
                        np.arange( T1_data.shape[2] ),
                        indexing='ij' )
V1, V2, V3 = V1.flatten(), V2.flatten(), V3.flatten()
V_T1 = np.column_stack([V1, V2, V3, np.ones(len(V3))])
V_T1 = V_T1.T # T1 Mash grid setup

# Determine the space and size T2 have to interpolated into
T2_space = Big_M @ V_T1
# Interpolation in T2_space
T2_interpolated = map_coordinates(T2_data, T2_space[:3], order=3)
T2_moved = T2_interpolated.reshape(T1_data.shape) # form final image

Viewer(T2_moved / T2_moved.max() + T1_data / T1_data.max())
```

6 Task 5 - Perform rigid landmark-based registration

Repeat Task 4, but this time using a *rigid* transformation model. Vary the number of landmarks that are used again, and comment. Which transformation model (affine or rigid)

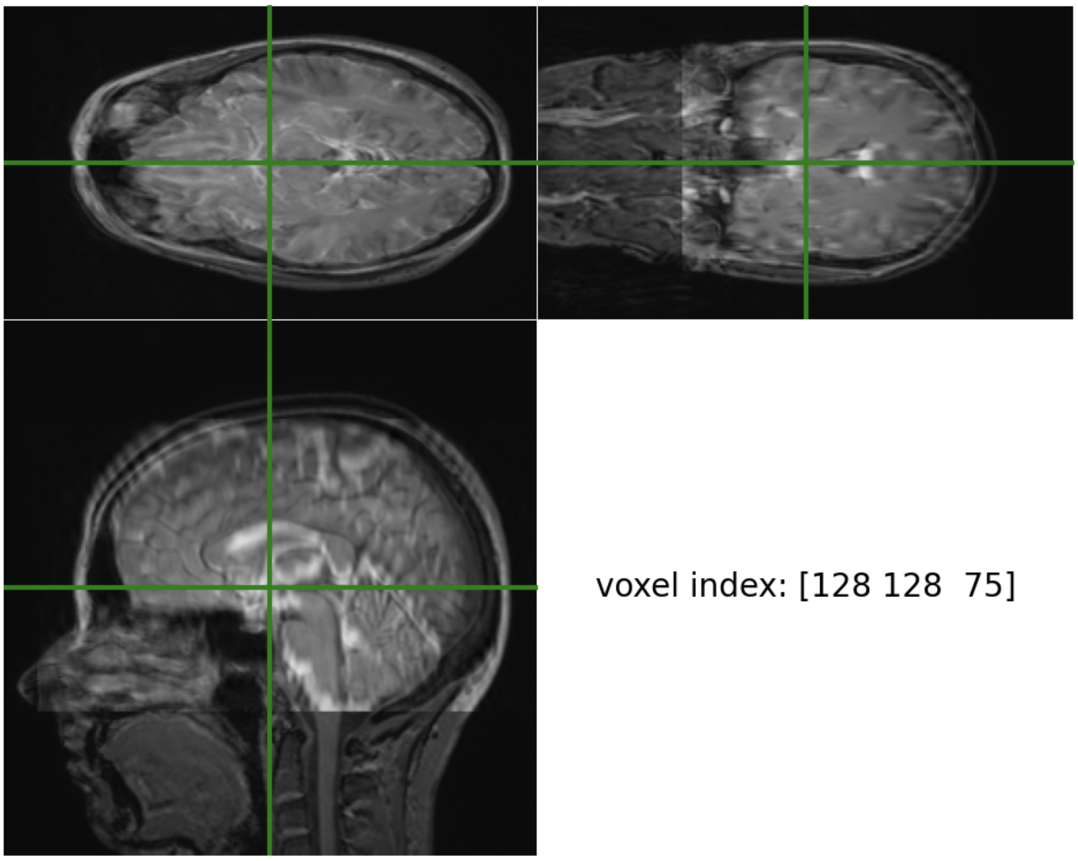

Figure 4: *T2 images registered to the T1 image using affine landmark-based registration.*

is more appropriate to use in this specific application?

Instead of using an affine transformation, in this task a rigid transformation to apply to the T2 image. Unlike the affine transformation, the rigid model can only rotate and translate the image, no scaling nor shearing as in affine transformation. This is important if an image has rigid structures, as the skull, then it's not desirable to change the structure of scans when perfroming a transformation. By using rigid transformation we preserve the true shape.
The rigid model is described as:

$$y(x, w) = Rx + t, \tag{8}$$

where **R** is the rotation matrix and has the constraints $R^T R = I$ and $det(R) = 1$, and t is the translation vector. By computing the singular value decomposition (SVD) of the matrix $\sum_{n=1}^{N} \tilde{x_n} \tilde{y_n}^T = U \Sigma V^T$. $\tilde{x}$ and $\tilde{y}$ are given by $\tilde{y}_n = y_n - \hat{y}_n$ and $\tilde{x}_n = x_n - \hat{x}_n$, $\hat{y}$ and $\hat{x}$ is the mean of the landmarks. From here the SVD can be computed, U, Σ and V^T, are used to compute the rotation matrix **R** and $\tilde{x}$ and $\tilde{y}$ to compute translation vector **t**:

$$R = VU^T \tag{9}$$

$$t = \tilde{y} - R\tilde{x} \tag{10}$$

then ensure constraints on R, U and V^T are respected. The rigid tranformation matrix M is then built from R and t.

```
# Applying rigid transformation to T2
T1_world = (M_1 @ LM_t1.T).T [:, :3] #  5x3
T2_world = (M_2 @ LM_t2.T).T [:, :3] #  5x3
x_hat = T1_world.mean(axis=0)
y_hat = T2_world.mean(axis=0)
x_tilde = T1_world - x_hat
y_tilde = T2_world - y_hat
H = x_tilde.T @ y_tilde # building x_tilde y_tilde matrix for SVD

U, S, V = np.linalg.svd(H)
R = V.T @ U.T # Makes sure determinate is
                det(R) = 1 and that V.T @ V,
                U.T @U = I identity matrix.
if np.linalg.det(R) < 0:
    V[-1, :] *= -1
    R = V.T @ U.T

print("det(R) =", np.linalg.det(R))    # should be     +1
print("R^T R =", R.T @ R)
t = y_hat - R @ x_hat # eq 2.10 finding t

# create M Rotation R and t
M = np.eye(4)
M[:3, :3] = R
M[:3, 3]  = t
```

Next, following the same procedure as in Task 4, we compute the overall transformation matrix using equation (4), define a voxel grid in T1 space,Map voxel coordinates from T1 space into T2 space using the rigid transformation matrix **M**, which represents the world-space alignment from T2 to T1, and perform interpolation of the T2 image.

```
#Mapping T1 voxels to T2 voxels
Big_M = np.linalg.solve(M_2, M @ M_1)

V1,V2,V3 = np.meshgrid( np.arange( T1_data.shape[0] ),
                                np.arange( T1_data.shape[1] ),
                                np.arange( T1_data.shape[2] ), indexing='ij' )
V1, V2, V3 = V1.flatten(), V2.flatten(), V3.flatten()
V_T1 = np.column_stack([V1, V2, V3, np.ones(len(V3))])
V_T1 = V_T1.T

T2_space = Big_M @ V_T1 # Calculation T2_transformation to T1_space

# Interpolate T2 image onto these coordinates, using B-spline order 3
T2_resampled = map_coordinates(T2_data, T2_space[:3], order=3)
T2_resampled = T2_resampled.reshape(T1_data.shape)

Viewer(T2_resampled / T2_resampled.max() + T1_data / T1_data.max())
Viewer(T2_resampled)
```

Figure 5 shows the result of rigid resgistation of the T2 image onto the T1. There is no obvious misalignment of skull, there are some misalignment in the back of the brain near the Occipital lope and cerebellum.

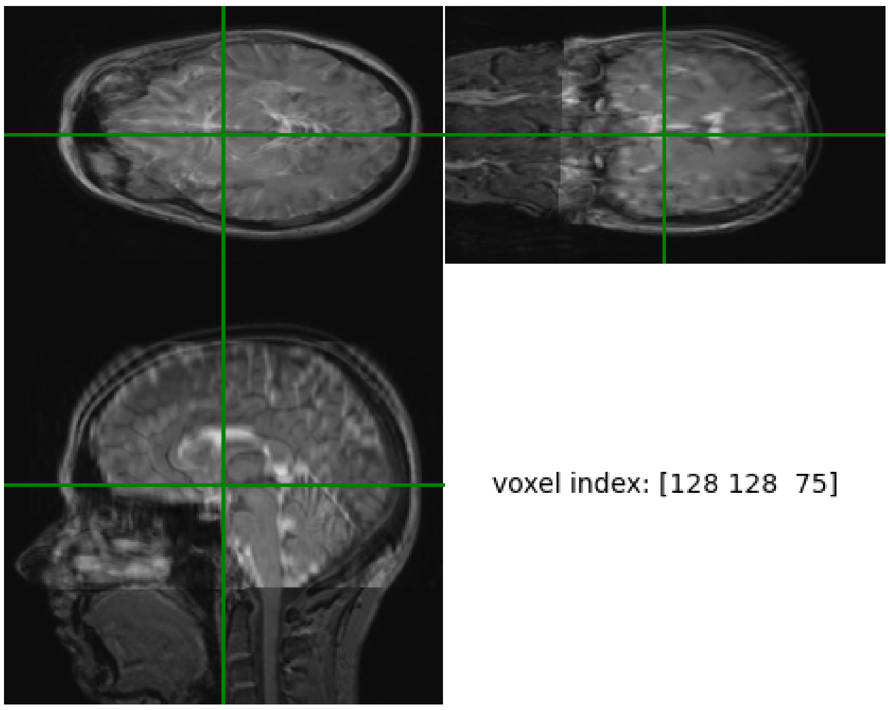

Figure 5: *T2 image registered to T1 image using rigid landmark-based registration*

7 Conclusion

In exercise we have performed affine and rigid transformations from T2 space into T1 space. During the different tasks we learned how we could change the voxel spacing in one image to fit that of another image. This is an important preprocessing step you need to do to get the landmarks aligned in two images. We also learned how to choose appropriate landmarks for registration, and how by adding more landmarks it depends on how accurately your landmarks are based. The affine transformation will always have a fixed number of degrees of freedom (2D = 6 parameters; 3D = 12 parameters). If landmarks are accurately placed, adding more can improve robustness, producing a transformation that better reflects the "average" alignment. However, inaccurately placed landmarks can bias the transformation, introducing distortions such as over-scaling or skewing to compromise between conflicting points.

For rigid structures — such as the head — a rigid transformation is preferable, as it preserves shape by avoiding unwanted scaling or skewing, regardless of the number of landmarks used. Still, careful placement of landmarks is essential to ensure accurate registration.

In medical image registration, it is essential to choose the correct transformation model based on the anatomical structure and desired accuracy. Preprocessing steps like voxel resampling and careful landmark selection strongly influence the quality of registration. Affine transformations offer flexibility but require careful control of landmark accuracy, while rigid transformations offer shape preservation for rigid anatomy, making them a safer choice when scaling or skewing is undesirable.