Image Registration

Method Review

REGISTRATION

What data are we working with?

- Source image: image we want to register
- Target image: image to register TO

There are two flavors of registration:

- **Intensity based**: match images based on minimizing differences between intensity values. You can do this for the entire image, or a sub-image. If you use a sub-image, you can then use corresponding sub-images as points of input for a "feature based" method (below).
- **Feature based:** matched images based on corresponding points, lines, angles, the "features"

For both methods, the "output" of the algorithm is a transformation, or a mapping from one image to the other, and this mapping is commonly created with an algorithm that is based on calculating similarity metrics between the two images.

HOW DO WE MODEL THE TRANSFORMATION?

Linear transformations: include rotation, scaling, translation, and other affine transforms. "Affine" means that we preserve straight lines, and the ratios of distances between points on those lines.

Non-linear: include elastic or non-rigid transforms. If you are doing anything that does not preserve straight lines and or the ratios of distances, it falls in this bucket.

Methods can also work with intensity values or features (spatial information) or the frequency domain (eg, the frequency Fourier transform), and can be human-assisted or automatic. In imaging, it is ideal to have an automatic method that can handle multi-modal data, meaning images with differently defined or scaled intensity values.

AFFINE REGISTRATION

Simple **linear registration** is common in medical imaging analysis, so it will be reviewed first. As a reminder, **affine** means that we preserve straight lines, and the ratios of distances between points on those lines.

It's easiest to think of an affine registration as the combination of two things, **a translation** (adding or subtracting distance in a direction), and a **linear map** (multiplying by some value to scale the data).

the new value, the "transformed" data

Matrix multiplication applied to old data to scale it

$$\vec{y} = f(\vec{x}) = A\vec{x} + \vec{b}.$$

Is a function of the old values

and a translation

Algebraically, we will represent the **translation** with vector addition and subtraction, and the **linear map** with matrix multiplication. So the complete translation to map one image to another will be to first multiply values, and then add or subtract from the result of that.

HOW DO I FIND THE TRANSLATION?

the new value, the "transformed" data

Matrix multiplication applied to old data to scale it

$$\vec{y} = f(\vec{x}) = A\vec{x} + \vec{b}.$$

Is a function of the old values

and a translation

Given two images, A and B, that we want to register, we therefore want to find the **linear map** (how to multiply values in image A) and the **translation** (what to add or subtract from each value) to best make them look like values in B). Therefore, we need to find the matrix A, and the vector b. How do we do this?

First, let's better understand how we can model the translation above in one, handy dandy matrix! This will simplify the optimization problem.

THE TRANSLATION, PIECEWISE

If we have a column vector of one coordinate, x, y, and z:

The coordinate

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

We might want to scale, skew, or translate, represented by multiplying by these matrices:

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

scale

$$\begin{bmatrix} 1 & \lambda_x & 0 & 0 \\ \lambda_y & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

shear or skew

$$\begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

translation

These can be modeled in one matrix, the affine transformation matrix, usually referred to as A. We will append a "1" to the bottom of each coordinate column, this is called homogenous coordinates [representing a 2-vector (x,y) as a 3 vector (x,y,1)

MODEL THE TRANSFORMATION IN ONE MATRIX

Matrix multiplication is pretty fantastic. We can actually model our transformation in one matrix, the **affine transformation matrix:**

$$\begin{bmatrix} \vec{y} \\ 1 \end{bmatrix} = \begin{bmatrix} A & \vec{b} \\ 0, \ldots, 0 & 1 \end{bmatrix} \begin{bmatrix} \vec{x} \\ 1 \end{bmatrix}$$

The matrix above is a simplification, showing only one coordinate we want to transform, x to y. To put this into a matrix from our original equation:

$$\vec{y} = f(\vec{x}) = A\vec{x} + \vec{b}.$$

All that we have done is added a row of 0's below the linear mapping, A, and a 1 to the bottom of our translation vector. If you think back to simple matrix multiplication (rows by columns) for each of our types of transformation…

MODEL THE TRANSFORMATION IN ONE MATRIX

$$\vec{y} = f(\vec{x}) = A\vec{x} + \vec{b}.$$

$$\begin{bmatrix} \vec{y} \\ 1 \end{bmatrix} = \begin{bmatrix} A & \vec{b} \\ 0, \ldots, 0 & 1 \end{bmatrix} \begin{bmatrix} \vec{x} \\ 1 \end{bmatrix}$$

This multiplication works as follows:

1. Each value of A first gets multiplied, element-wise, with each of the values x, our original data, which is equivalent to $A\vec{x}$.

2. When we reach the last element of each row, the vector b is always multiplied by the last element in the column, 1, so we are essentially adding the values of b, which is equivalent to $+\vec{b}$.

Here is an example of translating one coordinate with two dimensions, (x,y)

This is the matrix A

$$\begin{bmatrix} x_b \\ y_b \\ 1 \end{bmatrix} = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_a \\ y_a \\ 1 \end{bmatrix}$$

one coordinate

WHY DO WE HAVE A ROW OF 0s and 1?

The row of zeros and additional 1 in the affine transformation matrix s there for what seems like a technicality.

Regardless of the translation, the origin is always going to correspond to (0,0,0) , and so technically the multiplication in absence of this additional row could not be considered a translation, in which it is necessary that it be mapped to some other point. By adding an additional coordinate (the 1), we are increasing the dimension so that **our original image** is actually a **subset of our new**, higher dimension space.

The origin of the original image can now be found at (0,0,0,1)

THE FINAL AFFINE TRANSFORMATION MATRIX

If we load a .mat file into Matlab with an affine transformation matrix associated with mapping some 3D image A to image B, we will see something like this:

```
>> header.private

ans =

NIFTI object: 1-by-1
            dat: [91x109x91 file_array]
            mat: [4x4 double]
     mat_intent: 'Aligned'
           mat0: [4x4 double]
    mat0_intent: 'Aligned'
         timing: [1x1 struct]
        descrip: 'FSL3.3'
            cal: [0 9170]


>> header.private.mat

ans =

    -2     0     0    92
     0     2     0  -128
     0     0     2   -74
     0     0     0     1
```

For 3D brain imaging data, we have two matrices, an sform and a qform. The **sform** matrix is a translation to a standard space, and the **qform** is the translation from the original scanner coordinates.

HOW DO I FIND THE TRANSLATION?

Now that we know the format of our affine transformation matrix:

$$(X_{NEW}, Y_{NEW}) = \mathbf{A} \times (X_{OLD}, Y_{OLD})$$

It makes sense that we can easily solve for the values of matrix **A** by solving the linear equation above. We can then apply this matrix to whatever images (X_{OLD}, Y_{OLD}) we want to transform into our desired space.

vsochat@stanford.edu