IMAGE REGISTRATION WITH AND WITHOUT LABELED MASK

A Thesis Presented

by

NEHA GOYAL

Submitted to the Office of Graduate Studies,

University of Massachusetts Boston,

for the degree of

MASTER OF SCIENCE

December 2021

Computer Science Program

IMAGE REGISTRATION WITH AND WITHOUT LABELED MASK

A Thesis Presented

by

NEHA GOYAL

Approved as to style and content by:

ABSTRACT

IMAGE REGISTRATION WITH AND WITHOUT LABELED MASK

December 2021

Neha Goyal, B.Tech., Rajasthan Technical University
M.S., University of Massachusetts Boston

Directed by Professor Daniel Haehn

To investigate if adding biological features can improve the existing registration process in state-of-art and deep learning networks, mitochondria masks or lung masks data were used to guide the alignment procedures in real-time. The input datasets consist of unaligned 2D electron microscopy (EM) images that are computationally expensive to map and create 3D volumetric datasets. Feature matching methods and a deep learning framework, MONAI, were implemented to align 2D EM images and 3D lung CT scans, respectively. This approach will guide the registration methods to run faster and with better accuracy for biomedical image analysis.

TABLE OF CONTENTS

CHAPTER PAGE

LIST OF FIGURES

LIST OF TABLES

Tables Page

CHAPTER 1

INTRODUCTION

Image registration is a key processing step in a variety of bio medical image applications. It involves aligning of one dataset to another in the same coordinate space such that the mapped points in two datasets corresponds to the same anatomic features. Image registration is a prerequisite when required to estimate similarities between images acquired at different times or from different subject by different sensors. It is used in many imaging applications and clinical research such as segmentation, image reconstruction, image guidance, pathological detection and so on.

There are three main registration analysis categories according to image acquired:

1. **Multi-temporal analysis:** The images are acquired at different times on regular intervals which may be taken across different patients or same patient to find and evaluate changes. For example, to compare tumor growth across time.

2. **Multi-view analysis:** The datasets are acquired at different views to register multi-dimensional image. For example, to gain a 3D image reconstruction from 2D images.

3. **Multi-modal analysis:** The images are captured using different imaging

modalities to register mutual information for comparisons. For example, CT and MRI images together can produce additional information for clinical diagnosis.

The registration algorithms can be classified into rigid (translation and rotation), affine (shear) and deformation models which may use region of interest such as lung mask, brain mask and so on to align image pair from 2D to 2D/3D or 3D to 3D. In any pairwise registration algorithm, comparisons are made between the two images such that one of the images is referred as moving or source image and the other is referred as fixed or target image. All the spatial transformations are applied on moving image such that it maps to the fixed image.

Depending on the matching techniques, the registration algorithms can be classified as intensity based and feature based methods. Intensity based methods uses iterative process to make comparison with the help of pixel or voxel intensity matrix. Feature-based methods use contours, edges, lines point based keypoints to detect and extract features, make necessary comparisons, and apply transformations accordingly.

1.1 Deep Learning

Deep learning is a subfield of machine learning which is again a subfield of artificial intelligence. It is a learning inspired by the structure and function of human brain, and uses hierarchical structured algorithms called neural networks. These algorithms include statistics and predictive modelling in an autonomous way. Neural Networks comes in different forms such as convolutional neural network, recurrent neural network, artificial neural network and so on. Although each of these neural networks works best with their own use cases, the functionality is somewhat similar in all. Each individual layer in the network acts like a filter to detect similarities or to make right interpretation and decisions from the output of the previous layer. This layered strategy allows learning from previous results helps the model to make refined correlations and suppress any irrelevant information. These models can feed on large datasets to train and are able to make decisions with less or no human involvement. Mathematics involved in these algorithms is complex and hence different variations of layers are added to get the abstract representation of data which results optimized outputs.

1.1.1 Feature Learning

Traditional machine learning algorithms such as SVM, decision trees, logistic regression required a pre-processing step for feature extraction. Only after this step the methods could work on output data to make classifications. That means these methods could not be directly applied on raw datasets such as images, text to categorize data into various classes. Benefits of deep learning model is that it can perform automatic feature detection and extraction from raw data. This means the model feeds on raw data; the hidden layers perform the complex feature extraction step by resulting more compressed representation of raw data from each layer to categorize data according to its similarities as the output.

1.1.2 Homography Learning

In this, the algorithm directly learns the geometrical transformations to align the images. Researchers have found that a VGG style regression model can used for homography estimation task between two images. The homography matrix is a three-dimensional array.

1.1.3 Reinforcement Learning:

In this learning the model employs a system of rewards and penalties, in other words the model works on an unstructured input data where hidden layers learn through trials and error to reach the output goal. Here the model is not told what actions to take.

1.1.4 Supervised Learning:

In this the models are trained with the labels to guide the learning. These labelled data supervise and guide the model to predict the output correctly. This is possible because the model is provided with the output data for each input data. So, the model simply maps or predicts the input with the correct output. These models can be used in Risk Management, Spam filtering and so on. Regression and classification problems can be handled with supervised learning.

1.1.5 Unsupervised Learning:

Unlike supervised learning, in this learning there is no labeled data available. The model needs to predict the pattern in the input data to yield the correct output. Here the model finds the underlying patter, recategorize the data accordingly to similarities and then predict the correct output. Since in real world we always do not have a corresponding output available like in supervised leaning, we need more unsupervised learning models with better accuracy.

CHAPTER 2

TRADITIONAL FEATURE LEARNING AND HOMOGRAPHY

Feature extraction is an important technique used for object detection, image alignment, image stitching, car, or robot navigation systems and more. Any traditional model follows major four steps to align images using feature extraction method.

1. **Feature detection:** The model learns the features or patterns from the input dataset. These localized features include properties such as region of interest (ROI), edges, lines, corners which help the algorithm to detect the objects. The features are represented as keypoints including x and y coordinate of the bounding box or the object contour, size, or diameter of the keypoint neighborhood taken in consideration, angle of orientation, octave pyramid in which the keypoint was found, response or strength of the keypoint

2. **Feature extraction:** The keypoints are then used by feature learning algorithm to create descriptors. Descriptors are identity of each feature point. Each descriptor is encoded with additional information about its corresponding keypoint, such that it is unique and could be referred as numerical fingerprint of its keypoint. Ideally, descriptors should remain unaltered under any image

transformations applied such as translation, scaling, rotation and so on, so that one can find the feature again even when image is transformed. Any descriptor is represented as vector value.

3. **Feature matching:** In this step mapping of (x_i, y_i) of source image to (x_i', y_i') of target image is done. Descriptors so created by any feature learning algorithm is used in feature matching process. A list of matching keypoint vectors each associated with its descriptor vector is created using nearest search or distance calculations. This distance can be hamming distance or normalized distance used in brute-force matcher algorithm or k-nearest neighbor search used in the FLANN matcher algorithm to match the keypoints between the two images. The list contains (i, j) values such that ith keypoint and descriptor pair in source image matches with the jth keypoint and descriptor pair in target image according to the applied distance algorithm.

4. **Finding homography:** To align images the method finds the perspective transformation of one image to another. In this step method finds the homography with the help of matched points and returns a 3x3 matrix and mask data. RANSAC algorithm is applied when finding homography such that the method needs at least four pair points to construct the homography matrix. With the help of this 3x3 matrix the source image is warped to the target image and outputs the warped image.

2.1 SIFT (Scale-Invariant Feature Transform)

One can say that this is the first algorithm that involved keypoints and descriptors for feature matching. The algorithm performs a scale-space peak selection for finding features. Gaussian filter is used to blur the input image at different scales. The filter is applied at different octave levels in the scale-space. The output of these blurred images gives DOG (difference of gaussian) images which is used to find the keypoints.

Each pixel of image is compared with its neighbor and its previous and next scale. If a local extremum is found, then that is considered as the best keypoint in that scale. These potential keypoints are refined using contrastThreshold and edgeThreshold values. This way any low-contrast or edge keypoints are discarded.

Now, orientation is assigned to each keypoint such that keypoints with same location and scale have different directions. This contributes to stability of matching. For each keypoint, a descriptor is created. Each keypoint descriptor vector contains additional information about its corresponding keypoint. These list of descriptor vectors are used by feature matching algorithms.

2.2 FAST (Features from Accelerated Segment Test)

Features from accelerated segment test (FAST) is a corner detection method is used for feature detector. The method selects a pixel p having intensity I_p in the image to determine whether it to be considered as an interest point or not. It then selects a threshold value t and considers 16 continuous pixels in a circle around the pixel point which is under the test. This is called a Bresenham’s circle with a default radius 3. Now the method determines pixel p is a corner point if there exist a set of n continuous points in the circle of 16 pixels that are all brighter than $I_p + t$ or all darker than $I_p - t$.

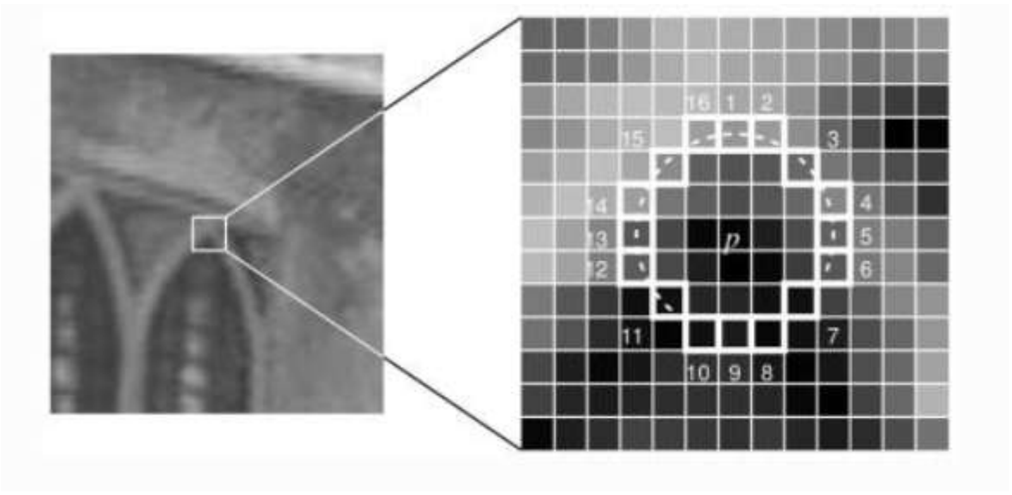


Figure 1: Keypoint detection with FAST

For fast performance, a high-speed test was proposed to exclude all the non-corner points. In this test like the four extremes in a clock, intensities at pixel 1,5,9 and 13 in the circle are compared with I_p. For p to be a corner, at least three out of the four pixels needs to satisfy the intensity threshold criteria, or it is considered that p is not a corner. If pixel p passes the test, it is not the algorithm checks for n continuous points in the circle to satisfy the intensity threshold criteria. This is iterated for all the pixels in the image.

But even when high speed test is applied some limitations were found. First, if n is greater than 12, the method does not perform well because a high number of interest points were detected in some cases. Second, the order in which these 16 pixels are considered determines the speed of the algorithm. Third, the results of high speed were thrown away. Last, the adjacent interest points detected were high in number.

To address these issues machine learning approach and non-maximal suppression was applied.

Machine learning approach was used for the first three limitations. In this, FAST algorithm was applied on a set of images to find feature points. For each interest point a feature vector P is created that stores the 16-pixel value around it. Each pixel point is categorized in three states according to the intensity threshold criteria. The states are: *darker* if n continuous pixels are less than or equal to $I_p - t$, or *brighter* greater than or equal to $I_p + t$, or *similar* if the value is in between the range $I_p - t$ and $I_p + t$. Depending on these states a boolean variable K_p is defined which holds true if p is corner otherwise false. A decision tree classifier (ID3 algorithm) is applied to query each subset with the help of true class knowledge using variable K_p. This classifier selects the pixels that gives the most information about the interest point to determine if it is corner using minimum number of queries. This is measured by the entropy K_p and is applied recursively to all the subsets until the entropy is zero. The decision tree so created is used for FAST detection.

To solve multiple interest points in adjacent location non-suppression maximum is applied. For this, the method calculates a score function V for all the detected keypoints, where V is the sum of absolute difference between p and the surrounding 16 pixels. For any two adjacent interest points, the point with lower V value is discarded.

2.3 BRISK (Binary Robust Invariant Scalable Keypoints)

This algorithm uses FAST method for keypoint detection. Here, BRISK considers at least 9 consecutive pixels out of the 16 pixels in a circle which are brighter or darker than the center pixel for a keypoint detection. In addition to that the method uses scale-space pyramid selection to achieve scale invariance. This also helps in achieving sub-pixel accuracy.

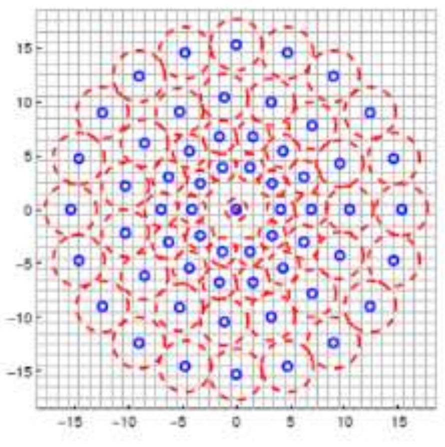


Figure 2: Concentric circle pattern in BRISK for keypoint detection

For keypoint description, BRISK samples pixels in a concentric ring pattern where for every sampling point, Gaussian smoothing is applied on a small neighboring patch around the keypoint. The smoothing avoids aliasing effects. For sampling, the method measures two components: long and short distance pairs.

Each descriptor is built by comparing intensities between sample pair having distance below a threshold distance-max and orientation of the keypoint is calculated with the help of large gradient between sample pair having distance above distance-min. This helps BRISK to achieve rotation invariance by rotating the sampled pattern by that orientation. For each short distance sample pair if the first pixel has greater intensity than the second, 1 is written to the corresponding bit else 0 is written. Thus, the method returns binary BRISK descriptor string with 512 bits which will help in fast matching between the descriptor pairs.

2.4 FREAK (Fast Retina Keypoint)

Like BRISK, this algorithm has defined sampling pattern. FREAK uses retinal sampling grids which are circular and have more point density around the center. This point density decreases exponentially with the distance from the center. Each sampling point is smoothed to avoid noise sensitivity. The retina model uses different kernel size for each sample point, such that the kernel size change exponentially and have overlapping receptive fields. For each circle, a standard deviation of gaussian kernel is applied to the sample point. The overlapping of receptive field gives more information which helps to achieve better accuracy.

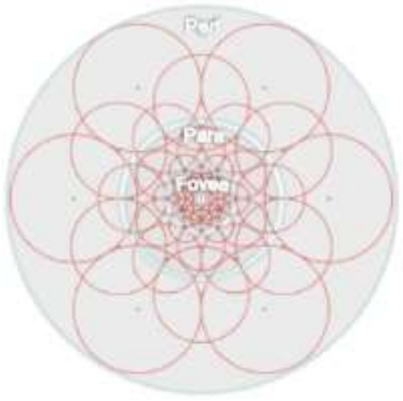


Figure 3: Pre-defined sampling pattern in FREAK for keypoint description

To measure the orientation of the keypoint, unlike BRISK that select pairs according to the spatial distance, FREAK uses ORB approach for better leaning of the pairs which are not correlated and are most discriminative. The first pairs are selected form the outer rings of the sampling pattern and last pairs are selected from the inner rings of the pattern. Just like a human eye, the peri area receptive fields are first used to determine the location of the interest point. The validation is made through the densely distributed receptive fields in the fovea area. Like BRISK, the orientation weights of the selected pairs are summed, and the sampling patch is rotated by this orientation to achieve rotation invariance.

To build a descriptor, intensity comparisons are made of a set of 512 sampling pairs. For each sample pair if the first point has higher intensity value than the second point, 1 is written to the corresponding bit of the descriptor otherwise 0 is written.

2.5 ORB (Oriented FAST and Rotated BRIEF)

This is fast algorithm when compared to SIFT or SURF. ORB uses FAST algorithm for keypoint detector and BRIEF algorithm for keypoint descriptor. Since FAST features do not have an orientation and multiscale feature components; ORB algorithm uses Harris corner detection that is applied on the FAST keypoints to find the top N points and uses pyramids to produce multiscale features. Each image is represented in a sequence of image versions at different resolutions. Now when detecting keypoints at different levels, ORB can also locate keypoints at different scale. To measure the orientation of these keypoints which depends on the change in level of intensity around that keypoint, ORB computes intensity centroid to detect this change. It calculates the offset of the located corner point at center with its centroid to obtain the direction vector which gives the orientation. Moments are used to find the centroid of the patch to measure the direction vector with the corner's center. Also, the centroid point should be in the circular region of the patch.

As we know that BRIEF algorithm is not invariant to rotation, so ORB steers BRIEF according to the orientation of the keypoints. A $2 \ x \ n$ matrix S is defined for any feature set of n binary tests at location (x_i, y_i). This matrix S contains the coordinates of these pixels. A rotation matrix is found using the orientation of the patch θ, ORB applies this rotation matrix on S to get the rotated (sheered) version of this $2 \ x \ n$ matrix; S_θ. The algorithm discretizes the angle to increments of 2π/30 (12 degrees), and a lookup table of precomputed BRIEF patterns is constructed. The correct S_θ will be used to compute the descriptors only when the orientation θ of the keypoint is consistent across the views.

But it was found that when orientation functionality is added to the BRIEF it loses its important property where each bit feature has high variance and a mean close to 0.5. For this ORB runs a greedy search among all the possible binary tests to obtain output tests with high variance and a mean near 0.5, With this it also finds tests which are not correlated since each test contribute to the result. Resulted BRIEF algorithm with added functionalities is called rBRIEF.

ORB is good choice in low powered devices because of this speed and performance.

2.6 Brute Force Matcher

It is a matcher algorithm and is used to find the similar keypoints between two images. Descriptors are used to find the matching keypoints and are matched based on the distance measured between the two keypoints. The algorithm matches each descriptor of source image with all the descriptors of the target image using distance calculation. OpenCV suggests different distance calculations according to the feature detection algorithm that is used. By default, it uses L2 norm, also known as Euclidean distance. This distance is the measure of one point to another.

For SIFT, SURF; OpenCV suggests L1 norm, also known as Manhattan distance. This distance is the sum of absolute difference of the magnitudes of the vectors in a space. For ORB, BRISK; OpenCV suggests Hamming norm, also known as Hamming distance. The distance is the measure of the number of positions at which the two equal length binary descriptors differ.

With crosscheck value true, the OpenCV method returns the matches having $(i,\ j)$ value where ith descriptor of the keypoint from the source image has best match with jth descriptor of the keypoint from the target image and vice-versa. In this thesis we are using the OpenCV method *BFMatcher.match()* that return the best match between the descriptors from the two images. At last, one can sort the matches in ascending order of distance to get the first few best matches having the minimum distance between the feature points.

2.7 FLANN Based Matcher

Fast Library for Approximate Nearest Neighbors (FLANN) algorithm uses the fast nearest neighbor search to match the feature points. It is found that for larger datasets FLANN works faster when compared with BF matcher. To get better quality of good matches, one can create FLANN object with the help of FLANN parameters. The first parameter is the index parameter is created by passing value of k-dimensional tree in a dictionary. This algorithm in index parameter organizes the data structure points in k-dimensional hierarchical space. The second is the search parameter and specifies the number of times the tree in the index parameter will traverse recursively. As the number of checks increase, matching accuracy increases but this will also increase the execution time.

To find the best matches with FLANN matcher, we have used an OpenCV method *matcher.knnMatch()* that yields the kth matches for each descriptor from the query. k=2 is second best match, k =3 is third best match and so on. To keep the best match, we have used David Lowe’s ratio test. The goal here is to reject all the keypoints which are not distinct enough. This can be done by comparing the distance between the first best match and the second-best match. If this distance is large enough that means the descriptor of that points is distinct. This is ratio test. The test compares a ratio r such that if first-best match distance is $r\%$ less than the second-best distance then it is a good match otherwise not. After this we get the list of best matches between the descriptor of source image and the descriptor of the target image.

CHAPTER 3

MITO FEATURE DETECTOR METHOD

This is our proposed method for feature detection. The keypoints are detected using mask data rather than image data. The mask data is a binary image data with 0 and 1, such that pixel value where mask is detected is 1 otherwise 0. Mask data is used to find the bounding boxes, contours, to relocate mask on image data. To implement this in OpenCV, *cv2.findContours()* method is used on mask data. This gives a list of all the bounding boxes found on the mask image. Iterating over this list, polygon is found for each bounding box. This polygon is formed with a list of (x, y) coordinates. *cv2.approxPolyDP()* method is measured for each bounding box to output a list of coordinate points to the draw the polygon on image data. To create the keypoint vector list *cv2.KeyPoint()* method is used to create a list keypoints vectors for all the interest points that were used to draw the mask polygon on image data.

This list of keypoint vectors is passed to any feature algorithm that can be used to create the descriptors for each keypoint in the list. With the help of the descriptors, keypoints are matched using BF and FLANN matcher algorithms. After matching the keypoints, the transformation matrix is calculated for mapping the source image to target image. This matrix is applied on source image and source label mask to yield the warped or aligned image and warped or aligned mask image. Since only label mask image is used to detect the keypoints, it can be said that it is a guided approach to register the source with target image with help of feature matching method.

CHAPTER 4

MONAI – A DEEP LEARNING FRAMEWORK

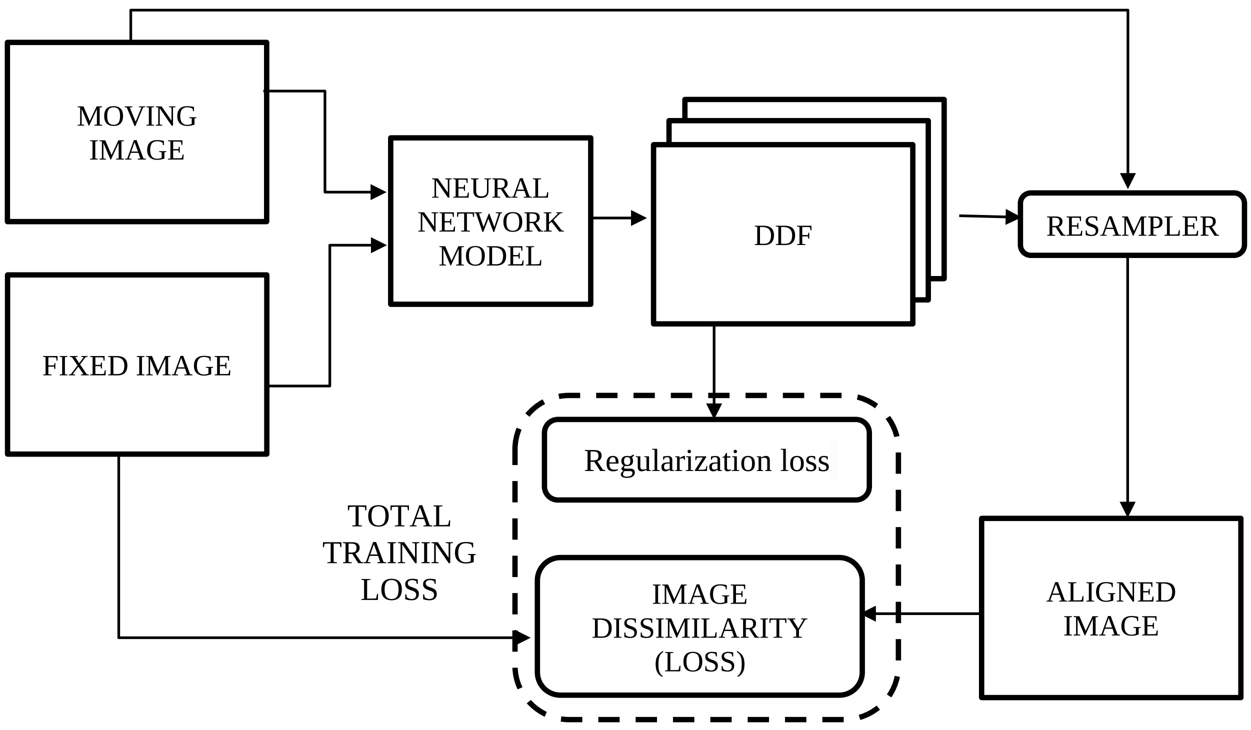
MONAI stands for Medical Open Network for AI. It is a deep learning open-source framework used for biomedical imaging and is a part of PyTorch community. It supports image processing for multimodal medical images and uses APIs for easy integration. It allows users to modify model designs according to the dataset requirements and supports multi-GPU data parallelism. MONAI also includes many image registration models and allows to calculate alignment scores and evaluation metrics.

Image registration methods in MONAI are inspired by DeepReg. For image alignment, this framework uses fixed and moving image to map the matched coordinates. With the help of deep learning, the registration network yields a dense displacement field (DDF) which informs the mapping coordinates between moving and fixed image to predict the aligned image. DDF has the same shape as the moving image and each value of DDF defies the displacement of the corresponding voxel of the moving image from the fixed image.

Depending on the input dataset, model is trained and evaluated with different approaches.

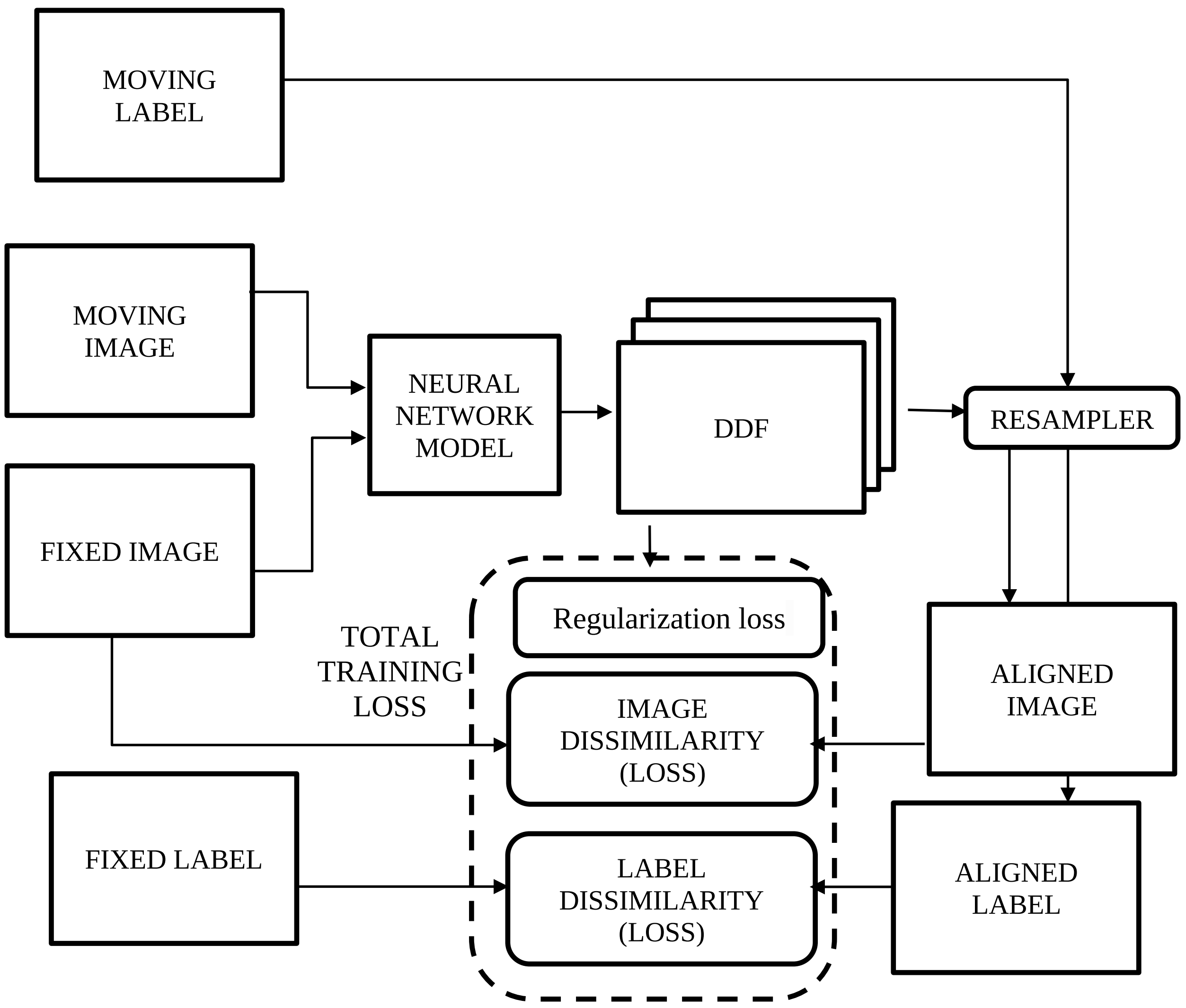
4.1 Unsupervised Learning

Here labeled data is not provided in input data and the training model calculated the unsupervised loss. The loss function is comprised of intensity-based loss and bending energy. Intensity-based loss is the measure between fixed and aligned moving image by calculating the mean square error loss (MSE). MSE is an easy loss calculation and is done by taking the difference between the predicted and ground truth value, squaring it for each pair value and finally average it out for the whole stack.



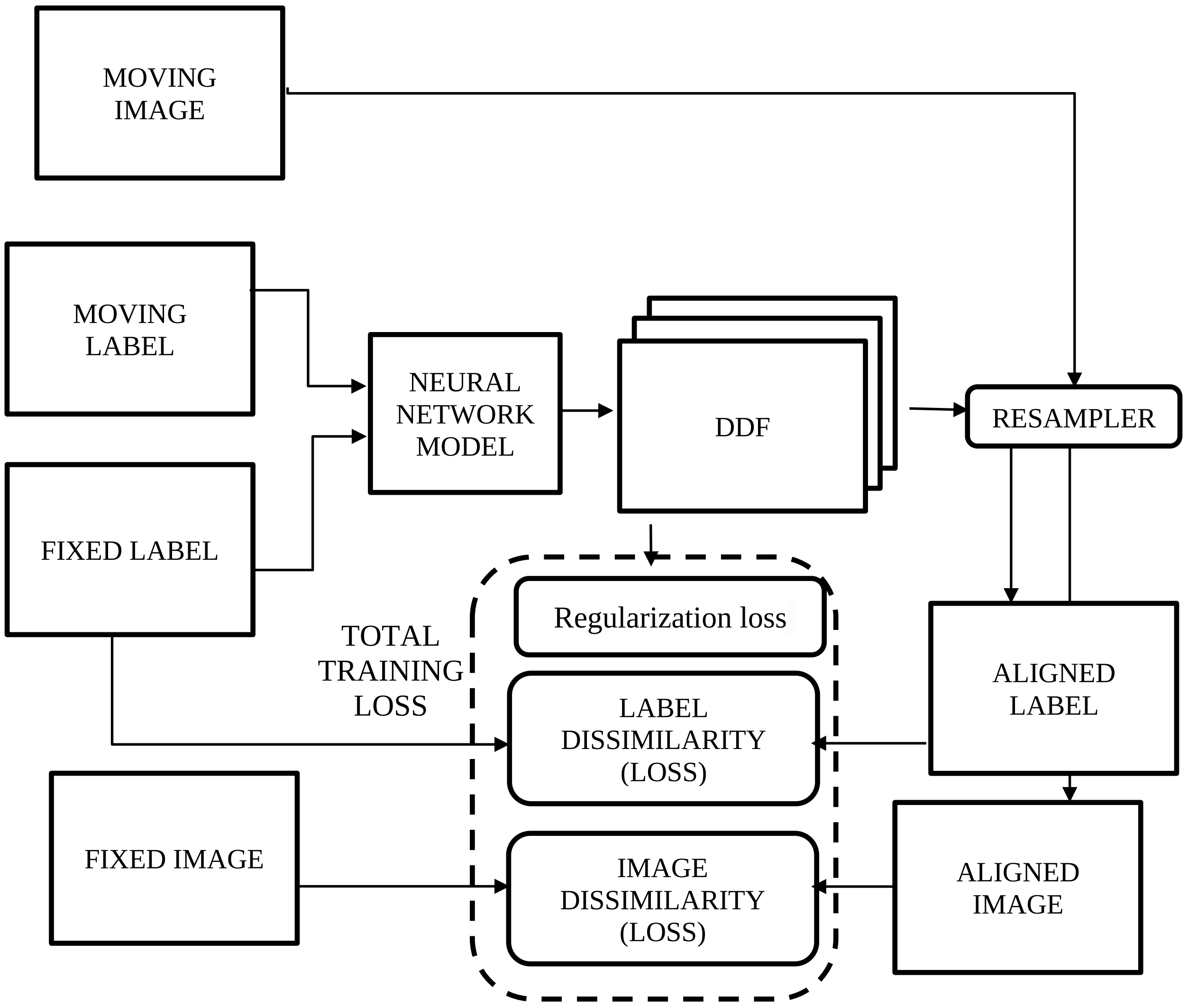
4.2 Image Supervised Learning

Here labeled data is available at the time of training and DDF is calculated with the help of moving and fixed image. Labels are used to calculate the feature-based training loss that measures the dissimilarity between fixed label and moving warped lab. The labels are the regions of interest in the image, in medical images this can be any segmented biological features such as lung mask, mitochondria mask, brain mask and so on.



4.3 Label Supervised Learning

Here labeled data is available at the time of training and DDF is calculated with the help of moving and fixed image. Labels are used to calculate the feature-based training loss that measures the dissimilarity between fixed label and moving warped label. The labels are the regions of interest in the image, in medical images this can be any segmented biological features such as lung mask, mitochondria mask, brain mask and so on.



4.4 Loss

1. Image loss: Mean square error loss is calculated for image dissimilarity between fixed image and aligned image. The loss is a measured by squaring the difference between the fixed image and warped image and then averaging it out.

$$MSE_{loss} = \frac{1}{n} \sum_{i=1}^{n} \left(Y_i - \hat{Y}_i\right)^2$$

2. Label loss: This loss is a measure of dissimilarity between fixed label and aligned moving label. To calculate this difference dice score is computed.

$$Dice\ Score = \frac{2(fixed_{label} \cap predicted_{label})}{|fixed_{label}| + |predicted_{label}|}$$

$$Dice_{loss} = 1 - Dice\ Score$$

3. Regularization loss: This loss is added to any machine learning or deep learning problems to prevent overfitting. This loss will impose a cost on the model to make unique optimal solution.

$$Total_{loss} = \ MSE_{loss} + Dice_{loss} + \lambda\ Regularization_{loss}(ddf)$$

CHAPTER 5

DATASETS

In this thesis research we have used two different datasets to perform experiments.

The first dataset includes two-dimensional electron microscopy (EM) images with mitochondria mask data as labeled mask data. This dataset is used in the field of connectomics and is called Lucchi++. The dataset was the result of study ‘Fast Mitochondria Detection for Connectomics ‘. Originally, we had 160 image tiles of (768, 1024, 3) shape with same orientation. To investigate image registration process using these EM slices, I used OpenCV and TensorFlow to rotate these image slices and there corresponding mask images at arbitrary angles to get an unaligned image stack. A pad of 250-pixel value was added on each side of the image and its mask image so that when rotated at any angle between $(-\pi, +\pi)$ no information is lost in that process.

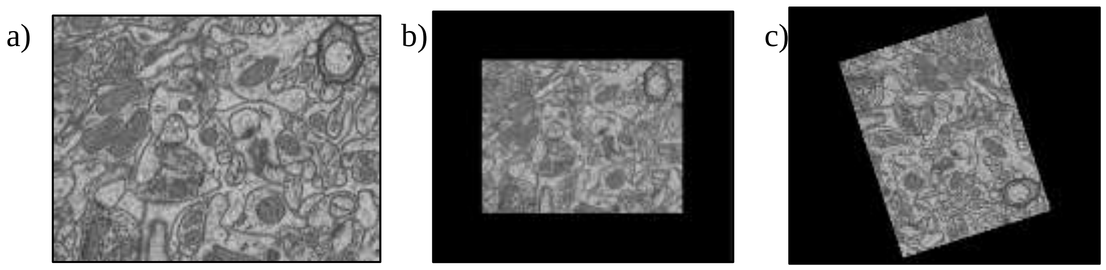


Figure 4: a) Original Lucchi++ image, b) Pad size of 250 px is added to original image, c) Unaligned input data for feature matching method

Finally, an input dataset of 160 image and 160 mask tiles of (1268, 1524, 3) shape was created for image alignment process.

Another dataset consists of three-dimensional high-resolution computed tomography (CT) images and its lung mask data as labeled image. These CT images are acquired for the same patient at different times. These are inhale and exhale thorax images of the same subject and the dataset includes 20 inhale 3D images and 20 exhale 3D images. This dataset was a part of Learn2Reg 2020 challenge.

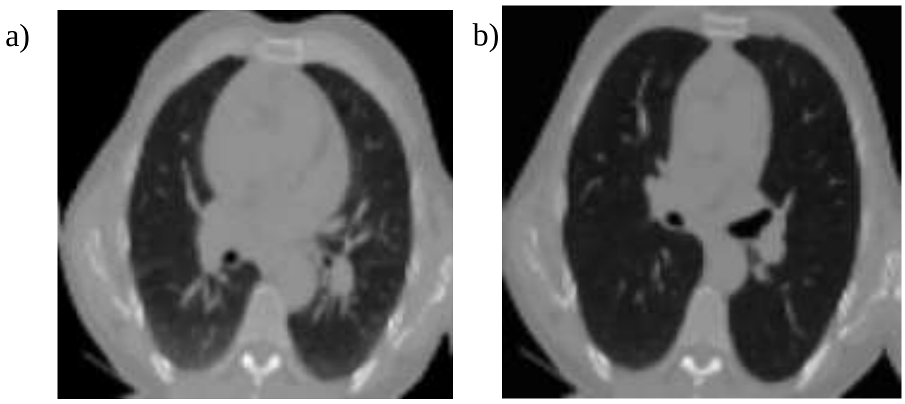


Figure 5: 3D input dataset acquired across time for same patient for deep learning image registration method. a) 3D image captured when subject inhaling, b) 3D image captured when subject exhaling.

CHAPTER 6

EXPERIMENTS

Two different experiments were performed one included state-of-art feature matching alignment method, other included deep learning image registration method.

6.1 Case 1

For feature matching alignment process we are using HP laptop with intel(R) core (TM) i5-1035G1 CPU with 12GB RAM. A pairwise 2D alignment was performed on unaligned EM images, the dataset consists of 160 images and 160 mask data. SIFT, FAST, ORB, BRISK and MITO were used to detect keypoints in this experiment. SIFT, ORB, BRISK and FREAK were used as descriptors.

Keypoints were matched using BF and FLANN matcher throughout the experiments. Only the first ten best matches are used to compute the transformation matrix for all the pre-defined feature detections algorithms. For MITO all keypoints that were matched were used in matrix calculations. This was done to because all the other pre-defined keypoints detection algorithm stored more information such as scale, octave level, orientation with each interest point, with MITO we have coordinate points information only. Two types of transformation matrix are calculated for this experiment, affine and homography matrix. Affine matrix calculation is done with SIFT only. This was to check how fast the pairwise alignment is executed with both the matrices. For larger part of the experiment, pairwise registration is executed with the help of homography matrix. These experiments were executed to investigate the alignment process with and without mask data. Guided and unguided approaches were applied to all the experiments using feature algorithms. In unguided approach, no mask data (mito) was provided at the time of keypoint detection and computation. Transformation matrix was computed with the help of keypoints matched between source image and target image. This matrix was applied to source image and source mask data to get the required warped or aligned image and mask image. In guided approach, additional mask data was provided at the time of keypoint detection and computation.

To validate and make the necessary comparisons, dice score, total execution time (in seconds) and throughput for the whole stack is measured.

Dice score is the measure between the similarity between ground truth mask image and the aligned source mask image. For this below is the pseudocode:

```
begin
   compute_dice(ground_label, predicted_label):
     product = np.dot(ground_label.flatten(),
predicted_label.flatten())
     dice_num = 2 * product
      dice_den = ground_label.sum() + predicted_label.sum()
     diceScore = dice_num / dice_den
end
```

6.2 Case 2

In this experiment, Monia image registration method *LocalNet()* was used. The input dataset consists of high-resolution 3D CT images, 18 inhale/exhale images were used in training set and 2 inhale/exhale images were used as testing set. The experiment consists of moving image or label and fixed image or label such that inhale images were considered as moving or source and exhale as fixed or target.

Additional transformations were applied on the training and testing set for better understanding of CT data. Each training data is randomly rotated and scaled for a given range so that transformation matrix can be calculated to complete the alignment process. The batch size is one and 200 epochs are run in this experiment. A warp image or label is predicted with the given DDF with warp block. Image dissimilarity loss, label dissimilarity loss (dice loss) and regularization loss (BendingEnergyLoss) are measure in each epoch. Dice metric is computed for testing dataset while the 200 epochs ran. Overall training loss calculation includes:

- $imageUnsupervised_{loss} = image_{loss} + \lambda * regularization_{loss}(ddf)$
- $imageSupervised_{loss}\ or\ labelSupervised_{loss} = image_{loss} + label_{loss} + \lambda * regularization_{loss}(ddf)$

where, $image_{loss} = MSE_{loss}(trainFixed_{image}, trainPredicted_{image})$

$label_{loss} = Dice_{loss}(trainFixed_{label}, trainPredicted_{label})$

All the loss measures while training as well as Dice metric score were computed with the help of pre-defined loss methods and metric calculations in MONAI framework.

CHAPTER 7

RESULTS

Below is the result for state-of-art feature matching alignment method. Four tables are computed for these experiments that includes average dice score, execution time (in seconds) and the throughput (in megapixels per second) for the whole stack.

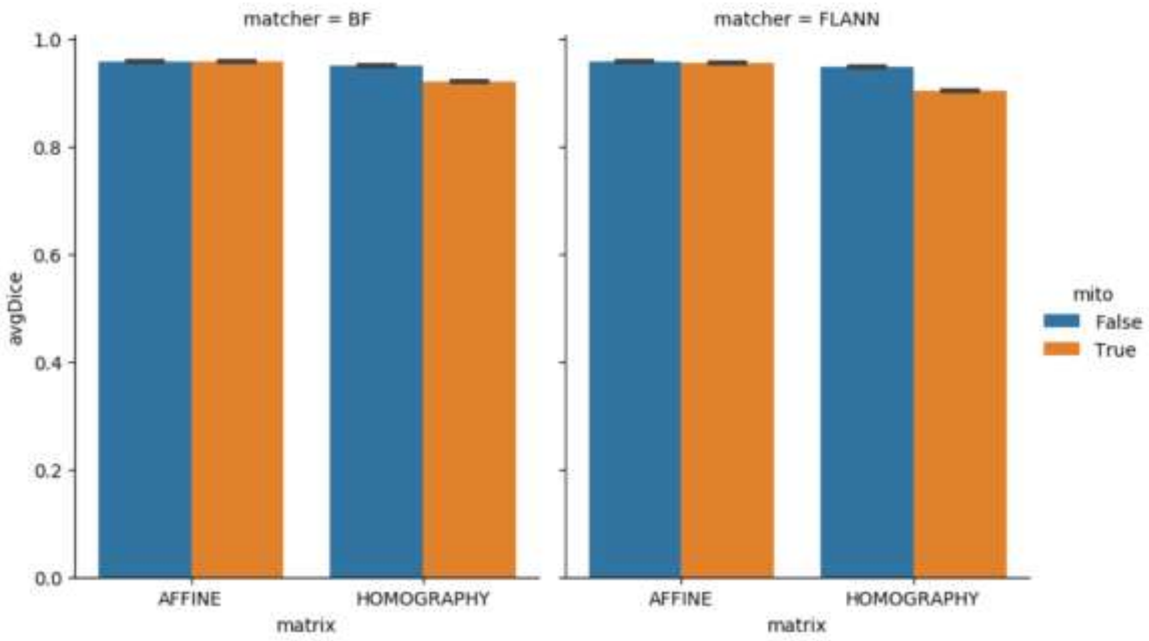


Figure 6: Plots between average dice sore and transformation matrix measured with and without mask data for image alignment using SIFT

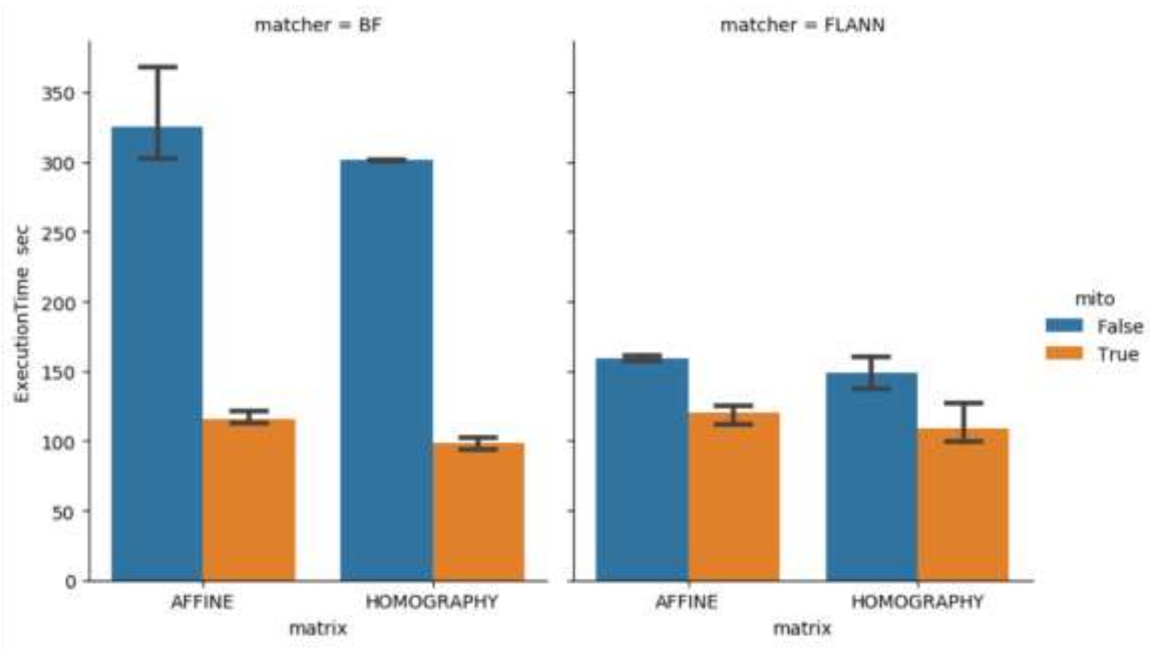
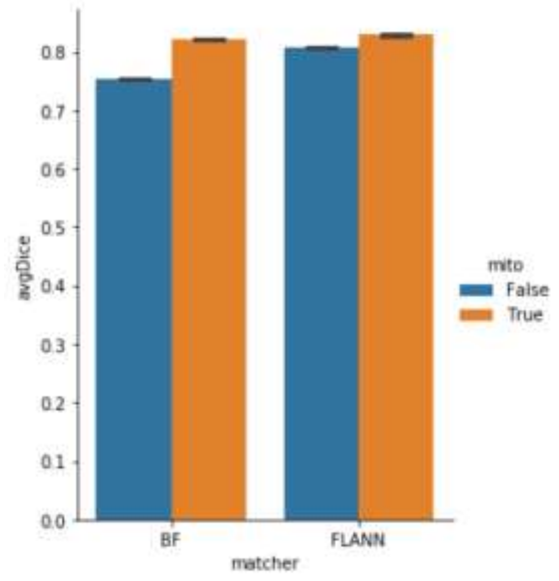


Figure 7: Plots between execution time in seconds and transformation matrix measured with and without mask data for image alignment using SIFT

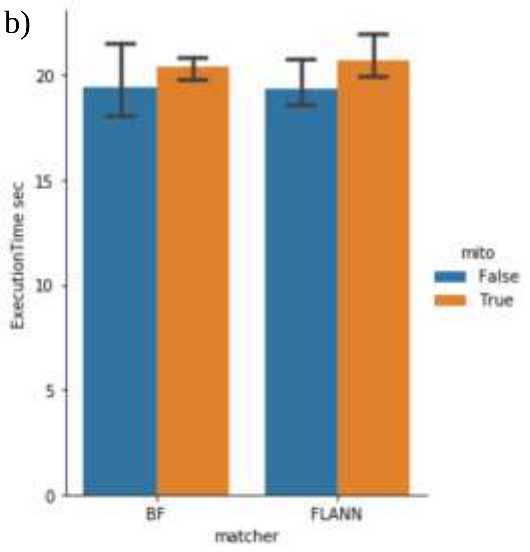
When comparisons drawn for figure 6, we can clearly see that dice score measured is similar for both transformation matrix calculation across both feature matching methods BF and FLANN methods, but figure 7 informs that the execution time is comparatively high when no mask features were added for image alignment. Thus, pairwise registration for the whole stack is completed relatively faster in less than 120 seconds with dice score greater than 0.9 when mask features were added.

In figure 8a, 8b and 8c plots show that dice score measured is higher with mask data for registration than without mask data for ORB algorithm. There is a difference of not more than 2 seconds for process to complete with and without mask in real time with overall throughput greater than 11 MP/sec.

a)



b)



c)

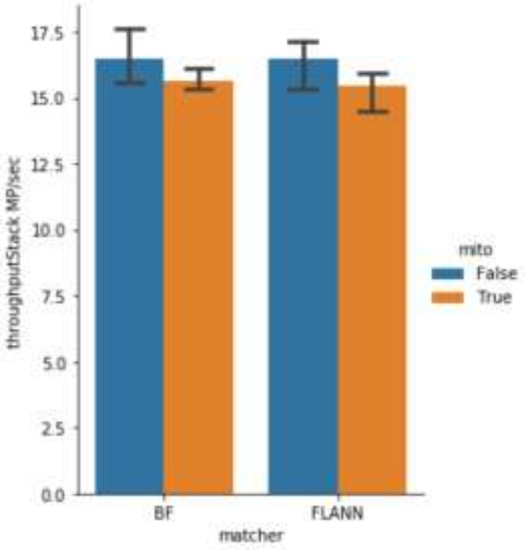
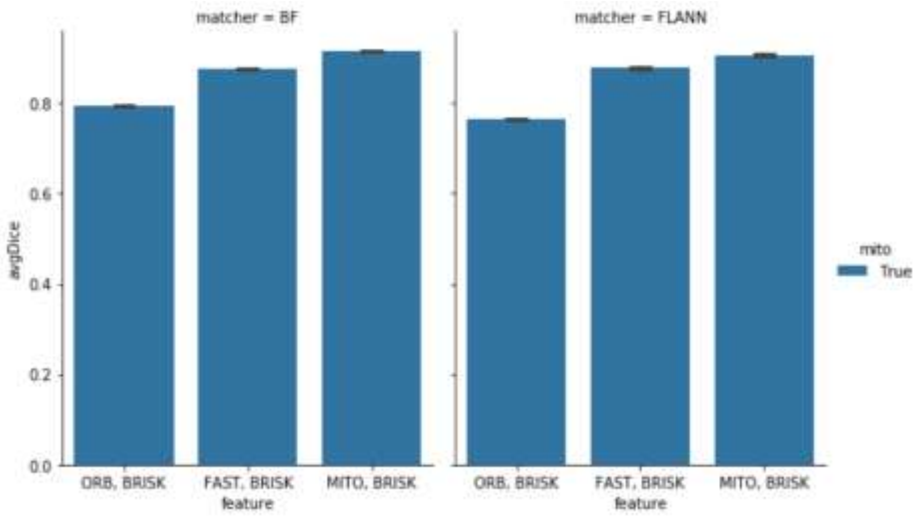


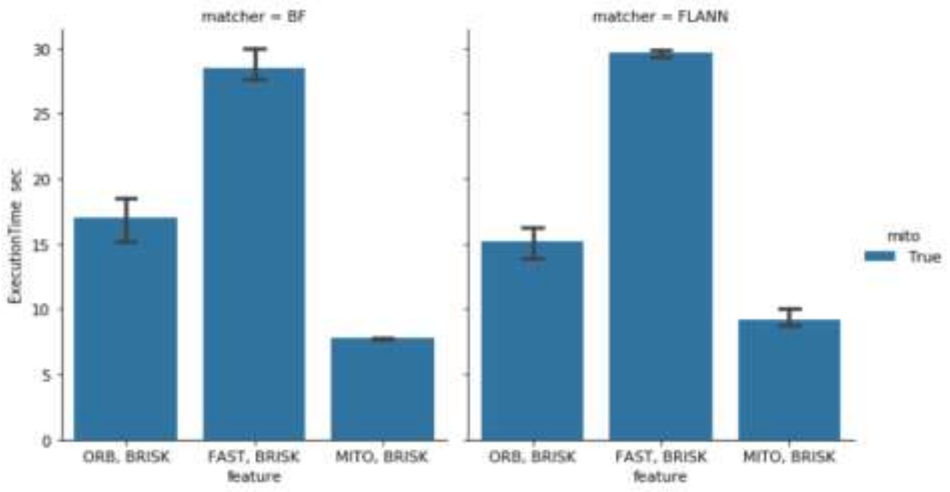
Figure 8: a) Plot between average dice score and matcher (BF, FLANN) computed with and without mask data for image alignment using ORB, b) Plot between execution time in seconds and matcher (BF, FLANN) computed with and without mask data for image alignment using ORB, c) Plot between throughput in megapixels per second and matcher (BF, FLANN) measured with and without mask data for image alignment using ORB

From graph plots in the next figure best results were observed when MITO detector was used in image alignment process. A comparison was drawn between dice score and ORB+BRISK, FAST+BRISK, MITO+BRISK methods when images are mapped with the help of mask data.

a)



b)



c)

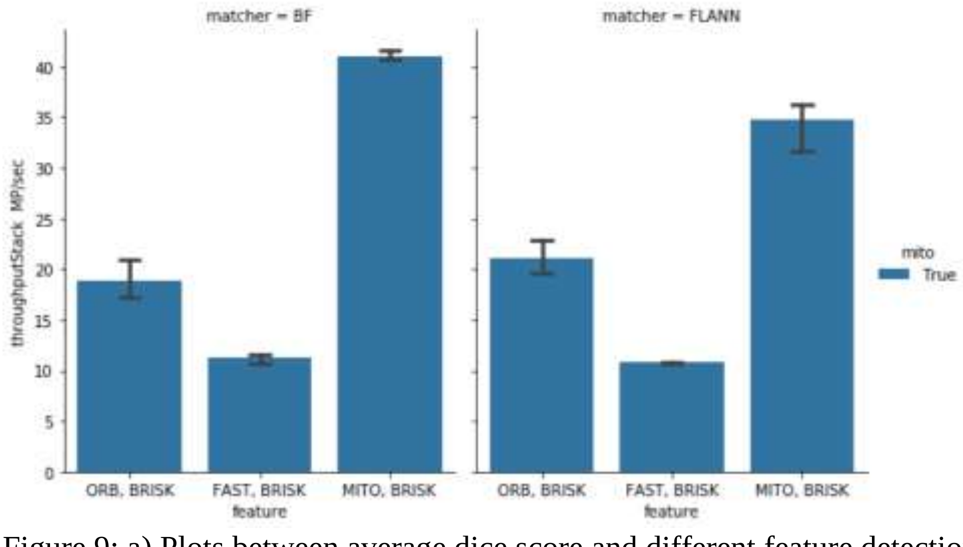
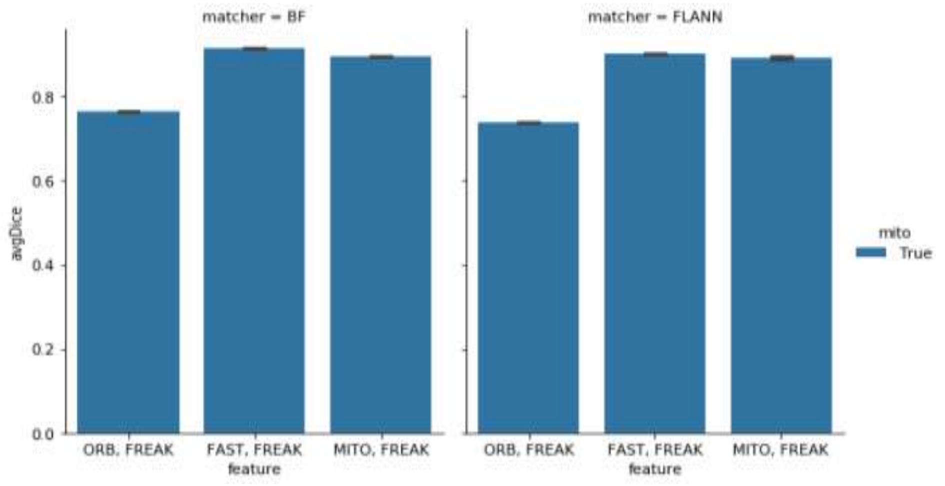


Figure 9: a) Plots between average dice score and different feature detection methods + BRISK with mask data, b) Plots between execution time in seconds and different feature detection methods + BRISK with mask data, c) Plots between throughput in megapixels per second and different feature detection methods + BRISK with mask data

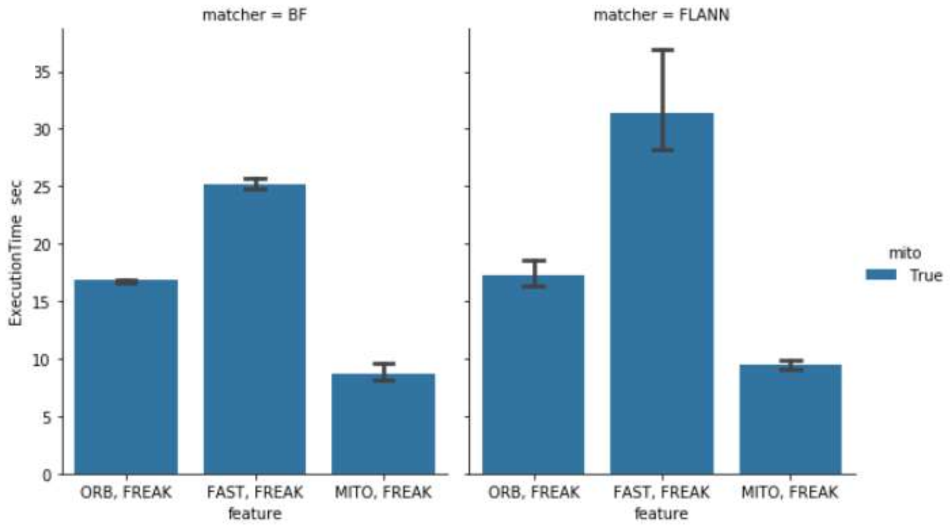
From figure 9 it is observed that MITO detector with BRISK detector registered the images in less than 10 seconds with dice score greater than 0.9 and throughput greater than 30 MP/sec. Both FAST + BRISK and ORB + BRISK takes more time to executes the alignment process in comparison to MITO + BRISK. The overall throughput all the methods is greater than 11 MP/sec that means the alignment process can be carried out in real time.

The next plots in figure 10 show that though the dice score is similar between FAST + FREAK and MITO + FREAK, it is observed that the execution time with MITO detector is less 10 seconds for both BF and FLANN matcher. ORB + FREAK measures dice score less than .75, which is relatively low when compared with the other feature detectors with FREAK The overall throughput for the whole stack measured in alignment process involving mask data, using different feature detection methods with FREAK descriptor is greater than 11 MP/sec. Thus, these pairwise alignments are processed in real time.

a)



b)



c)

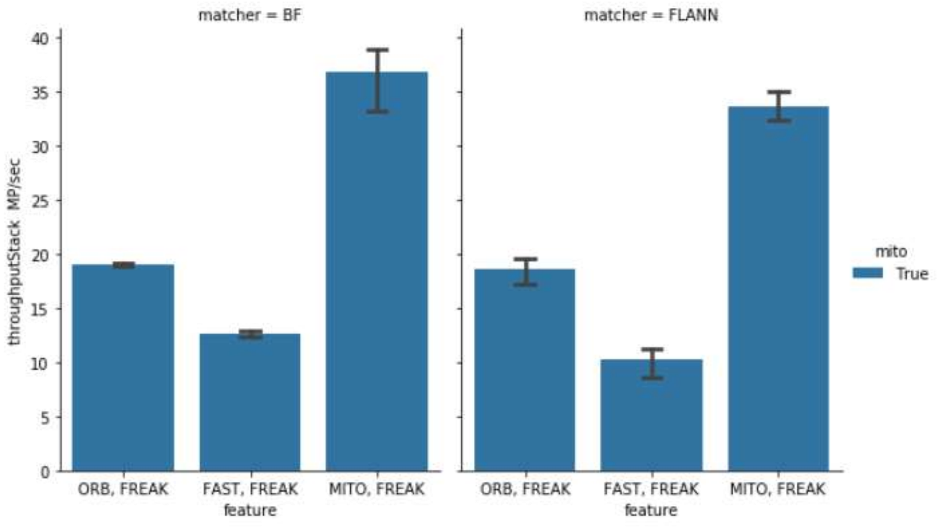


Figure 10: a) Plots between average dice score and different feature detection methods + FREAK with mask data, b) Plots between execution time in seconds and different feature detection methods + FREAK with mask data, c) Plots between throughput in megapixels per second and different feature detection methods + FREAK with mask data

Below are the numerical table for all the experiments carried out to perform pairwise registration using feature-based methods.

Feature	Matcher	Matrix	Mask (mito)	Dice Score	Execution Time sec	Throughput Stack MP/sec
SIFT	BF	Affine	False	.9580	293.0833 (±6.2073)	1.0882 (±0.0232)
			True	.9563	97.1562 (±3.7115)	3.2850 (±0.1244)
SIFT	FLANN	Homography	False	.9496	292.75 (±2.2100)	1.0892 (±0.0082)
			True	.9214	101.2864 (±6.8282)	3.1579 (±0.2211)
SIFT	BF	Affine	False	.9581	137.2291 (±2.4903)	2.3240 (±0.0425)
			True	.9553	99.1562 (±3.3026)	3.2179 (±0.1051)
SIFT	FLANN	Homography	False	.9485	146.2031 (±5.5072)	2.1829 (±0.0837)
			True	.9043	106.7552 (±0.8330)	2.9868 (±0.0233)

Table 1: Dice score, execution time (sec), throughput (MP/sec) calculation for image alignment with SIFT algorithm

Table 1 is the matrix calculations for image alignment with and without mask data using SIFT algorithm with two different feature matching methods. Figures 6 and 7 are plotted with the help of table 1. The next two tables inform the dice score, execution time and throughput measured for pairwise registration using a mix of feature detection and description algorithms. Table 2 shows the numerical calculations when mapping is completed using BF matcher and table 3 shows the numerical calculations when mapping is completed using FLANN matcher.

Feature	Mask (mito)	Dice Score	Execution Time sec	Throughput Stack MP/sec
BRISK	False	.9354	47.0052 (±1.5173)	6.7879 (±0.2170)
	True	.8569	19.3020 (±0.2625)	16.5210 (±0.2256)
ORB	False	.7529	19.4427 (±1.8462)	16.4941 (±1.4953)
	True	.8226	20.4218 (±0.5493)	15.6208 (±0.4259)
FAST + BRISK	False	.9184	2419.9270 (±99.9857)	0.1319 (±0.0053)
	True	.8762	28.4635 (±1.2776)	11.2167 (±0.4908)
ORB + BRISK	False	.6291	16.3020 (±1.4923)	19.6693 (±1.8124)
	True	.7935	16.9687 (±1.6858)	18.9180 (±1.9290)
FAST + FREAK	False	.9405	2391.9479 (±137.7484)	0.1335 (±0.0074)
	True	.9140	25.1302 (±0.5)	12.6912 (±0.2498)
ORB + FREAK	False	.8320	16.6458 (±1.8088)	19.2979 (±1.9733)
	True	.7637	16.8072 (±0.1365)	18.9718 (±0.1545)

Table 2: Dice score, execution time (sec), throughput (MP/sec) calculation for image alignment with feature algorithm using BF matcher

Feature	Mask (mito)	Dice Score	Execution Time sec	Throughput Stack MP/sec
BRISK	False	.9344	40.1145 (±0.9393)	7.9514 (±0.1887)
	True	.8338	19 (±2.4111)	16.9513 (±2.0058)
ORB	False	.8069	19.3802 (±1.2145)	16.4941 (±0.9979)
	True	.8280	20.6875 (±1.1149)	15.4417 (±0.8082)
FAST + BRISK	False	.9338	3082.2343 (±130.2627)	0.1035 (±0.0043)
	True	.8784	29.6041 (±0.2350)	10.7709 (±0.0856)
ORB + BRISK	False	.6297	16.9322 (±1.7772)	18.9655 (±1.9261)
	True	.7648	15.2031 (±1.1735)	21.0579 (±1.6571)
FAST + FREAK	False	.9450	2628.3229 (±32.5343)	0.1213 (±0.0015)
	True	.9091	31.4166 (±4.7502)	10.2940 (±1.4380)
ORB + FREAK	False	.8285	16.2812 (±0.0563)	19.5841 (±0.0676)
	True	.7402	17.2083 (±1.2107)	18.5882 (±1.2665)

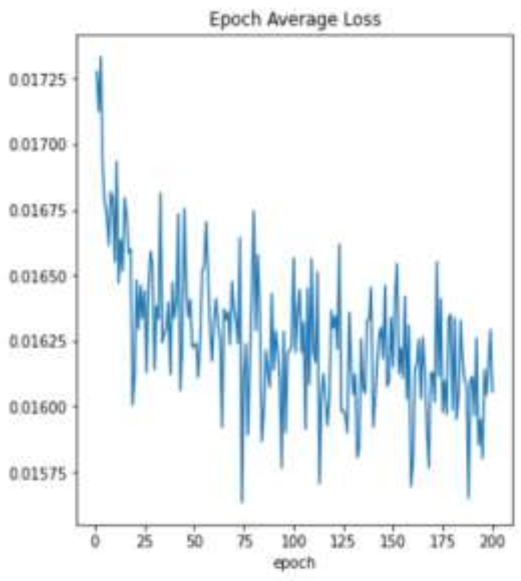
Table 3: Dice score, execution time (sec), throughput (MP/sec) calculation for image alignment with feature algorithm using FLANN matcher

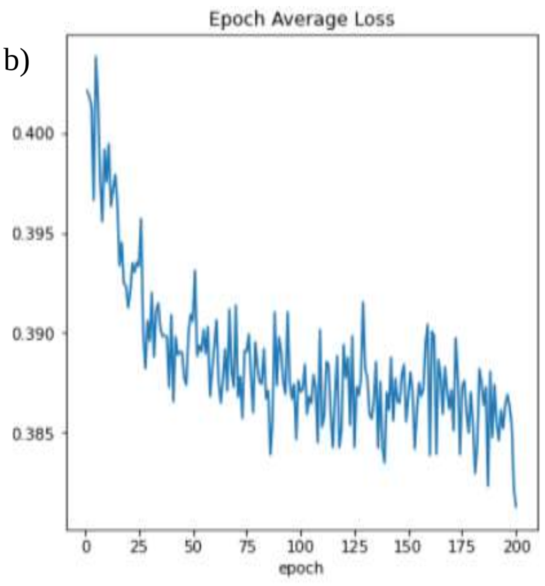
The next table gives the matrix calculation for MITO detector with BRISK and FREAK as descriptor algorithms are shown. Other algorithms such as ORB, SIFT when used to create descriptors for MITO keypoints, no good alignment between source and target images was observed. The dice score throughout the table for both BF and FLANN matcher is greater than 85%. The proves that the alignment is better. The execution time taken to complete this alignment is less than 12 seconds with overall throughput is greater than 30 MP/second. Thus, in real the if electron microscope images are captured such that they are unaligned then the registration process to align these images in a stack is carried in at max 12 seconds.

Feature	Matcher	Dice Score	Execution Time sec	Throughput Stack MP/sec
MITO + BRISK	BF	.9142	7.7708 (±0.0888)	41.035 (±0.4713)
	FLANN	.9062	9.2239 (±0.7265)	34.7050 (±2.6154)
MITO + FREAK	BF	.8963	8.3697 (±0.0888)	38.0983 (±0.4027)
	FLANN	.8928	9.4843 (±0.3694)	33.6528 (±1.3213)

Table 4: Dice score, execution time (sec), throughput (MP/sec) calculation for image alignment with MIYO feature detection method

The next alignment results are with deep learning registration using MONAI framework. Total training loss for each epoch is compared between unsupervised and supervised learning in figure 11. It is to be noted that since only sum of image loss and regularization loss is computed for unsupervised learning, there is a scale difference in total training loss with respect to each epoch in figure 11a. We can clearly see in the three graph plots in figure 11, total training loss is decreasing with each epoch. Thus, performance of model learning with train data is increasing with an exponential decrease in training loss. Next, it is observed that the decrease in training loss with each epoch are similar for both image supervised and label supervised registration methods.

a)


b)


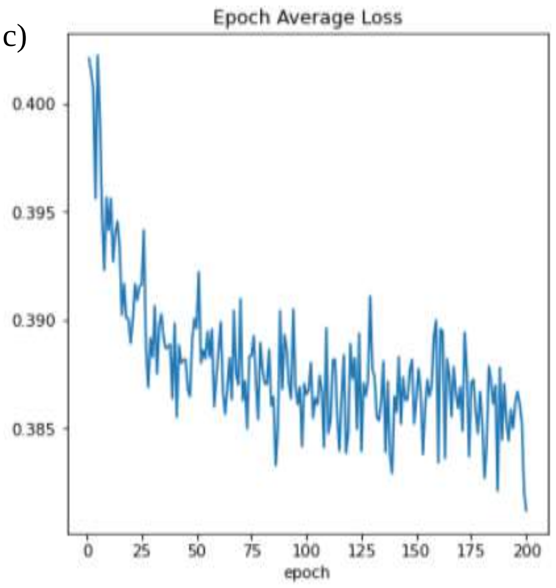
c)


Figure 11: a) Epoch average loss with image unsupervised registration method, b) Epoch average loss with image supervised registration method, c) Epoch average loss with supervised registration method

The next figure shows the dice matrix computed for predicted labels in test data. As we can clearly see that mean dice score is decreasing in figure 12a, thus the performance of image registration is worst in unsupervised learning. On comparing figure 12b and 12c it is observed that the dice score is increasing with each epoch which means that the performance is better in both image supervised learning and label supervised learning. Also, from plot 12c it is observed that the dice score measured with label supervised learning is greater than image supervised learning. Thus, the overall performance of image registration is best in label supervised learning.

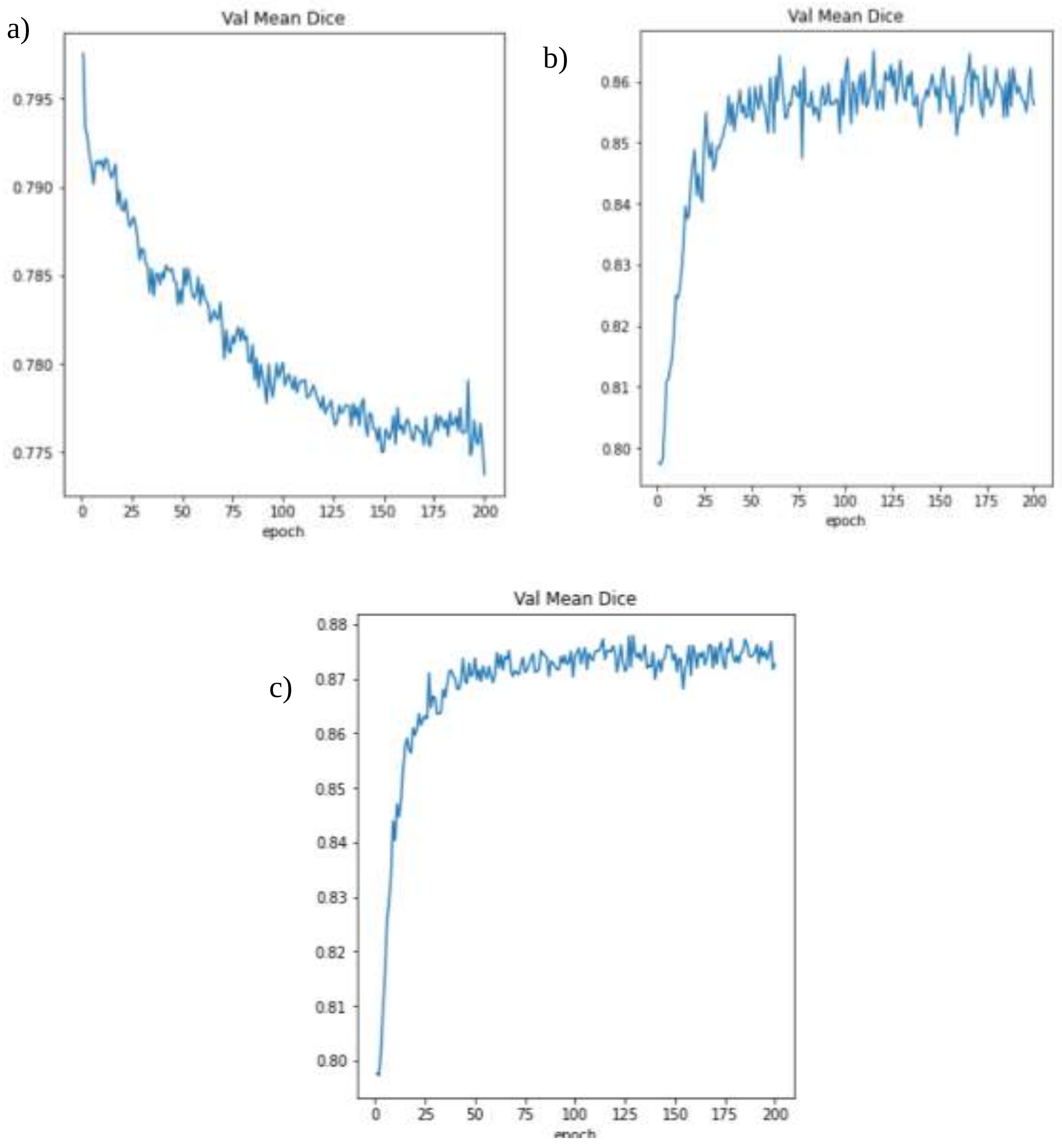


Figure 12: a) Mean dice score with image unsupervised registration method, b) Mean dice score with image supervised registration method, c) Mean dice score with supervised registration method

CHAPTER 8

CONCLUSIONS

From results of two different registration methods, we can conclude that with guided approach and supervised learning with labeled mask we gain better alignment scores.

Our takeaway from results for image alignment based on feature matching methods is:

- though the dice score computed is similar or slightly lower when registration compared between with and without labeled mask data, the execution time is much faster for the guided approach.
- overall throughput is always greater than 11 MP/sec for most of the image alignments computer with labeled mask data.
- the best results were observed with MITO feature detector, the complete registration process was finished in less than 12 seconds, with a dice score always greater than 0.89 and throughput greater than 33 MP/sec.
- MITO detector performs exceptional and can render aligned images in real time. Thus, the time to acquire images and compute the pairwise alignment is processed parallel in the real time.

Conclusion from image registration performed with deep learning models in MONAI:

- the overall training loss is exponentially decreasing for all, image unsupervised method, image supervised method and label supervised method.
- the dice matrix computed on test data; the alignment performance is better for label supervised method than image supervised method. The best diced score recorded for label supervised is 0.8779.

Both the registration methods show that guided approach for mapping with feature matching methods and label supervised deep learning method gives better alignment results.

All the data and source codes are available on GitHub: mito/image_registration at main · nehagoyal1994/mito (github.com)

REFERENCES

D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision, 2004.*

Stefan Leutenegger, Margarita Chli, Roland Y. Siegwar. BRISK: Binary Robust invariant scalable keypoints. *IEEE International Conference on Computer Vision, 2011.*

Yipeng Hu, Marc Modat, Eli Gibson, Wenqi Li, Nooshin Ghavami, Ester Bonmati, Guotai Wang, Steven Bandula, Caroline M. Moore, Mark Emberton, Sébastien Ourselin, J. Alison Noble, Dean C. Barratt, Tom Vercauteren. Weakly-Supervised Convolutional Neural Networks for Multimodal Image Registration. *Medical Image Analysis, 2018.*

Vincent Casser, Kai Kang, Hanspeter Pfister, Daniel Haehn. Fast Mitochondria Segmentation for Connectomics. *Medical Imaging with Deep Learning (MIDL'20), 2020.*

Ethan Rublee, Vincent Rabaud, Kurt Konolige, Gary R. Bradski. ORB: an efficient alternative to SIFT or SURF. *International Conference on Computer Vision, 2011, 2564-2571*

Sevada Khachikian, Mehran Emadi. Applying FAST & FREAK Algorithms in Selected Object Tracking. *International Journal of Advance Research in Electrical, Electronics and Instrumentation Engineering, 2016.*

Learn2Reg Challenge: CT Lung Registration - Training Data | Zenodo (https://zenodo.org/record/3835682#.XsUWXsBpFhE)

Gil's Computer vision blog (https://gilscvblog.com/2013/11/08/a-tutorial-on-binary-descriptors-part-4-the-brisk-descriptor/)

Image Registration: From SIFT to Deep Learning | Sicara (https://www.sicara.ai/blog/2019-07-16-image-registration-deep-learning)

FREAK · ImageFeatures (juliaimages.org)

Image Feature Extraction: Traditional and Deep Learning Techniques | by Krut Patel | Towards Data Science (https://towardsdatascience.com/image-feature-extraction-traditional-and-deep-learning-techniques-ccc059195d04)

DeepReg — DeepReg documentation
(https://deepreg.readthedocs.io/en/latest/index.html)

Connectomics (google.com) (https://sites.google.com/view/connectomics/)

FAST Algorithm for Corner Detection — OpenCV-Python Tutorials (https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_fast/py_fast.html)

Rotation of image using tensorflow (https://developpaper.com/an-example-of-random-rotation-of-image-using-tensorflow/)

Project MONAI — MONAI 0.8.0rc2 Documentation
(https://docs.monai.io/en/latest/index.html)

Contour Properties — OpenCV 3.0.0-dev documentation (https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_contours/py_contour_properties/py_contour_properties.html#contour-properties)

seaborn.catplot — seaborn 0.11.2 documentation
(https://seaborn.pydata.org/generated/seaborn.catplot.html)