Feature-Accelerated Iterative Kabsch – A Hybrid Approach to Partial-to-Partial Volume Registration

E4040.2022Fall.VOCT.report
Brian L. Frost bf2458
Columbia University Department of Electrical Engineering

***Abstract*—Optical coherence tomography (OCT) is a powerful modality capable of both volumetric imaging and angstrom-scale vibrometry. It is used by researchers studying cochlear mechanics to measure micromechanical motions in the organ of Corti – the sensory tissue responsible for transduction of pressure waves into neuronal stimulus. OCT measurements are uniaxial, meaning a single motion measurement only measures a projection of the true 3-D motion onto the measurement axis. To assess the complete 3-D motion patterns of a particular structure, measurements can be taken from multiple angles and backprojected to reconstruct the total motion. Doing so requires the ability to register measurement positions after rotation and translation of the volume, moreover introducing occlusion due to structures blocking light differently at different angles. In the present work, I provide an analysis of three neural network solutions to this registration problem – one based on the convolutional ResNeXt architecture, a second using operations on nearest neighbors rather than standard convolutions, and a third method that uses this neural network output to initialize the classical iterative Kabsch algorithm. Testing on synthetic data, I find that this hybrid algorithm, termed Feature-Accelerated Iterative Kabsch (FAIK), shows incredible prediction power for the registration problem in the median case. The other presented methods outperform FAIK on certain metrics, and each method may be preferable in different applications.**

I. Introduction

Optical coherence tomography (OCT) is an imaging modality capable of volumetric imaging with micron-scale resolution [1]. An OCT image is formed from A-Scans – one-dimensional maps representing reflectivity as a function of depth into the sample. Taking many A-Scans along one scanning axis forms an image (B-Scan), and taking many B-Scans along an orthogonal scanning axis yields a volume scan.

While the current clinical applications of OCT are mostly limited to ophthalmology, development of OCT catheters in recent years indicates its potential for pre-surgical probing of pathologies in the heart, and recent handheld OCT systems show potential for recognizing pathologies in the middle ear. It has also been widely adopted in research settings – for example, it is used to probe organ structures *ex vivo*.

For all such applications, the limited field of view (FOV) of OCT could necessitate volume stitching to arrive at full three-dimensional pictures of the structure of interest. This problem is analogous (but not equivalent) to taking a panoramic image with a standard camera. Work has been performed by Dr. Christine Hendon's group at Columbia University to perform such stitching in the *ex vivo* setting, wherein organs are scanned, then translated, then scanned again [2]. This means that individual volumes are related by a three-dimensional shift.

More generally, volume stitching may need to account for rotation of the sample as well – a six-parameter transformation that could arise in more complicated *in vivo* settings. This problem is of particular interest to researchers studying the mechanics of the cochlea – the mammalian organ responsible for the transformation of sound from a mechanical stimulus to a frequency-encoded neural response. On top of imaging, OCT is also capable of incredible angstrom-scale vibrometry, which allows researchers to probe the *in vivo* motion of intra-cochlear structures [3].

However, each motion measurement is taken only along the OCT's beam axis (the A-Scan axis), so that it is a one-dimensional projection of a three-dimensional motion. Recent work has shown interest in reconstructing all three components of motion, which requires measuring the same structure at multiple measurement angles [4]–[6]. To measure the same structure, points must be registered between the viewing angles – a registration between samples that have been both rotated and translated in three dimensions.

While OCT volumes are often referred to as reflectivity maps, this is not actually the case – as light reflects from surfaces nearer to the imaging lens, less light can get to and reflect from structures deeper into the sample. This leads to *angle-dependent occlusion*, wherein structures in a volume will appear darker (or will not appear at all) if a very reflective structure is nearer to the lens along the imaging axis. This deeply complicates the problem of registration in the cochlea, which is encased in opaque bone. The problem is thereby one of *partial-to-partial* registration, wherein many structures apparent in one volume are not apparent in the other.

Work from the Fowler Memorial Laboratory at Columbia University has developed a physiology-based method for such registration, but this method is time-consuming and not easily generalized [5], [6]. Other groups have devised methods that provide useful results but are either extremely expensive and challenging to implement, or limited in scope [7]–[9]. Instead, we would like a fast, reliable and generalizable method that can be used in any organ given only a single OCT system.

In this work, I present three deep learning-based methods that rely on a) the convolutional ResNeXt architecture, used for full-to-full registration by Guo *et al.* [10], b) the edge convolution (EdgeConv) layer for point-cloud inputs proposed by Wang *et al.* [11], and c) a deterministic point cloud registration method known as the iterative Kabsch algorithm [12], [13].

First, I evaluated the ResNeXt model for partial-to-partial registration. I then compared this model to one with a similar architecture, but with convolution layers replaced by EdgeConv layers – I call this the RedgeNeXt network (a portmanteau of ResNeXt and Edge). I also developed a hybrid algorithm that uses RedgeNeXt to determine initial conditions for the Iterative Kabsch algorithm. I call this algorithm Feature-Accelerated Iterative Kabsch (FAIK).

Testing on partial-to-partial registration of randomly generated polynomial surface combinations, I found that FAIK had excellent prediction power for most tested volumes, but diverged massively in many outlier cases. I found that ResNeXt was more consistent, and did best in predicting the translation between two volumes. I found that RedgeNeXt was similarly consistent, and performed better than ResNeXt in predicting rotation angles. These properties indicate that each algorithm could be useful for different applications, and that combined and informed use of these three algorithms can yield excellent predictions.

II. Prior Work

The present study draws from three prior works. In this section, I will discuss them in turn, as well as their relevance to, and differences from, the problem at hand.

A. Multi-Stage ResNeXt

The work of Guo *et al* [10] which concerns full-to-full 3-D registration of rotated, translated volumes taken using different modalities. The inputs to their model are volumes and the outputs are six-vectors containing the 3-D translation vector and the three Euler angles of rotation.

Their architecture (shown in their Fig. 2) uses a multi-path ResNeXt as its backbone, wherein the volumes undergo convolutions, followed by 32 parallel residual blocks each containing 3 convolution layers. The outputs of these residual paths are added, and then concatenated with the volumes prior to passing through the residual block. Finally, they pass through a standard convolutional neural network consisting of two convolution layers and two dense layers.

This network is only one stage of their total network. As seen in their Fig. 4, they chain many such networks together, updating the volumes in between stages based on the previously determined transformation.

Their problem is posed differently from ours in that it concerns full-to-full registration, which is not subject to occlusion. This enviable feature means that each volume pair is more similar, and intuitively it would be easier to learn a mapping between the two. It also means that they can use an error metric known as surface registration error (SRE) wherein the average distance between points on the edge of a surface is computed. Error metrics based on structural similarity (e.g. SRE, mean point distance or volume correlation) are more reliable, but unfortunately are not well-posed in a partial-to-partial setting.

B. Edge Convolution

Wang *et al.* [11] have developed a layer for point clouds (sets of 3-D points, rather than volumes with intensity channels) known as EdgeConv (for edge convolution). Their generalized EdgeConv layer consists of three components: 1) for every point in the point cloud, its k nearest neighbors are found; 2) an operation is then performed between the original point and its nearest neighbors; 3) a pooling operation (such as sum, mean or maximum) aggregates the values from step 2.

One special case of such a layer would be a densely connected layer that acts only on the nearest neighbors. This would be analogous to convolution on a regular volume (i.e. not a point cloud) in that each point in the output is determined from only some local information in the point cloud. To describe this mathematically, suppose I have a 3-D point cloud containing N points: $X \in \mathbb{R}^{N\times 3}$. For each point x_i, we find the set of k indices of the nearest neighbors I_i. The set of such points is thereby $\{x_j\}_{j\in I_i}$.

Such a set exists for each point in the set $i = 1, 2, \ldots, N$, so the set of all nearest neighbors can be written as $R \in \mathbb{R}^{N\times k\times 3}$, where each $R_{i,j,:}$ is the j^{th} nearest neighbor of x_i. A dense layer of D weights per dimension ($3D$ total) is then applied to the list of neighbors for each point separately, producing an array $K \in \mathbb{R}^{N\times D}$.

Wang *et al.* use networks containing this layer to perform segmentation and classification of point clouds. While my task of interest is distinct in that it requires the comparison of two volumes, the representation of a volume as a point cloud is useful for the task at hand. Namely, due to the occlusion present in OCT images, the intensity channel is unreliable and obscures the interpretation of the volume. Instead, a point cloud represents a binary volume.

The method also helps with the partial-to-partial nature of my problem – global features of the volumes may be quite different, with large pieces of the sample potentially being visible in only one of the two volumes. Instead, EdgeConv considers local structures, some of which will certainly be maintained between the volumes at different orientations.

C. Iterative Kabsch

The Kabsch algorithm is an analytic method for finding the least-squares-optimal rotation and translation vector relating two point clouds with known correspondence. For a more general problem in which correspondence is not known *a priori*, this method is not viable.

Algorithms have been developed which apply the Kabsch algorithm iteratively – one makes a best guess for the correspondence, uses Kabsch to find the optimal rotation and translation between the point clouds, then updates the guess at the correspondence. While several authors have arrived at similar methods, e.g. Besl and McKay, their derivations are usually incomplete or difficult to follow [12]. Here, I will describe the practical steps of the method, and I defer the derivation to a comprehensive post on my blog [13].

Suppose we have two ordered 3-D point clouds $V \in \mathbb{R}^{N_V \times 3}$ and $X \in \mathbb{R}^{N_X \times 3}$. The clouds are ostensibly related by rotation by map R and translation by vector t, but as the correspondence between the clouds is unknown, the problem of finding the optimal guess for R and t is ill-posed. We make a naive guess at correspondence by assuming each point in V corresponds to its nearest neighbor in X. That is, we form the ordered point cloud $W \in \mathbb{R}^{N_V \times 3}$ such that the i^{th} row of W is given by

$$w_i = \underset{x_n}{\operatorname{argmin}} \, ||v_i - x_n||_2, \;\; n = 1, 2, \ldots, N_X \tag{1}$$

where v_i and x_n are the rows of V and X, respectively. Each row in W is also a row in X, so W is called the *closest set* to V in X, written $W = \mathcal{C}(V, X)$. The rows of V and W are corresponding points.

To find the optimal rotation and translation between V and W, we apply the Kabsch algorithm. The goal is to find the least-square optimal values of translation t and rotation R:

$$\tilde{R}, \tilde{t} = \underset{R,t}{\operatorname{argmin}} \sum_{i=1}^{N_V} ||(R[v_i] + t) - w_i||_2^2. \tag{2}$$

The rotation mapping R can be described in several ways, such as a rotation matrix, a set of Euler angles or a unit quaternion (versor). I have derived on my blog [13] that the optimal versor $\tilde{q}$ is the unit eigenvector corresponding to the maximum value of a matrix relating to the cross-covariance of the point sets. It can be show that the optimal rotation matrix can also be found via singular value decomposition of the cross-covariance matrix.

In either case, once $\tilde{R}$ is determined, $\tilde{t}$ is easily found. Calling the centers of mass of V and W, μ_V and μ_W respectively, we have:

$$\tilde{t} = \mu_W - \tilde{R}[\mu_V]. \tag{3}$$

The iterative algorithm begins by initializing some rotation and translation R_0 and t_0, and initialize $W_0 = X$. Then, apply the following for each step k:

1) Let $V_k = R_k[V] * +t_k$.
2) Compute $W_k = \mathcal{C}(V_k, X)$.
3) Using this new corresponding set, compute the optimal rotation and translation R_{k+1} and t_{k+1} between V and W_k.

We run this for some number of iterations. It is guaranteed to converge monotonically, and under certain constraints performs very well. In particular, it does well if the rotation angles are relatively small (e.g. at most 30°), but is vulnerable to convergence to local minima when the rotation angles are large.

Table I shows the MSE (averaged over 100 samples of random $64 \times 64 \times 64$ polynomial point clouds, whose generation is described below) of iterative Kabsch for several maximum angles and translation when run for 1000 iterations. This shows the challenge this algorithm has in the case of large angles, but its incredible success for arbitrary translation at small angles.

Maxima	15°	30°	45°	80°
0 px	0	71.1	388	157000
10 px	0	70.0	544	219000
20 px	4.66	78.1	460	184000

TABLE I
MEAN SQUARE ERROR (UP TO THREE SIGNIFICANT FIGURES) FOR THE ITERATIVE KABSCH ALGORITHM APPLIED TO RANDOMLY GENERATED POLYNOMIAL VOLUMES, WITH MAXIMAL ROTATION ANGLE IN DEGREES AND MAXIMAL TRANSLATION IN PIXELS. ERROR IS DETERMINED BY AVERAGING OVER 100 REALIZATIONS RUNNING THE ALGORITHM FOR 1000 TIME STEPS. WE MULTIPLY TRANSLATIONAL ERROR BY 100 WHILE COMPUTING THE MSE, TO ACCOUNT FOR THE DIFFERENCE IN UNIT SIZE BETWEEN ANGULAR DIFFERENCE AND TRANSLATIONAL DIFFERENCE.

The accuracy does not vary much with translation, and as such, we will focus more on angle in this report.

As a note about the error – the algorithm tends to either converge with 0 error, or do a horrible job approximating the transformation. For example, at 30° maximum angle, the error is 0 for more than 50% of tested volumes, but in the tens of thousands for a few outliers. While in most applications, an MSE of 0 is not achievable, we see that iterative Kabsch may make it possible for data of this type, but only if its inconsistencies can be ameliorated. I take this into account when considering the efficacy of my models below.

III. Mathematical Formulation

Our problem concerns two OCT volume scans $V_1, V_2 \in \mathbb{R}^{X \times Y \times Z}$. The value of the volume at index $(x \; y \; z)$ is the intensity of the voxel. The volumes are related by 3-D rotation R, 3-D translation t and nonlinear intensity transformation T caused by occlusion. R can be defined by a matrix, a set of Euler angles, or a versor equivalently. We write

$$V_2 = T[R(V_1) + t]. \tag{4}$$

We want to find R and t, but not T. T is data-dependent, and its form would be hard to learn. Moreover, it is unnecessary for an experimenter to know T explicitly for the purpose of volume registration.

The volumes are very large, and their intensity contains difficult-to-interpret information. To solve these problems, we pre-process the volumes by downsampling to size $X' \times Y' \times Z'$, and thresholding, leaving us with smaller *binary* volumes W_1 and W_2. R and t are now unchanged, but T now acts in a deleterious and inserting fashion. That is, it can be seen as turning some 1s into 0s between the two volumes and vice versa.

We can represent these binary volumes in two ways. The first is as 3-D arrays $W_1, W_2 \in \mathbb{Z}_2^{X' \times Y' \times Z'}$, which are volumes containing 1s and 0s. The second is as point clouds $P_1 \in \mathbb{Z}^{N_1 \times 3}$ and $P_2 \in \mathbb{Z}^{N_2 \times 3}$, where the rows of P_i are the indices of W_i containing 1s. The mathematical relationship between these clouds is discussed in the section on the iterative Kabsch algorithm above.

The volume representation can be used as the input to a network such as ResNeXt, while the point clouds are suited to use with the EdgeConv block or the iterative Kabsch algorithm. As most of an OCT volume is usually black (0), the point clouds are usually smaller than their corresponding volume representations, which is a benefit in neural network training.

The two volumes in either representation are the inputs to our model. The outputs will be R and t represented in some decided fashion. We choose to represent the translation vector t as a three-vector and the rotation R in Euler angles – that is, the rotation angles about each of the Cartesian coordinate axes. Thus, our output is a vector $(\hat{\theta}\ \hat{\phi}\ \hat{\psi}\ \hat{t}_x\ \hat{t}_y\ \hat{t}_z) \in \mathbb{R}^6$, to be compared with the ground truth $(\theta\ \phi\ \psi\ t_x\ t_y\ t_z)$.

IV. Methods

In this section I will discuss the design and implementation of the network, as well as its training process. To begin, we discuss the synthetic data set used as the input to the network. Then, we discuss the network architecture and details of the implementation.

A. Training and Testing Data

My object of interest for this network is OCT volume scans. In the ideal scenario, I would have thousands of volume scans from different preparations with known rotational and translational relationships. This is untenable in the scope of this course project, as a) my volumetric scans have not historically been labeled with known rotations and translations, b) taking more volume scans in more preparations would require access to many organs in a short period of time, and c) volume scans are very large files.

Instead, I designed an algorithm that randomly generates volumes that model OCT volumes. These volumes comprise several three-dimensional polynomial surfaces with spatially varying thickness. The entire training set is randomly generated at train time, so giant OCT volume files need not be loaded into RAM (a common bottleneck for OCT volume processing).

The polynomial volumes are formed in the following way:

1) Randomly pick the number of surfaces to include in the volume as an integer between 1 and N.
2) Randomly pick orders of each polynomial separately as an integer between 0 (a constant function) and M.
3) Randomly choose the coefficients for these polynomials between -c and c. The graph of each polynomial defines a surface.
4) For each point on the surface, randomly assign a thickness t between 1 and T. Then, t pixels around the point are included in the binary volume. This volume (or corresponding point cloud) serves as the first input.
5) Randomly pick translation and rotation parameters $(\theta\ \ \phi\ \ \psi\ \ t_x\ \ t_y\ \ t_z)$, where some maximum constraints are imposed. This vector serves as the label.
6) Generate the second input by rotating and translating the first according to the vector defined above.

The result is a relatively complex set of surfaces that are not precisely polynomial due to their varying thickness. Each volume can be quite different in that they can contain a different number of surfaces of different orders and thickness profiles, giving us a varied data set that approximates an arbitrary OCT volume. An example of such a randomly generated polynomial volume is shown in Fig. 1. This representation is motivated by the fact that structures measured in OCT volumes often appear as surfaces, as light may not penetrate deeply into thicker tissues.

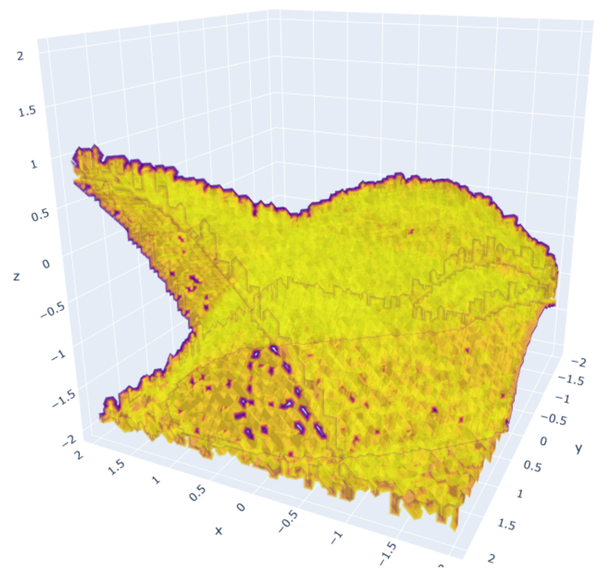


Fig. 1. An example of the volumes used for testing and training, which are randomly generated via adding the graphs of polynomials with random spatially-varying thicknesses.

In the point cloud representation, a different random sample of points is selected from the cloud in the original volume than in the rotated volume. This corresponds to the random insertion and deletion of points between volumes, modeling occlusion.

In my implementation, I use a maximum angle of 80° and a maximum translation of 9 px. I also choose $N = 3$, $M = 5$, $T = 5$, and $c = 1$. These parameters are chosen based on qualitative features of gerbil cochlea and heart volumes. We choose the point clouds to contain 200 points.

The test data is a set randomly generated polynomial surfaces as discussed above.

B. Architecture

We consider three architectures: 1) one-stage ResNeXt using the volume representation as the input; 2) RedgeNext using the point clouds as inputs; 3) FAIK, using the point clouds as inputs.

1) ResNeXt: I begin with a single stage of the ResNeXt network, which has been shown to succeed in full-to-full volume registration. As a baseline, I trained this network to probe it's performance in the partial-to partial case. I would like to emphasize that ResNeXt is not of my own design, and that I use it only as a baseline to which I compare my own models.

The architecture I have implemented is outlined in Fig. 2. The architecture relies on a residual block shown in Fig. 3**A**. The inputs are two $64 \times 64 \times 64$ volumes concatenated in the channel dimension, and the output is a 6-vector containing the Euler angles and the translation vector, $r = (\hat{\theta}\ \hat{\phi}\ \hat{\psi}\ \hat{t}_x\ \hat{t}_y\ \hat{t}_z)$.

The loss function applied is

$$L(r) = \frac{1}{Q}||\hat{t} - t||_2^2 + ||\hat{a} - a||_2^2, \tag{5}$$

where t is the translation vector, a is a vector containing the Euler angles, and Q is a hyperparameter that adjusts the proportions with which angular and translational error contribute to the total loss.

My particular architecture used: 1) $2 \times 2 \times 2$ windows for every pooling layer; 2) 128 units in each dense layer (save the output, which has 6 units); 3) 16 $5 \times 5 \times 5$ filters in the first convolutional layer; 4) 32 $3 \times 3 \times 3$ filters in the second and third convolutional layers; 5) and a residual path that sums 32 parallel paths of 3 convolution blocks, each containing 32 $3 \times 3 \times 3$ filters; 6) 16 $3 \times 3 \times 3$ filters in the last two convolutional layers. Relus are used as the activation functions of all layers except the output, which uses a linear activation function. This is simply because I am performing a linear regression.

2) RedgeNeXt: A similar architecture is used, with some significant distinctions. 1) All convolution layers are replaced with EdgeConv layers of the dense type described above; 2) the concatenation layer is replaced with a sum; 3) MaxPool layers are replaced with BatchNorm layers; 4) residual unity-gain paths are added between convolution layers; 5) inputs are concatenated in the batch domain rather than the channel domain. This architecture is shown in Fig. 4. It relies on the residual block shown in Fig. 3**B**.

The use of addition (rather than concatenation) and extra residual paths was due to an issue I noticed in training: without these additions, I saw that the weights were likely to converge to 0. These alterations, inspired by ResNet and the LSTM unit's methods for solving the vanishing gradient problem, were sufficient to avoid this local minimum at 0. The choice to not use max pooling and to use batch normalization is inspired by the work of Wang *et al* [11].

I made one other alteration to this architecture as compared to ResNeXt – instead of concatenating volumes spatially or in the channel domain at the input, I chose to concatenate the point clouds in the batch dimension. That is to say, I ran the first half of the network (up to the concatenation stage) on each volume separately, then concatenated and applied the second half of the network. This implementation felt more intuitive, as it means that the first half of the network will indiscriminately pick out the same sorts of important features from each volume, regardless of whether it is the target or base volume. Preliminary tests showed that this improved accuracy slightly.

All EdgeConv filters contain 16 units and act on the 5 nearest neighbors. The dense layers before the output each contain 256 nodes. The residual block contains 16 paths with 3 EdgeConv layers each. All layers use the Relu nonlinearity, except the output layer which has a linear activation function. This is simply because I am performing linear regression. The inputs are two 200×3 point clouds, and the output is a 6-vector containing the Euler angles and the translation vector, $r = (\hat{\theta}\ \hat{\phi}\ \hat{\psi}\ \hat{t}_x\ \hat{t}_y\ \hat{t}_z)$.

3) Feature-Accelerated Iterative Kabsch: The RedgeNeXt architecture is slightly altered so that it only outputs three values, $\tilde{a} = (\tilde{\theta}\ \tilde{\phi}\ \tilde{\psi})$, as a first guess at the Euler angles. These Euler angles, along with a translation of $(0\ 0\ 0)$, are used as the initial states in the iterative Kabsch algorithm applied to the two volumes, outputting $r = (\hat{\theta}\ \hat{\phi}\ \hat{\psi}\ \hat{t}_x\ \hat{t}_y\ \hat{t}_z)$. The choice to include only the angles in the outputs of the RedgeNeXt layer is to reduce the problem complexity, allowing a network of the same size to achieve higher accuracy. As iterative Kabsch's major failures occur only when the rotation angle (rather than the translation) is large, choosing to forego a first guess at the initial translation is not expected to reduce accuracy.

We train the RedgeNeXt layer by itself, as iterative Kabsch requires no training. The loss function for training the 3-output RedgeNeXt simplifies to

$$L(\tilde{a}) = ||\tilde{a} - a||_2^2. \tag{6}$$

C. Training

Two custom training loops were written. The first, used for the RedgeNeXt architecture (as well as for FAIK) is called `train` in the file `trainloops/polyGenTrainLoop.py`. The second, used for ResNeXt, is called `res_train` in `trainloops/res_polyGenTrainLoop.py`. The training loops are generally quite standard, comprising passes of multiple minibatches through the network and parameter update based on backpropagating the gradients of the loss functions described above. TensorFlow's automatic differentiation capabilities are used for this process. The only major difference between the two is that the former uses a point cloud representation of the volume and the latter uses a full volume representation.

I chose to use the Adam optimizer based on preliminary tests that indicated it would perform better than stochastic gradient descent. More work could be done in the future to determine which optimizer performs best.

1) Design of the Algorithm: For each minibatch, we generate a new dataset using the method of polynomial volume generation described above. The functions used here are contained in folder `dataGen` as files `randPoly.py` to generate the base polynomial volume, and `transRot.py` to generate the translated polynomial volume. Both take inputs that determine whether the volumes are represented as point clouds or as standard volumes.

An epoch is usually defined as a pass-through of the entire dataset. As our dataset is entirely randomly generated, an epoch cannot be defined. However, I have abused the terminology a bit by considering an epoch to be a large set

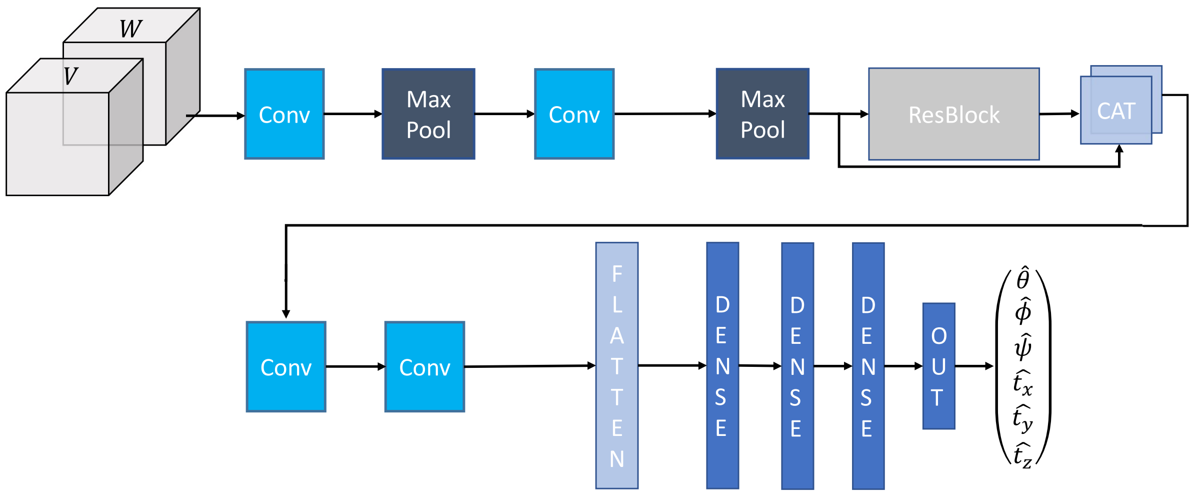


Fig. 2. Architecture of the single-stage ResNeXt network, which takes as an input two volumes stacked in the channel dimension, and outputs a predicted rotation and translation that relate the two. This network contains a ResBlock unit, which is depicted in Fig. 3**A**. Details of the parameters used in my implementation are shown in the Methods section.

of mini-batches, after which I can print the training MS. We define a mini-batch as comprising 32 volumes or point clouds, and an epoch as comprising 64 such mini-batches.

Although it is not necessary, I also print one example of a model prediction alongside the corresponding ground truth after each epoch. This gives me (incomplete) information as to whether the model has hit a local minimum, is converging in translation but not in angle, or vice versa.

2) Encountered Problems in Training: One problem I encountered in training was that RedgeNeXt would often converge to a local minimum where all of the weights tended to zero. This problem is not seen in the model presented in this work, due to the addition of residual paths with unity gain.

I also found that ResNeXt often converged well in the Euler angles, but did a poor job in converging to the correct translation vector. The solution was to add Q to the loss function, as presented in Eqn. 5, which allows control over how much error in either angle or translation contributes to the total loss.

I found that use of $Q = 0.01$, which weighs angles less than translation. A good value of Q can be chosen based on the maximum angle (in our case $80°$) and the maximum translation (in our case 9 px) used in volume generation. As the angles tended to be about 10 times larger (simply due to the chosen units), their squared error should be weighed about 100 times less. This alteration greatly improved the network's translation accuracy.

D. Software Design

An overview of the software is given below in three parts – 1) the file structure is described; 2) the Keras model and layer definitions are described; 3) the workflow is described with the functions used as blocks.

1) File Structure: The Base Directory contains a `README` file and a Jupyter notebook titled `main.ipynb`. Keras model,

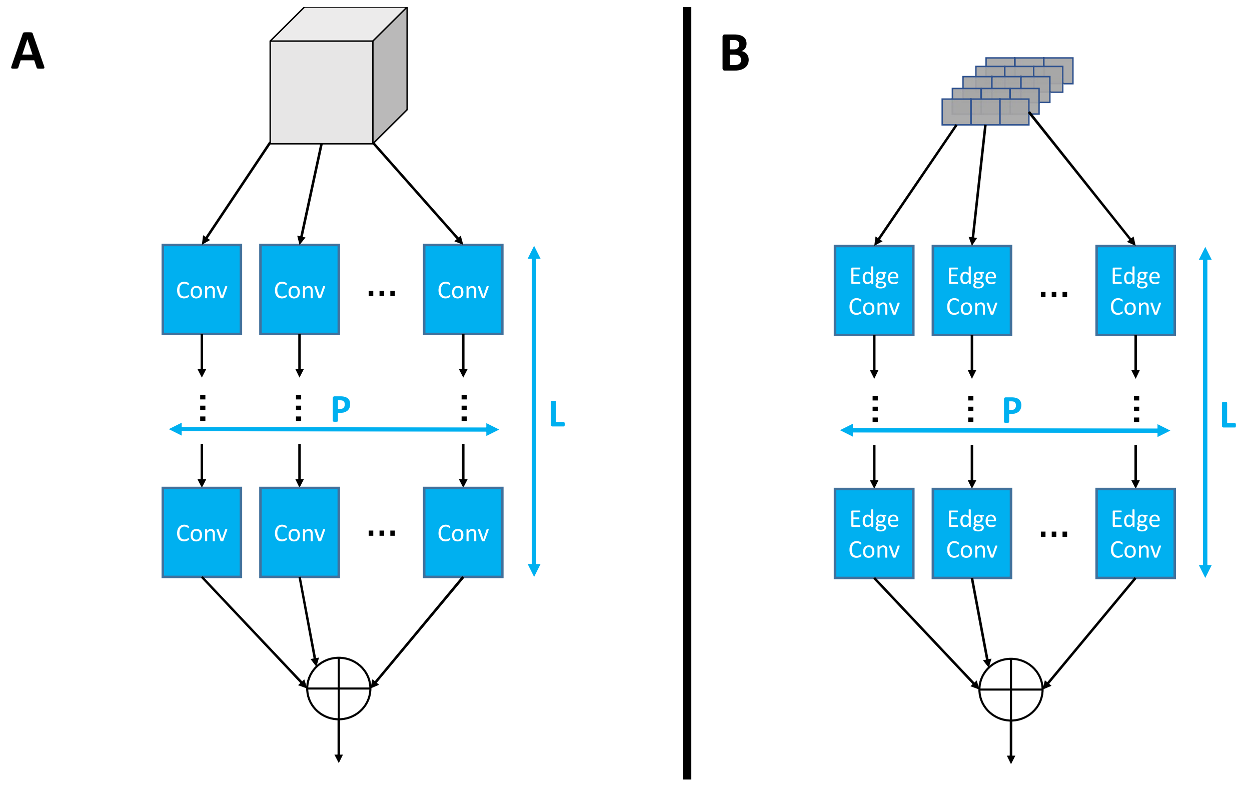


Fig. 3. **A** – The ResBlock layer used in the ResNeXt architecture in Fig 2, containing standard 3-D convolution layers. **A** – The ResBlock layer used in the RedgeNeXt architecture in Fig 4, containing EdgeConv layers. Both blocks consist of P parallel paths of L filters, and outputs the sum of the outputs from each path. Details of the parameters used in my implementation are shown in the Methods section.

layer and block definitions are in the `blocks` folder. Functions used for generating data are in the `dataGen` folder. Functions related to applying the Kabsch algorithm at the end of the FAIK algorithm are in the `Kabsch` folder. Training loops are in the folder titled `trainLoops` and testing loops are in the folder titled `testLoops`. The model parameters are saved in the `weights` folder.

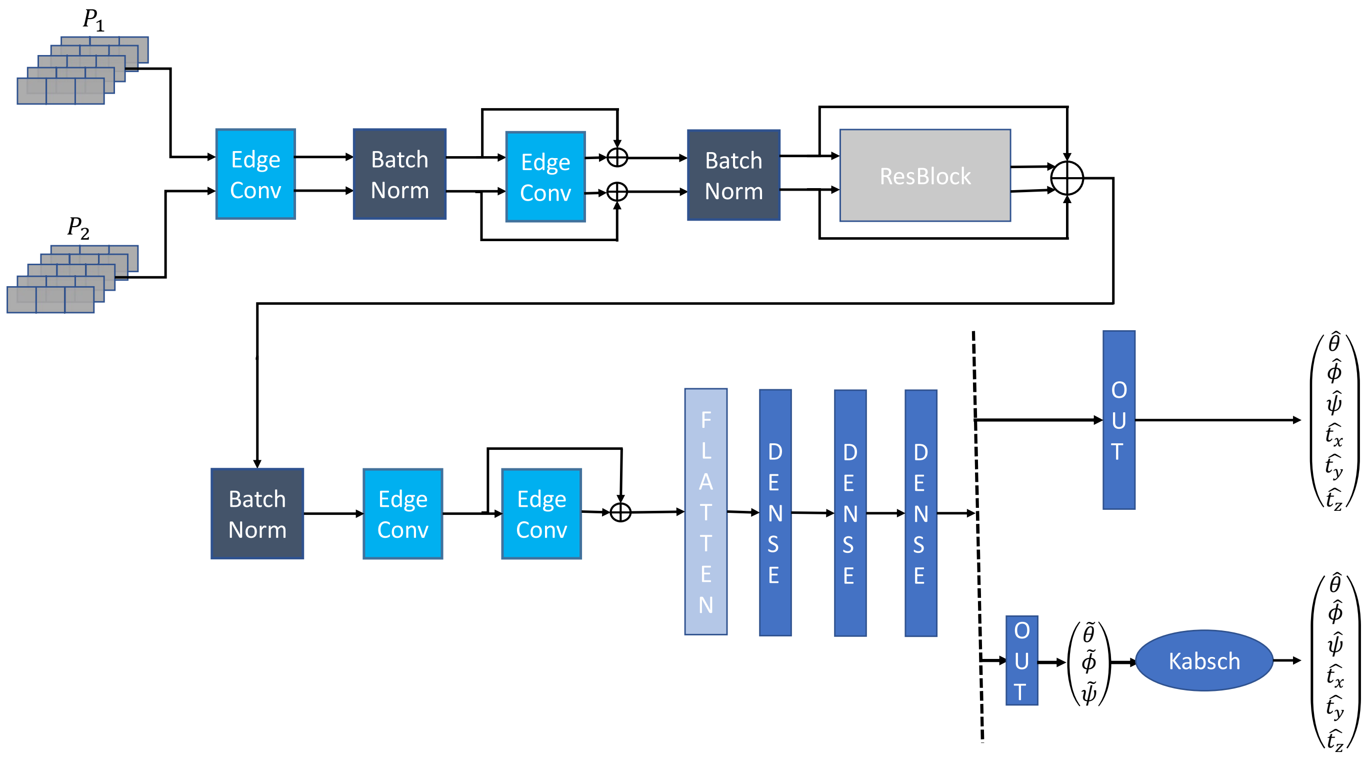
A folder full of extra functions and scripts used for testing different layers and architectures are in a folder titled `misc`, which do not contribute to this report directly. However, these prototypes were invaluable at testing and validating my custom layers and algorithms.

2) Models and Layers: I implemented subclassed Keras models rather than using the standard functional or sequential APIs. These models rely on some basic Keras layers, including convolution, max pooling, flatten, sum, batch normalization and dense layers. They also rely on a) a custom layer called `EdgeConv`, and b) a block called `ResBlock`.

The `EdgeConv` layer is an implementation of the layer described in Wang *et al.* in the special case of dense layers applied to the nearest neighbors (described above). The skeleton of this implementation was downloaded from Niklas Uwe Langner's GitLab page, with alterations made for the dense special case [14].

`ResBlock` is the characteristic residual component of ResNeXt, containing P paths of L convolutional or `EdgeConv` layers, with filters having size F. The function `makeResBlock` creates an array containing these layers for given P, L, and F, while the function `useResBlock` applies this array of filters in the parallel path structure. Different implementations of this function are used for the ResNeXt and RedgeNeXt architectures, using standard convolution or `EdgeConv`, respectively.

The `ResNeXt` model does not include `EdgeConv` layers



P_1
P_2
Edge Conv
Batch Norm
Edge Conv
Batch Norm
ResBlock
Batch Norm
Edge Conv
Edge Conv
FLATTEN
DENSE
DENSE
DENSE
OUT
$\hat{\theta}, \hat{\phi}, \hat{\psi}, \hat{t}_x, \hat{t}_y, \hat{t}_z$
OUT
$\tilde{\theta}, \tilde{\phi}, \tilde{\psi}$
Kabsch
$\hat{\theta}, \hat{\phi}, \hat{\psi}, \hat{t}_x, \hat{t}_y, \hat{t}_z$

Fig. 4. Architecture of the RedgeNeXt network, which takes as an input two point clouds. Parallel arrows denote that the point clouds independently pass through the block, as if concatenated in the batch domain. The output could be either 1) 6 parameters, describing a predicted rotation and translation that relate the two (top path in the block diagram), or 2) 3 parameters, describing only the predicted rotation. In the case that only 3 parameters are in output, the FAIK algorithm is built by applying iterative Kabsch to the output. This is seen on the bottom path in the block diagram. This network contains a ResBlock unit, which is depicted in Fig. 3**B**. Details of the parameters used in my implementation are shown in the Methods section.

or unity gain residual paths. It always outputs a 6-vector containing Euler angles and translation parameters. Its input is a list of two minibatches of volumes.

The `ResNeXt` model does not include `EdgeConv` layers or unity gain residual paths. It always outputs a 6-vector containing Euler angles and translation parameters. Its input is a list of two minibatches of volumes.

The `DeepEdgeConv` model is the RedgeNeXt backbone used for RedgeNeXt and FAIK. It does include `EdgeConv` layers and unity gain residual paths. Its input is a list of two minibatches of point clouds, size `numPts` $\times$ 3. It has a parameter `C` that determines the output size, set either to 3 or 6. This determines if it will output only the Euler angles (as in FAIK) or if it will also output the translation parameters.

3) Workflow: We begin by training ResNeXt, RedgeNeXt with 3 outputs and RedgeNeXt with 6 outputs using the training methods designed above, saving their weights. TheThey are then each tested on 800 randomly generated polynomial volumes and their errors are recorded (see Results below).

We use `FAIKtest` to test the FAIK model, which uses the saved three-output model as the initial condition of `sKabschInit`, followed by iterative application of `sKabschStep`. These use utility functions from `KabschRoutines.py`. After 1000 iterations, we record the MSE.

To test the iterative Kabsch algorithm by itself and generate Table I, we used the function `KabschTest.py`. All Kabsch-related programs were ported from MATLAB.

V. Results

In this section, the performances of the discussed algorithms are compared on several metrics. Each network has been

Algorithm	ResNeXt	RedgeNeXt-6	RedgeNeXt-3
MSE	1020	2600	–
aMSE	859	415	157
tMSE	1.66	22	–
Train Time [sec/Sample]	0.3	0.25	0.23

TABLE II

ERROR AND TRAINING TIME FOR THREE TESTED NEURAL NETWORK MODELS – ONE-STAGE RESNEXT, 6-OUTPUT REDGENEXT AND 3-OUTPUT REDGENEXT. WE PRESENT MSE FOUND USING EQN 5 WITH $Q = 0.01$, AND THE CONTRIBUTIONS OF ANGULAR AND TRANSLATIONAL ERROR INDIVIDUALLY TO THIS MSE. IN COMPUTING TRANSLATIONAL ERROR ALONE, WE DO NOT DIVIDE BY Q. ONLY AMSE IS SHOWN FOR REDGENEXT-3, AS THIS NETWORK ONLY PRODUCING ESTIMATES OF THE EULER ANGLES. ALL ERRORS ARE FOUND BY AVERAGING OVER 800 TEST VOLUMES OR POINT CLOUDS.

Algorithm	ResNeXt	RedgeNeXt-6	FAIK	RedgeRes
MSE70	857	2170	0.00	285
MedSE	975	2520	0.00	403
MSE	1020	2600	81700	323

TABLE III

ERROR FOR FOUR TESTED ALGORITHMS, INCLUDING ONE-STAGE RESNEXT, 6-OUTPUT REDGENEXT AND FAIK. WE ALSO SHOW THE PERFORMANCE OF A MODEL WE CALL REDGERES, WHICH TAKES THE TRANSLATION FROM RESNEXT AND THE ANGLE FROM REDGENEXT-3. WE PRESENT MSE FOUND USING EQN. 5 WITH $Q = 0.01$, STANDARD MSE FOR THE ANGLE AND TRANSLATION VECTORS AND MEDIAN SQUARE ERROR. ALL ERRORS ARE FOUND BY AVERAGING OVER 800 TEST VOLUMES OR POINT CLOUDS.

trained on 102400 samples. All computations were performed on a computer running Windows 10 Home, with 16 GB of RAM and an Intel i7 processor. The computer is equipped with an NVIDIA GeForce RTX 2070 Super GPU.

A. ResNeXt vs. RedgeNeXt

As stated above, I chose to build the FAIK algorithm from the RedgeNeXt architecture, rather than the ResNeXt architecture. To see why this decision was made, we consider the performance of ResNeXt and RedgeNeXt in terms of 1) total MSE found using Eqn 5; 2) angular contribution to MSE, aMSE; 3) translational contribution to MSE, tMSE (not including the Q multiplier, so that units are in px^2).

Table II shows the performance of these algorithms averaged over 800 test volumes or point clouds. We show angular MSE for both RedgeNeXt with all 6 parameters as outputs and with only angles as outputs (labeled RedgeNeXt-6 and RedgeNeXt-3 respectively). We also show training time of all three networks per sample, which are similar across the board. The largest training bottleneck is data generation.

What can be gleamed from these results is that, using this value of Q, ResNeXt greatly outperforms RedgeNeXt in terms of total MSE. However, isolation of the error components shows that RedgeNeXt performs significantly better than ResNeXt in predicting *angle*, while ResNeXt does a far better job in predicting *translation*.

As discussed above and quantified in Table I, the iterative Kabsch algorithm is most sensitive to conditions in rotation angle, and performs near-invariant with respect to translation. This indicates that RedgeNeXt is better suited for FAIK than ResNeXt.

When the network's task is reduced only to predicting the Euler angles, we see that 3-output RedgeNeXt performs exceptionally compared to ResNeXt and the 6-output RedgeNeXt model. This makes it the best choice for our feature-acceleration stage in the FAIK algorithm.

B. Comparison of all tested methods

We compare the methods in three ways – 1) Median square error (MedSE); 2) total MSE found using Eqn 5; 3) the MSE computed from the best 70% of results, MSE70. We average over 800 volume pairs, with volume parameters as described above. These errors are shown in Table III.

Included is an as-of-yet not discussed method we call ResRedge, which is built on parallel application of ResNeXt and RedgeNeXt-3. The predicted Euler angles are the outputs of RedgeNeXt-3, while the predicted translations are from the output of ResNeXt. This is motivated by the results in Table II, which shows that ResNeXt is far superior in translation prediction, while RedgeNeXt-3 is far superior in angle prediction.

We choose to show the median square error because error in the FAIK algorithm is largely binary – either it converges very well (near-0 error) or it converges very poorly. This massively skews the MSE in the FAIK case only, as this binary error is not as present in the other networks tested. In these cases, MedSE is similar to MSE.

The same goes for MSE70, which removes the effect of outliers while still predicting the most likely behavior of the method. In certain contexts, it is unimportant *how poorly* the algorithm converges if it is not performing perfectly. For example, there is no practical difference between being off by 10 degrees and being off by 180 degrees in the cochlear mechanics experiments of interest.

In observing this table, it is attempting to interpret that RedgeNeXt performs very poorly compared to ResNeXt. It is important to recall that in Table II, RedgeNeXt does outperform ResNeXt in angular prediction, but suffers higher losses in MSE due to its poorer translation prediction.

VI. DISCUSSION

The FAIK algorithm shows advantage over all other tested algorithms in three fashions – 1) its training time is smallest of all tested algorithms; 2) its MSE70 and MedSE are superior to all other tested algorithms; 3) it most commonly converges with an incredible error of 0, as expected from iterative Kabsch when well-conditioned.

This confirms that this problem can be addressed by a hybrid approach, using both a neural network and a classical method in series. This is mathematically intuitive, given that a neural network only approximates the least-error solution up to what is possible given its architecture, while a well-conditioned classical method arrives at precisely the least-error solution.

One of FAIK's drawbacks is run-time. Because FAIK relies on an iterative algorithm being performed after network training, it takes much longer to run than methods that simply pass volumes through a neural network once. As this application time is still less than one minute, it is a very reasonable tradeoff for the application at hand.

Surprisingly, FAIK also yields a very large MSE despite excellent performance most of the time. It diverges wildly in the cases in which it diverges, but it performs perfectly in the cases in which it converges. Whether this behavior is desirable, as opposed to moderate across-the-board loss, is application-dependant. Perfect registration most of the time is more desirable for cochlear volume registration, which is my object of interest.

RedgeNeXt and ResNeXt are better if more consistent performance is desired. Although they are not nearly as accurate as FAIK in most cases, they are not subject to massively diverging outliers like FAIK is. Their faster run-times may be beneficial in certain applications that require on-the-fly approximation on small timescales.

ResNeXt provides better translation accuracy, while RedgeNeXt provides better rotation accuracy. RedgeRes, which requires training and applying both of these algorithms in parallel, achieves overall better performance than either network at the cost of higher training time and run-time.

Informed use of these three algorithms in tandem, if reasonable, could yield both consistent and accurate predictions. For example, with some knowledge of your problem, the massive divergence of FAIK will be clear when it occurs (e.g. it may tell you your translation is 500 px when you only reasonably maximum translation of $\sim$10 px). In the cases where this divergence is not occurring, you are very likely to achieve perfect registration with only FAIK. In the divergent case, one can use the results from both ResNeXt (for translation) and RedgeNeXt (for angle) for a lower-error prediction.

VII. Conclusion and Future Work

The present work shows that a hybrid approach combining a neural network and a classical method performs with incredible success in volume registration subject to rotation, translation and occlusion. This algorithm, FAIK, has been shown to outperform pure neural network and pure classical models for synthetically generated volumes in the median case. However, it suffers from massive divergence in outlier cases.

I have also developed a pure neural network approach, RedgeNeXt, which performs worse than FAIK in the median case, but is far more consistent. Its error is moderate, but it outperforms other known methods such as ResNeXt at rotation angle prediction.

Comparison of methods shows that ResNeXt outperforms RedgeNeXt at translation prediction, inspiring a parallel method called ResRedge wherein each network is used to predict different transformation parameters. This method performs worse than FAIK in the median case, but is more consistent (in that it has lowest average MSE) than any other studied method.

All methods proposed in this study show promise, and which method is chosen depends on the application at hand.

Future work will focus mostly on data acquisition. Although these volumes were generated to model OCT volumes, a large enough set of OCT volumes was not readily available to train or test the model on real data. Presently, I am in the process of crafting a large data set of volumes from the gerbil heart and cochlea. With a sufficiently large data set, the present work indicates that the FAIK algorithm can be trained to register these real volumes as well.

Other improvements could be made by testing different optimizers – I explored only Adam and SGD, but it is possible that other methods would be more useful here. One could also try having the algorithm learn a versor representation of rotation as opposed to Euler angles – this form may be easier to learn, and these tests could be performed with minor alteration to the software used in this project.

If the method proves to be successful on real OCT data, I will incorporate it into current OCT experiments in cochlear mechanics at Columbia's Fowler Lab. In particular, it can be used to quickly measure motion from the same sets of points after having rotated the preparation relative to the OCT scanner. This would be useful in conducting a reconstruction of 3-D micromechanical motion in the cochlea – a goal I have been working towards for my PhD thesis. As of now, I have only succeeded in performing 2-D registration via a complex physiology-based registration approach, which is very slow. A faster method would open the door for 2- and 3-D reconstructions on short enough timescales to probe micromechanics in many cell groups, potentially on the timescale of a recovery experiment. I plan to work with New York University PhD candidate Nikola Janjusevic this Winter and next Spring to perfect the FAIK algorithm so that it can be used in real OCT applications.

Acknowledgments

I would like to acknowledge the hard work of the professor and the teaching assistants, who have provided me with so much knowledge of the subject material! I would like to thank New York University PhD candidate Nikola Janjusevic, whose familiarity with contemporary literature helped me find Wang *et al.*'s EdgeConv layer. I would like to thank my co-advisors, Dr. Christine Hendon and Dr. Elizabeth Olson. Finally, I would like to thank the National Institute of Deafness and other Communication Disorders for funding my research and schooling at Columbia.

References

[1] J. A. Izatt and M. A. Choma, "Theory of Optical Coherence Tomography," in book *Optical Coherence Tomography: Technology and Applications*, pp. 47-72, 2008.

[2] Yu Gan *et al.*, "An Automated 3D Registration Method for Optical Coherence Tomography", *Conference Proceedings IEEE Engineering Medical Biological Society*, pp. 3873-3876, 2014.

[3] M. A. Choma *et al.*, "Spectral-Domain Phase Microscopy", *Optics Letters*, vol. 30, num. 10, pp. 1162-1164, 2005.

[4] Brian L. Frost, C. Elliott Strimbu, and E. S. Olson, "Using volumetric optical coherence tomography to achieve spatially resolved organ of corti vibration measurements", *Journal of the Acoustical Society of America*, vol. 151, pp. 1115–1125, 2022.

[5] Brian L. Frost, C. Elliott Strimbu and Elizabeth S. Olson, "Reconstruction of transverse and longitudinal motion in the organ of Corti complex via optical coherence tomography", *Mechanics of Hearing 2022*, https://moh2022.dtu.dk/ .

[6] Brian L. Frost, C. Elliott Strimbu and Elizabeth S. Olson, "Reconstruction of transverse-longitudinal vibrations in the organ of Corti complex via optical coherence tomography", in revisions, *Journal of the Acoustical Society of America*, 2022.

[7] H. Y. Lee *et al.*, "Two-Dimensional Cochlear Micromechanics Measured In Vivo Demonstrate Radial Tuning within the Mouse Organ of Corti", *Journal of Neuroscience*, vol. 36, no. 31, pp. 8160-8173, 2016.

[8] Wihan Kim *et al.*, "Vector of motion measurements in the living cochlea using a 3D OCT vibrometry system," *Biomedical Optics Express*, vol. 13, pp. 2542-2553, 2022.

[9] Sebastiaan W. F. Meenderink and Wei Dong, "Organ of Corti vibrations are dominated by longitudinal motion in vivo", *Communications Biology*, vol. 5, no. 1, pp 1285, 2022.

[10] Hengtao Guo *et al.*, "Deep Adaptive Registration of Multi-Modal Prostate Images", *Computerized Medical Imaging and Graphics*, vol. 84, 2020.

[11] Yue Wang *et al.*, "Dynamic Graph CNN for Learning on Point Clouds", arXiv:1801.07829v2 cs.CV, 11 June 2019.

[12] Paul J. Besl and Neil D. McKay, "A Method for Registration of 3-D Shapes", *IEEE Transactions on Pattern Analysis and Machine Intelligence* vol. 40, no. 214.2, pp. 239-256, 1992.

[13] Brian L. Frost, "Quaternions and Volume Registration – A Comprehensive Approach", https://brian-frost-laplante.github.io/blog/Optimal_rotation/.

[14] Niklas Uwe Langner, EdgeConv_Keras, https://git.rwth-aachen.de/niklas.langner/edgeconv_keras .