Bonn-Aachen International Center for Information Technology
University of Bonn
Master Programme in Life Science Informatics
Master Thesis

Brain Tractography Registration with Nonrigid ICP

Mohammed Abdelgadir Awadelkareem Hassan

Supervisor: Prof. Thomas Schultz
Bonn-Aachen International Center for Information Technology
University of Bonn

Supervisor: Dr. Martin Vogt
Bonn-Aachen International Center for Information Technology
University of Bonn

Date: February 11, 2019

ACKNOWLEDGMENT

I would like to start by expressing my sincere gratitude to Prof. Dr. Thomas Schultz for providing me an excellent opportunity to work on this project under his supervision.

Also, I would like to thank Mohammad Khatami for his valuable support throughout the course of the project.

I would like to extend my sincere thanks to Dr. Martin Vogt for his expert guidance and valuable input.

Last but not least, I express my very profound gratitude to all my friends and family especially my parents for always being supportive and encouraging throughout the course of my studies.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The registration problem (also known as alignment, absolute orientation) is one of the outstanding and very basic problems in computer vision. In this problem, two or more datsets of points are given and the task is to optimally align them by estimating a best transformation (combination of translation, rotation and scaling). Due to its fundamental importance, it arises as a subtask in many different applications (e.g., object recognition, tracking, range data fusion, graphics, medical image alignment, robotics and structural bioinformatics etc [5].
In This thesis we implement the method mentioned in [1], we show how to extend the ICP framework to nonrigid registration, while retaining the convergence properties of the original algorithm. The resulting optimal step nonrigid ICP framework allows the use of different regularisations, as long as they have an adjustable stiffness parameter. The registration loops over a series of decreasing stiffness weights, and incrementally deforms the template towards the target, recovering the whole range of global and local deformations. To find the optimal deformation for a given stiffness, optimal iterative closest point steps are used. Preliminary correspondences are estimated by a nearestpoint search. Then the optimal deformation of the template for these fixed correspondences and the active stiffness is calculated. Afterwards the process continues with new correspondences found by searching from the displaced template vertices. We present an algorithm using a locally affine regularisation which assigns an affine transformation to each vertex and minimises the difference in the transformation of neighbouring vertices. It is shown that for this regularisation the optimal deformation for fixed correspondences and fixed stiffness can be determined exactly and efficiently. The method succeeds for a wide range of initial conditions, and handles missing data robustly. It is compared qualitatively and quantitatively to other algorithms using synthetic examples and real world data [1].

1

INTRODUCTION

1 BRAIN STRUCTURE (FIBER PATHWAYS)

Human brain which is our focus in this thesis, is the central organ of the human nervous system, it is made up of two main components, gray matter and white matter. Scientists have learned a lot about gray and white matter and the two halves of the brain through autopsies and imaging techniques and by studying diseases or conditions associated with brain damage.
The thesis focused on **White Matter** which is refers to areas of the central nervous system that are mainly made up of myelinated axons, also called tracts or fiber pathways [6]. It is composed of bundles, which connect various gray matter areas of the brain to each other, and carry nerve impulses between neurons. Myelin acts as an insulator, which allows electrical signals to jump, rather than coursing through the axon, increasing the speed of transmission of all nerve signals [7].
Long thought to be passive tissue, white matter affects learning and brain functions, modulating the distribution of action potentials, acting as a relay and coordinating communication between different brain regions [8].

The Human Brain consits of these tracts in left and right side:

- Anterior Thalamic Radiation (ATR)
- Corpus Callosum (CC)
- Genu of the Corpus Callosum (genu)
- Splenium of the Corpus Callosum (splenium)
- Body of Corpus Callosum (truncus)
- Body of Corpus Callosum (truncus)
- Cingulum (Cing)
- Corticospinal Tract (CST)
- Inferior Fronto-occipital Fasciculus (IFO)
- Inferior Longitudinal Fasciculus (ILF)
- Superior Longitudinal Fasciculus (SLF)

- Ventral Tegmental Area (VTA)

2 REGISTRATION

The registration problem (also known as alignment, absolute orientation) is one of the outstanding and very basic problems in computer vision. In this problem, two or more datsets of points are given and the task is to optimally align them by estimating a best transformation (combination of translation, rotation and scaling). Due to its fundamental importance, it arises as a subtask in many different applications (e.g., object recognition, tracking, range data fusion, graphics, medical image alignment, robotics and structural bioinformatics ... etc) [5].

2.1 *Iterative closest point (ICP)*

ICP, which is an algorithm employed to minimize the distance between two or more points clouds, is one of the widely used algorithms in aligning three dimensional models given an initial guess of the rigid body transformation required [9]. In ICP (in our case) one points cloud (vertex cloud), the reference, or target, is kept fixed, while the other one, the source, is transformed to best match the reference. The algorithm iteratively revises the transformation (combination of translation, rotation and scaling) needed to minimize a distance from the source to the reference points cloud.

3 PCA TRANSFORMATION

PCA is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by some projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on[10]. PCA is used in the code as a preliminary step, so that the template and target are aligned as much as possible before the registration can begin.

4 LEAST SQUARES (LSQR)

LSQR is iterative method for solving $Ax = b$ and $min||Ax - b||^2$, where the matrix A is large and sparse. The method is based on the bidiagonalization procedure of Golub and Kahan. It is analytically equivalent to the standard method of conjugate gradients, but possesses more favorable numerical properties

[11]. It is used in this thesis to solve the cost fuction which we will discuss in the next chapter in details.

2

METHOD

1 THE ICP FRAMEWORK

The main subject of this thesis is demonstrating how the ICP framework can be extended to nonrigid registration, whilst retaining the convergence properties of the original algorithm. The principle idea is the application of the work presented in [1]. The resulting optimal step nonrigid ICP framework allows for the use of different regularisations, as long as they have an adjustable stiffness parameter. The registration loops over a series of decreasing stiffness weights and incrementally deforms the template towards the target, recovering the whole range of global and local deformations. To find the optimal deformation for a given stiffness, optimal iterative closest point steps are used. Preliminary correspondences are estimated by a nearest point search. Subsequently, the optimal deformation of the template for these fixed correspondences and the active stiffness is calculated. Afterwards, the process continues with new correspondences found by searching from the displaced template vertices. We present an algorithm using a locally affine regularisation which assigns an affine transformation to each vertex and minimises the difference in the transformation of neighbouring vertices. It is shown that for this regularisation, the optimal deformation for fixed correspondences and fixed stiffness can be determined exactly and efficiently. The method is successful for a wide range of initial conditions, and handles missing data robustly. Furthermore, it is compared qualitatively and quantitatively to other algorithms using synthetic examples and real world data.

As has been defined in the introduction, vertex registration is a problem in which two or more datasets of points are given and the task is to optimally align them by estimating a best transformation. In our case, we use a dense registration method to find a mapping from each point in the template onto the target while sparse methods find correspondence only for selected feature points. This is done by deforming the template, locally moving it closer in each iteration to the target in order to wrap them together with respect to stiffness. ICP moves the template S towards the target T step by step. In each iteration, it minimizes the difference between the template S and the target, T as illustrated in figure {1}, to reach the minimal value by solving the main equation (1):

$$||Ax - b||^2 \tag{1}$$

In equation (1) x is a list of X_i, each X_i is a 3×4 affine matrix which uses homogeneous coordinates $[x, y, z, 1]$ in 3D Euclidean space. By stacking X_i together we get $4n \times 3$ matrix x as shown in equation (2):

$$X = [X_1, X_2, ..., X_n]^T \tag{2}$$

If we were to simply solve the equation (1) by assigning A as the template and b as the target, we will get exactly $Ax = b$, which means the difference between them is zero. That leads to the deforming of A and a complete loss of its shape, therefore, we need to add the stiffness part to the equation that will prevent this deformation by keeping the vertices originally as close to each other as possible which we will later explain in this chapter.

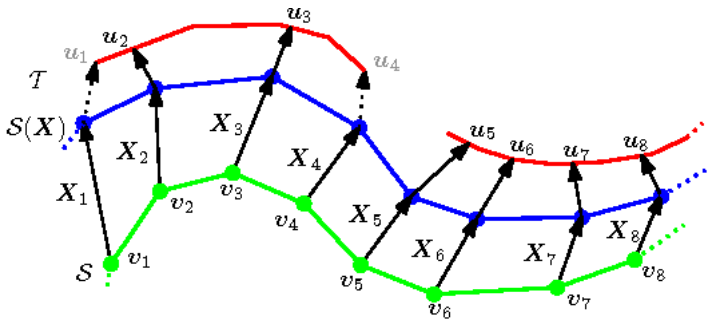


Figure 1: The template graph S (green) is deformed by locally affine transformations (X_i) onto the target graph T (red). The algorithm determines closest points (u_i) for each displaced source vertex $(X_i v_i)$ and finds the optimal deformation for the stiffness used in this iteration. This is repeated until a stable state is found. The process then continues with a lower stiffness. Due to the stiffness constraint the vertices do not move directly towards the target graph, but may move parallel along it. The correspondences u_1 and u_4 are dropped as they lie on the border of the target [1].

2 SETUP DATA

The method presented in [1] deals with graph representations of the data $S = (V, E)$ where S, V and E are assigned to graph, vertices and edges, respectively. As described in the introduction, our data contains human brain fiber bundles (pathways) saved in a ***ply*** data format, as shown in the simplified figure {2}. The ***ply*** format consists of three parts; the first, uppermost part is the header of the file which contains the file description (i.e. format, comment, etc) and the major components of the header element parts which describe how many elements each part has. In the sample in figure {2}, there are 69283 vertices in x,y,z and 603 fibers. The second part of the ***ply*** files contains vertices in 3D Euclidean coordinate space while the last part of the ***ply*** file represents the end

index of each fiber. As the method in [1] require the template to be in graph format, we develop a tool to read ***ply*** files into a numpy array of arrays which can subsequently be used as a graph. To be able to use ***numpy ndarray*** as a graph, we put each tract in a separate array with respect to the link between points (Edges E). To do so, we put each point linked together next to each other, and then wrap all the tracts belonging to the same bundle (i.e ATR, CC, genu, splenium, etc) in a new ***numpy ndarray***.

```
ply
format ascii 1.0
comment DTI Tractography, produced by fiber-track
element vertices 69283
property float x
property float y
property float z
element fiber 603
property int endindex
end_header
-4.71338558197 -19.9100589752 4.76097154617
-4.73113059998 -19.3581771851 4.68979740143
-4.72901630402 -18.8120174408 4.5764541626
-4.79067516327 -18.2503032684 4.52395439148
......
152
299
364
494
637
767
```

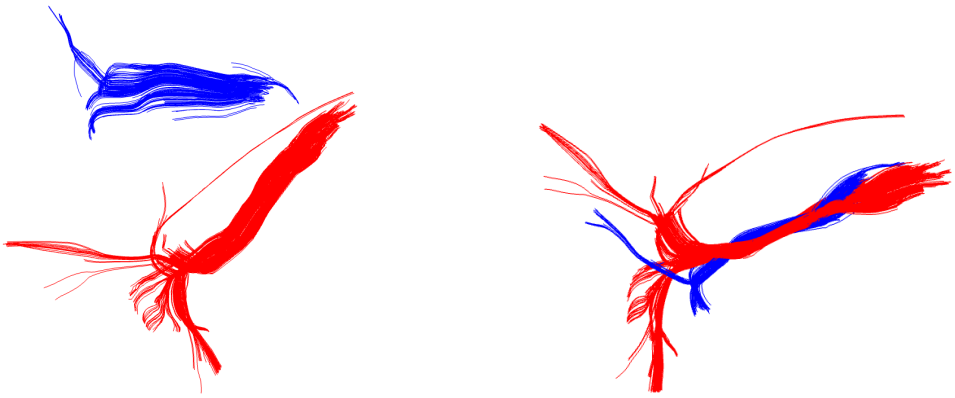
Figure 2: Simplified ***ply*** file sample

3 PREPARATION STEPS

After we read the data in graph format, we must align the template graph to the target graph as closely as possible. In order to do so, we use *Principal Components Analysis (PCA)*. Before applying PCA, we scale the template graph and the target graph to a $[0, 1]$ scale to match each other. This is done by subtracting all points from the minimum value in the graph and then dividing them by the maximum value in the graph. Next, we apply PCA and use $[-1, 1]^3$ cube combination and measure the distance between each point in the template graph to the closest point in the target graph $\sum ||v_i - v_j||_F^2$ where $v_i \in S, v_j \in T$ eight times to select the combination with the minimum distance as is illustrated in table {1}.

1	2	3	4	5	6	7	8
(x,y,z)	(x,y,-z)	(x,-y,z)	(x,-y,-z)	(-x,-y,z)	(-x,y,-z)	(-x,y,z)	(-x,-y,-z)

Table 1: $[-1, 1]^3$ cube combination



(a) Original position (b) After PCA

Figure 3: PCA alignment

4 DISTANCE MEASUREMENT

In all process in our code we use K-D tree (or k-dimensional) to measure the euclidean distance between the points, either on PCA transformation, ICP or any where we measure the distance.

4.1 *The K-D tree*

The K-D tree is a space-partitioning data structure (binary tree) for organizing points, every leaf node is a k-dimensional point. Every non-leaf node can be thought of as implicitly generating a splitting hyperplane that divides the space into two parts, known as half-spaces. Points to the left of this hyperplane are represented by the left subtree of that node and points to the right of the hyperplane are represented by the right subtree. The hyperplane direction is chosen in the following way: every node in the tree is associated with one of the k dimensions, with the hyperplane perpendicular to that dimension's axis. So, for example, if for a particular split the "x" axis is chosen, all points in the subtree with a smaller "x" value than the node will appear in the left subtree and all points with larger "x" value will be in the right subtree. In such a case, the hyperplane would be set by the x-value of the point, and its normal would be the unit x-axis [12].

5 FINDING THE AFFINE TRANSFORMATION

After we prepare our data, we refer back to [1] and determine how we need to build the cost function. As we are going to explain, it consists of three parts: distance term, stiffness term and landmark term.

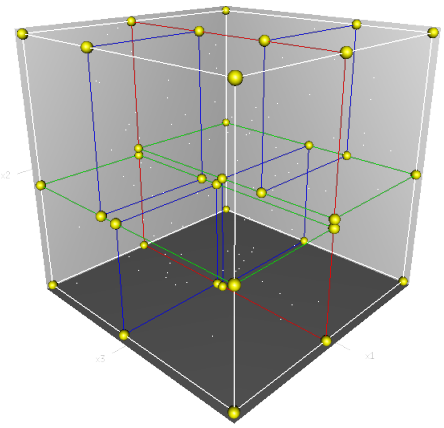


Figure 4: A 3-dimensional k-d tree. The first split (the red vertical plane) cuts the root cell (white) into two subcells, each of which is then split (by the green horizontal planes) into two subcells. Finally, four cells are split (by the four blue vertical planes) into two subcells. Since there is no more splitting, the final eight are called leaf cells [2].

5.1 *The original cost function:*

As we mention above, the cost function as it shown in equation (3) consists of three terms, each of which represent different concepts: the distance term, stiffness term and landmark term. The distance term represents the distance between the template graph S and target graph T which has to be as minimal as possible (i.e., which must be minimized). The stiffness term regularizes the template graph to prevents it from being exactly the target graph. The final term is the landmark term which gives the initial case for the template graph.

$$E(X) = E_d(X) + \alpha E_s(X) + \beta E_t(X) \tag{3}$$

Let us illustrate the first term, **the distance term**. As we mention before it is euclidean distance between points in template graph S and points in target graph T, which can be written as represented in equation (4), this distance has to be minimized as much as possible to align template graph S to target graph T with respect to the original shape of template graph S as much as possible, that means we need to make sure (by using stiffness term, which we will discuss later in this section) that all points which close to each other stay as possible as they are.

$$E_d(X) = \sum_{v_i \in V} w_i dist^2(T, X_i v_i) \tag{4}$$

In equation (4), v_i represent the vertices V in the template graph S in homogeneous coordinate $v_i = [x, y, z, 1]^T$, as we mentioned before, to be able to multiply them with affine matrices X_i which are 3×4 matrices including transformation part which illustrated in equation (1). The distance between each vertex $v_i \in V$ in the template graph S and it is closest vertex $u_i \in U$ in target graph T noted

with $dist^2(T, X_i v_i)$, and $w \in [0,1]$ is weight which is *one* if the correspondence for this vertex v_i is found in the target graph S, and *zero* if not. This concept will let each vertex v_i in the template graph T moves toward its correspondence vertex u_i in the target graph T if it found, otherwise, it moves with its neighbors because of the stiffness. Simply we use distance threshold as parameter to distinguish correspondances.

Now we come to the second term, **the stiffness term**, which regularizes the shape deformation, represented in equation (5):

$$E_s(X) = \sum_{i,j \in E} ||(X_i - X_j)G||_F^2 \tag{5}$$

Regularizing the deformation occurs by penalising the weighted difference of the transformations of neighbouring vertices under the Frobenius norm $||.||_F$ using a weighting matrix $G := diag(1,1,1,\gamma)$. γ can be used to weight differences in the rotational and skew part of the deformation against the translational part of the deformation as it mentioned in [1]. The value of parameter γ depends on the units of the data, and on the type of deformation that shall be expressed. In our case we choose it to be one because we already scaled our data into the $[-1,1]^3$ cube.

As it mentioned in [1] and in this thesis as well we use twelve parameters per vertex in 3×4 shape. This will allow us to write the cost function as a quadratic function which can be solved directly.

The Frobenius norm in equation (5) defined for matrix A as demonstrated in equation (6):

$$||A||_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2} \tag{6}$$

By applying this equation we keep the vertices, which linked together (there is an edge between them, they are in the same tract) or neighbors, close to each other.

The α parameter in equation (3) is a constant that manages the effect of the stiffness term, when it is high the neighbor vertices stay close to each other and vice versa, when it is low, more deformation can occur.

The final part of equation (3) is **Stiffness term** which demonstrated in bellow in equation (7):

$$E_l(X) = \sum_{(vi,l) \in L} ||(X_i v_i - l)||^2 \tag{7}$$

Refer back to [1], it mention that landmark uses for initialization and guidance of the registration. Given a set of landmarks $L = \{(v_{i1}, l_1), (v_{i2}, l_2), ..., (v_{il}, l_l)\}$

mapping template graph S vertices V_i into the target graph T vertices U_i, the landmark weight β in equation (3) is used to fade out the importance of the potentially noisy landmarks towards the end of the registration process.

The registration can be found even without landmarks. Without landmarks the cost function has global minima where the template is collapsed onto a point on the target surface, but the local minimum corresponding to the correct registration can be found for a wide range of initial conditions.

In our case, we use PCA transformation as initial step to align two graphs, therefore, we omit the landmark term later on from our cost function, this consider to be a first optimization to the cost function due to reduction of calculation effort and time the landmark term in our code.

Let us go deeper to illustrate the cost function more, and put the graphs in matrices that suit our cost function to ease implementation of the method.

Distance term in equation (3) and (4) become:

$$\bar{E}_d(X) = \sum_{v_i \in V} w_i ||X_i v_i - u_i||^2 \tag{8}$$

$$= \left\| (W \otimes I_3) \left(\begin{bmatrix} X_1 & & \\ & \ddots & \\ & & X_n \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} - \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix} \right) \right\|^2$$

where $W = diag(w_1, \ldots, w_n)$ is corresponded weight in diagonal matrix multiplied by Kronecker product $\otimes$ to I_3 identity matrix, X is diagonal matrix of X_i that we need to solve multiplied by V which is vertices of template graph S subtracted by the (corresponded) vertices U of the target graph T.

the **Kronecker product**, denoted by $\otimes$, is an operation on two matrices of arbitrary size resulting in a block matrix. It is a generalization of the outer product (which is denoted by the same symbol) from vectors to matrices, and gives the matrix of the tensor product with respect to a standard choice of basis [13].

For instance, If A is an $m \times n$ matrix and B is a $p \times q$ matrix, then the Kronecker product $A \otimes B$ is the $mp \times nq$ block matrix:

$$A \otimes B = \begin{bmatrix} a_{11}B & \ldots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \ldots & a_{mn}B \end{bmatrix}$$

So, the result of $W \otimes I_3$ is $3n \times 3n$ matrix, where X matrix has size $3n \times 4n$, V has $4n \times 1$ shape and U has shape $3n \times 1$.

We continue reform the equation to be differentiated easily by putting it into canonical form. We swap the position of X and the V matrices. For V matrix

we define a sparse matrix D which contains the the vertices $v_i \in V$ in diagonal position as demonstrated below:

$$D = \begin{bmatrix} v_1^T & & & \\ & v_2^T & & \\ & & \ddots & \\ & & & v_n^T \end{bmatrix} \tag{9}$$

The new format of the equation become:

$$\bar{E}_d(X) = ||W(DX - U)||_F^2 \tag{10}$$

Where the matrix W has size $n \times n$, the matrix D has size $n \times 4n$ (vertices v_i are in homogeneous coordinates $[x, y, z, 1]^T$), X has size $4n \times 3$, and U doesn't change with shape $n \times 3$ (vertices u_i are in 3D coordinate).

We continue to reform **the stiffness term** which penalises differences between the transformation matrices (affine matrices) X assigned to neighboring vertices. The simplified form of stiffness term will be:

$$E_s(X) = ||(M \otimes G)X||_F^2 \tag{11}$$

where M is node-arc incidence matrix which defined for directed graph. The number of rows in this matrix equal the number of nodes (vertices) and the number of columns equal the number of arcs (edges) on the graph and the values has to be *zeros, ones and/or minus ones*. the value will be *zero* if the edge on this column not connected to the vertex on this row where the value lie, otherwise the value has to be either 1 or -1, it will be 1 if the edge direction coming to the vertex and -1 otherwise, it is illustrated in the figure {5}, the matrix M has size $e \times n$ where *eandn* are the number of edges and vertices respectively. $G = diag(1, 1, 1, y)$ is diagonal matrix and the result of $M \otimes G$ is a matrix with size $4e \times 4n$.

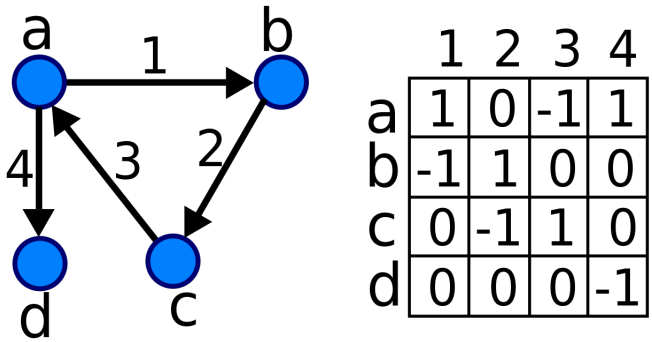


Figure 5: Oriented graph with corresponding incidence matrix [3]

Final term is **the landmark term** needs to be reformed as well, we reform it similar to distance term. We only consider those vertices from D (D *from the*

distance term (8)) are landmarks in a new matrix called D_L, and vertices from U (which are vertices from target graph T) are corresponded to those landmarks and put them in a new matrix called $U_L = [l_1, l_1, \ldots, l_l]^T$.

The final shape of the cost function is quadratic function as shown below:

$$\bar{E}(X) = \left\| \begin{bmatrix} \alpha M \otimes G \\ WD \\ \beta D_L \end{bmatrix} X - \begin{bmatrix} 0 \\ WU \\ U_L \end{bmatrix} \right\|^2 = ||AX - B||_F^2 \tag{12}$$

The current form of the function can be minimized by setting its derivative to zero and solve it as linear equation. The minimum value of $\bar{E}(X)$ is when $X = (A^T A)^{-1} A^T B$. Even the A is a sparse matrix and most of its values are zeros, it is still large matrix and it takes the most time of solving the equation. To solve this problem, we will remove the landmark term as mentioned before, because we already have the initial step by applying PCA, after we omit the landmark term the function will look like this:

$$\bar{E}(X) = \left\| \begin{bmatrix} \alpha M \otimes G \\ WD \end{bmatrix} X - \begin{bmatrix} 0 \\ WU \end{bmatrix} \right\|^2 = ||AX - B||_F^2 \tag{13}$$

6 APPLYING THE TRANSFORMATIONS

Let us review the principal of affine transformation briefly. If we have a point $[x, y, 1]$ in homogeneous coordinate in 2D euclidean space and we want to move this point three unit through the X *axis*, $[x, y, 1]^T a = [x + 3, y, 1]^T$, in this case a is called affine matrix.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x+3 \\ y \\ 1 \end{bmatrix}$$

The example mentioned above is only for translation, we have still some other moves (i.e. sheer, rotation and scale) as demonstrated in figure {6} for 2D shape.

We have shown that how 2D transformation occurs, and the same applies to 3D as illustrated below:

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \\ 1 \end{bmatrix} \begin{bmatrix} a & b & c & e \\ f & g & h & i \\ j & k & l & m \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

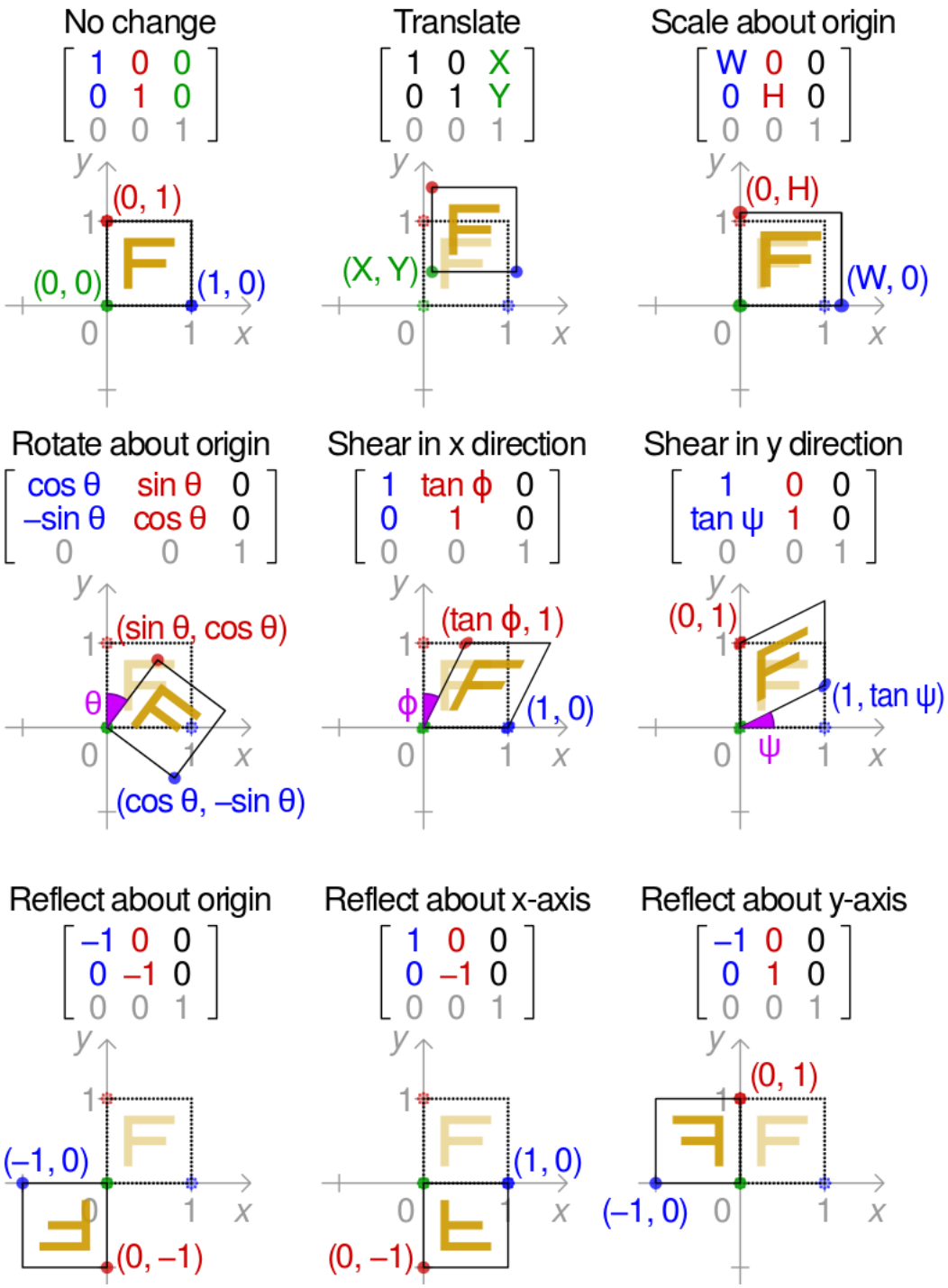


Figure 6: 2D affine transformation matrices [4]

But this is still on point, if we have more points like our case, the equation above become:

$$\begin{bmatrix} \hat{x_1} & \hat{x_2} & \dots & \hat{x_n} \\ \hat{y_1} & \hat{y_2} & \dots & \hat{y_n} \\ \hat{z_1} & \hat{z_2} & \dots & \hat{z_n} \\ 1 & 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} a & b & c & e \\ f & g & h & i \\ j & k & l & m \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \\ z_1 & z_2 & \dots & z_n \\ 1 & 1 & \dots & 1 \end{bmatrix}$$

7 NONRIGID OPTIMAL ICP ALGORITHMS

As we mentioned in the introduction, ***Iterative Closest Point (ICP)*** is an algorithm employed to minimize the distance between two or more points clouds, to achieve this goal, we use ***Sparse Least Squares algorithm (LSQR)*** which is iterative method for solving $Ax = b$ and $min||Ax - b||^2$, where the matrix A is large and sparse (as in our case). The method is based on the bidiagonalization procedure of Golub and Kahan. It is analytically equivalent to the standard method of conjugate gradients, but possesses more favorable numerical properties [11].

We will demonstrait the idea of using *lsqr* to solve our cost function in the next chapter implementation.

3

IMPLEMENTATION

To implement the method we have discussed before, we wrote the code in *Python Programming Language* and *Spyder* version 3 and *Integrated Development Environment (IDE)* are used.

As we mentioned before, our data is in ***ply*** format as shown in figure {2}, to read this format we wrote our own method using *plyfile* library due to special case of our data which has different arrangment files. The method we wrote return data in *numpy ndarray* structure where each array tract is putted in an array and all tracts belong to the same bundle were putted together with respect to the sequences of the neighboring vertices.

Then we apply PCA transformation (*sklearn.decomposition* library) to the data and generate histogram (*matplotlib.pyplot* libray) for distances between each vertex in the template graph S and its closet vertex in target graph T, before and after PCA transformation to compare the distance difference, because in some cases where the two graphs (template and target) are already in the best alignment before ICP, PCA transformation increase the distance between them as we will show in the result chapter. If the distance between two graphs is increased we drop the PCA transformation step. For distance measurement we use *K-D tree* from *scikit learn*.

We continue to build variables as shown in equation (13), to do so, we use histograms generated in the previous step to determine the threshold for the maximum distance we need to consider for W in the **distance term** for cost function, if the value is equal or below the threshold $w_i = 1$, otherwise $w_i = 0$. Then we build W as diaginal sparse matrix using *scipy.sparse* library. Then we continue using the same libray (*scipy.sparse*) to build D sparse matrix and calculate WD and WU.

Now we have **the distance term** of the cost function, we need to build the **stiffness term**, we use the same libray (*scipy.sparse*) to build M sparce matrix and G diagonal sparse matrix and calculate Kronecker product ($M \otimes G$).

The final step of building the cost function is to virticaly stack MG and WD to have A, and virticaly stack *zero* sparse matrix and WU to have B as require $||Ax - b||_F^2$.

The last step is solving the cost fuction $||Ax - b||_F^2$ by using ***LSQR*** from *scipy.sparse.linalg* library. As mentioned in the *scipy.sparse.linalg.lsqr* documentation [14], b in quation $Ax - b$ must be a vector, and as b in our case is a matrix size $n \times 3$, we use matrices fundamental concept to solve this problem as we use **lsqrt** three times for each column and horizontaly stack the result to get the affine matrices combination.

Finally for visualization we customize functions from *open3d* library to view brain bundles due to special case of our data.

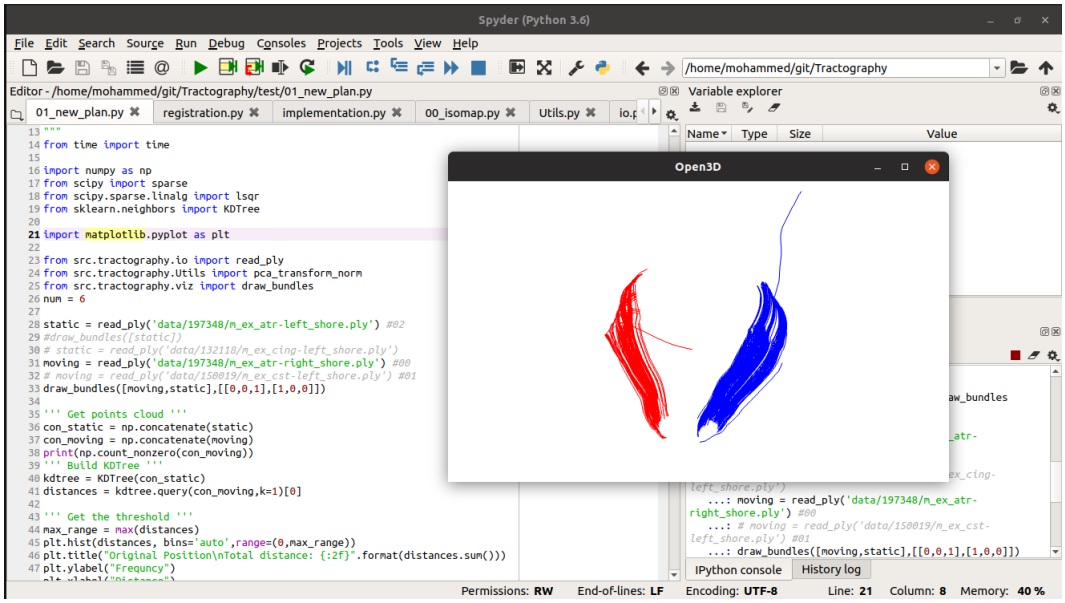


Figure 7: IDE and visualization tool

4

RESULT

5

CONCLUSION

BIBLIOGRAPHY

[1] Brian Amberg, Sami Romdhani, and Thomas Vetter. Optimal step nonrigid icp algorithms for surface registration. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[2] Wikipedia, the free encyclopedia. 3d tree, February 2006. [Online; accessed Feb 9, 2019].

[3] Wikipedia, the free encyclopedia. Oriented graph with corresponding incidence matrix, 2010. [Online; accessed Feb 9, 2019].

[4] Wikipedia, the free encyclopedia. 2d affine transformation matrices, 2016. [Online; accessed Feb 10, 2019].

[5] Hongdong Li and Richard Hartley. The 3d-3d registration problem revisited. In *2007 IEEE 11th International Conference on Computer Vision*. IEEE, 2007.

[6] Hal Blumenfeld. *Neuroanatomy through Clinical Cases*. Oxford University Press, 2010.

[7] Stephen B. Klein. *Biological Psychology*. Worth Publishers, 2008.

[8] R. Douglas Fields. White matter matters. *Scientific American*, 298(3):54–61, mar 2008.

[9] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2):119–152, oct 1994.

[10] I.T. Jolliffe. *Principal Component Analysis (Springer Series in Statistics)*. Springer, 2002.

[11] Christopher C Paige and Michael A Saunders. Lsqr: An algorithm for sparse linear equations and sparse least squares. *ACM Transactions on Mathematical Software (TOMS)*, 8(1):43–71, 1982.

[12] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[13] Wikipedia contributors. Kronecker product, February 2019. [Online; accessed 9-February-2019].

[14] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001. [Online; accessed February 11th, 2019].