Image Analysis
Exercise - Advanced registration

Theory

The exercise is to get experience working with 3D image registration pipelines (fig. 1) and the theory part describes the different steps in the pipeline. The focus is on the considerations to be made when selecting the various algorithm parameters to ensure a robust registration result.

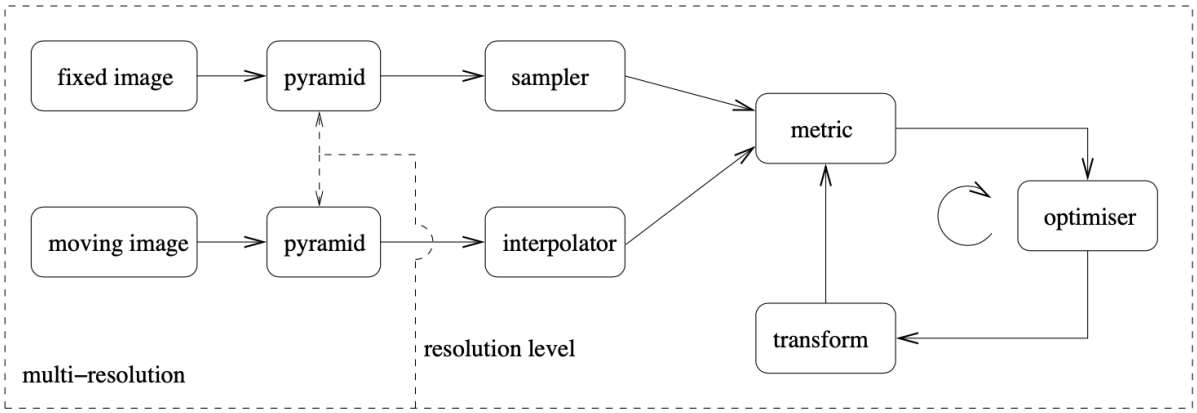

Figure 1: A general outline of an image registration pipeline at its different steps as implemented in the Eleastix toolbox that the exercise and the lecture note are based on (Elastix - The manual by Stefan Klein and Marius Staring).

The exercise will use the “SimpleITK” Python library which is based on the “Advanced registration” lecture note (DTU Learn) on the Elastix toolbox - The manual (chapter 2). The SimpleITK library builds on the algorithms in Elastix but there is no guarantee that the function naming is the same.

In image registration, our aim is to find a geometrical matrix that best registers the moving image with a fixed image of the same scene. Often, the moving image can be named as the "reference" image whereas the fixed image is named as the "template" or the "static" image. In this exercise, we will have different geometrical transformation matrices (the transform step) to be combined and applied to an image or a series of images. Therefore, we wish to combine the transformation matrices in an efficient way - and for this, we will use the Homogeneous coordinate system. We will combine geometrical transformation matrices to an affine matrix and apply it to the moving image by reslicing to place the moving image on the same image grid as the fixed image. In reslicing, we use backward mapping of a data point in the fixed image grid into the moving image and use intensity interpolation between neighboring voxels to find the intensity in the fixed image grid (the interpolation step). There exist different interpolation methods and, in the exercise, we use a simple linear (bilinear) interpolation method. Note that all types of interpolation methods always introduce some degree of blurring and the more advanced interpolation methods e.g., Sinc-interpolation the less blurring is introduced. The price for less blurring is a longer processing time. Further, we will explore the pyramid (the pyramid step) procedure that uses a multi-resolution strategy step to robustly find an optimal affine geometrical transformation matrix (the optimiser and the metric step).

In this exercise, we will work with affine transformations which is a linear geometrical transformation up to 12 degrees of freedom for 3D images i.e., up to 12 free parameters are to be estimated. The 12 parameters are three for the translation, three for rotation, three for scaling, and three for shearing each in the x, y, and z-axis respectively. In this exercise, we will constrain the affine transformation to a ridge transformation including only translation and rotation, reducing the number of free parameters to 6. Hence, the scaling and shearing are set to an identity matrix.

An affine transformation (A) for a 3D image is 3x3 (Eq.1) and is constructed from geometrical transformation matrices being the translation (A_T), rotation (A_R), scaling (A_S), and shearing (A_Z).

$$A_{T,R,S,Z} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \quad (1)$$

Often the affine transformation matrix is constituted by a combination of the different geometrical transformations. However, translation is a summation operation i.e., $\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = A_T + \begin{bmatrix} x \\ y \\ z \end{bmatrix}$ whereas scaling, rotation, and shearing are matrix multiplications of the input points i.e., $\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = A_{S,R,S} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$. Therefore, we wish to find a way to enable multiplication also for the translation matrix so any combination of the geometrical transformations can be combined via matrix multiplication operations.

To combine different matrices via multiplication, we use the Homogeneous coordinate system where we add an extra dimension *W* which is set to 1 (for more information on the Homogeneous coordinate system widely used in computer vision read this blog[1]). Hence, the geometrical matrix for a 3D image becomes a 4D matrix with the form

$$A = \begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

The idea of using the Homogeneous coordinate system is that the translation matrix now can be formulated as

$$A_T = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

and the rotation matrix A_R includes three rotations matrices (R_X, R_y, R_Z) that expresses

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, R_y = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, R_z = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

The scaling (A_S) and the shearing (A_z) matrices become

$$A_s = \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad A_z = \begin{bmatrix} 1 & Sxy & Sxz & 0 \\ Sxy & 1 & Syz & 0 \\ Sxz & Syz & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

[1] www.tomdalling.com/blog/modern-opengl/explaining-homogenous-coordinates-and-projective-geometry/

Now we can write the 4x4 affine matrix as a multiplication between any of the geometrical transformations for 3D images. Often the standard order of combining the different geometrical transformations is

$$A = A_T * A_R * A_s * A_z \quad (6)$$

In the same way as combining the different geometrical transformations, we can combine a series of affine matrices between images as multiplications. Here, the order in which matrices are combined is important too. Beware, that different image registration programs may use different combinatorial orders, and especially the rotations can be counterclockwise or clockwise. Obviously, that will generate different affine matrices, and care should be taken when combining geometrical matrices obtained from different programs - Therefore, always visually inspect your registration results.

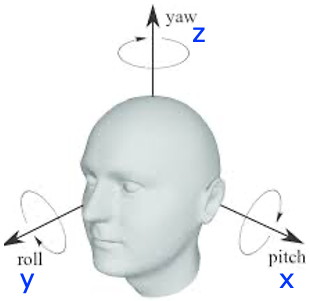

Figure 1: 3D rotation coordinate system

The rotation matrix in Eq 6 defines a 3D rotation coordinate system (Fig. 1) composed of three elements rotations α, β *and* γ for the R_X, R_y, and R_Z matrices, respectively (Eq 4). The rotation around the x-axis is named the pitch, the roll is around the y-axis, and the yaw is around the z-axis. The three rotations angles are combined using the Euler angle conventions in the following order

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = R_x \cdot R_y \cdot R_z \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

When performing rotation by applying a rotation matrix, it is important to define the center of rotation used also when combining different affine matrices e.g., from a sequence of affine image rotations. The center of rotation is defined as an origin and often uses as default the middle point of the image matrix as is the case in the exercise or at position (0,0) (x,y), or the origin can be user-defined so check it for the implementation you use.

Finding the optimal parameters for an affine registration matrix

To find the optimal parameter set for a geometrical transformation that optimally registers the fixed and the moving image we use a similarity matrix as a cost function, and a numerical optimization algorithm (the optimiser step, the metric step, and the transform step). In the exercise, we use the mean square error as a similarity metric (Eq 2.5 in the lecture note). It is fast to estimate and relatively robustly can find the global minimal hence the optimal parameter set. The best robustness is if the structural intensity information in the fixed and the moving image is the same. If that is not the case, the mean square error metric may have more local minima leading to a higher risk of a sub-optimal registration solution. Alternatively, the mutual information metric is based on image histogram matching and is especially good if the fixed and moving images are having similar structural information but contain different intensity information i.e., different intensity histograms. The “price” to pay for the more robust similarity metric is a longer processing time.

The optimizer algorithm iteratively searches in small steps the parameter space of the cost function to find a global minimum. At this point, we expect to find the solution for an optimal parameter set of the affine matrix that best registers the moving image to the fixed image. There exist many different optimizer algorithms and we select the Powell optimizer that better defines the gradient direction in

each iteration than the gradient decent algorithm described in the lecture. Still, one needs to select the step length along the gradient direction and the maximal number of iterations. If the step length is too small one becomes more sensitive to local minima whereas too big a step length can pass the global minimum. If too few iterations are selected, the global minimum will not be reached whereas for too many steps the processing can take a long time especially if no clear solution exists e.g., in the case of very noisy images. So, a suitable number of iterations and not too large nor too small step lengths are to be selected - and good guidance is using the default in the program or simply finding it empirically.
Some types of images can have a risk of containing many local minima i.e., several "local" structures look like others, and we have many suboptimal solutions to the registration problem. This is often the case in MRI where the finer anatomical structures such as cortical folding look the same in many places in the brain. It could also be the case for noisy images.
The pyramidal multi-resolution approach (the pyramid step) is a way to reduce the number of local minima. As the name indicates it works in different image resolution levels like a pyramid shape. The procedure is as the following. We start at the lowest image resolution. Here, one only sees the major/coarse brain structures outlined in the moving and fixed images. For this resolution level, a first rough estimate of the transformation matrix is found. Next at the finer resolution, we use as a starting guess the transformation matrix obtained from the previous lower resolution step and refine it based on the finer anatomical details appearing at the finer resolution. The procedure is typically repeated at three levels i.e., factors of four, two, and one times the image resolution. Instead of changing the image resolution, one can use different sizes of Gaussian kernels to introduce blurring which has the same effect. In the exercise, we will use the pyramid step, and one must set the number of levels in the pyramid. Further, for each resolution level, one set the down-sampling factor, the number of iterations for the optimiser, and the size of the Gaussian blurring kernel.

When we finally have found the optimal affine matrix for the moving image it would be nice to be able to inspect the individual transformations i.e., translation, rotation, scaling, and shearing parameters. This is not possible by inspecting the affine matrix outputted by the registration pipeline where the different transformations are multiplied. Therefore, we must decompose the affine matrix back into the different geometrical transformation matrices using a second numerical optimizer algorithm. The Python library we use offers such decomposing possibility of the affine matrix so we easily can evaluate if the transformation parameters are as expected.

The world-coordinates system versus the grid system
In most registration programs one just gives as input the fixed and the moving image and asks to find the optimal geometrical transformation matrix. This is most often the case for 2D image registration methods. However, in 3D registration and especially in medical image registrations an extra set of affine matrices are used for both the moving and the fixed image. In the exercise, we ignore these matrices by setting them to an identity matrix. The idea of the two transformation matrices is to convert between two coordinate systems being the word-coordinate and the image data coordinate system. The world-coordinates system describes a reference system of the image in millimeters how the MR image of a person's head is positioned in relation to the coordinate system of the MR scanner. The image coordinate system expresses data indices of the MR image matrix - also references as indices to a data grid. In image registration, the affine matrix describing the world-coordinate system can be

$$A_{world} = \begin{bmatrix} V_x & 0 & 0 & O_x \\ 0 & V_y & 0 & O_y \\ 0 & 0 & V_z & O_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

where the diagonal ($V_{x\text{-}z}$) is the same as the scale transformation matrix in Eq (5) but describes the voxel size in millimeters, and $O_{x\text{-}z}$ is the origin of the image. Off-diagonal values are often non-zero if the head is tilted in relation to the coordinate system of the MRI scanner. Naturally, there exists an affine world-coordinate matrix for each image i.e., one for the moving and one for the fixed.
The affine matrix describing the image data grid is simply the geometrical transformation between the moving and the fixed image that we are looking for (Eq 6). The affine transformation for the world-coordinate system of a 3D image is often stored in the header information of the image that most registration programs automatically read which includes the program we are using in the exercise.