UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

MEDICAL SIGNALS ALIGNMENT AND PRIVACY PROTECTION USING BELIEF PROPAGATION AND COMPRESSED SENSING

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

AMINMOHAMMAD ROOZGARD
Norman, Oklahoma
2014

MEDICAL SIGNALS ALIGNMENT AND PRIVACY PROTECTION USING BELIEF PROPAGATION AND COMPRESSED SENSING

A DISSERTATION APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY

Dr. Samuel Cheng, Chair

Dr. Pramode Verma, Co-chair

Dr. James Sluss

Dr. Hong Liu

Dr. William Ray

Acknowledgements

I would never have been able to finish my dissertation without the guidance of my committee members and my wife.

I would like to express the deepest appreciation to my advisor Dr. Samuel Cheng for his guidance, caring, patience, and providing me with an excellent atmosphere for doing research. I would like to thank Dr. Pramode Verma for his advice and financially supporting my research. His guidance has made this a thoughtful and rewarding journey. I would like to thank my committee members, Dr. Hong Liu, Dr. William Ray and Dr. James Sluss for their support to finish this dissertation in the last five years.

I would like to thank all the students and staff of the school of electrical and computer engineering, Tulsa campus, in particular, Mrs. Renee Wagenblatt for her support and help and patiently reading and correcting my writings.

Words can not express my thanks to my parents and my brother. They always encouraged me to follow my dreams in my life, advised me to make the right decisions, supported me in my journey of learning and prayed for my success. I also would like to thank my family in law for their support and belief in me.

Finally, I would like to thank my lovely wife, Nafise. She worked with me, even harder than I did everyday and was always close to me to cheer me up and stand by me through every event of my life, both good ones and bad ones.

Table of contents

List of Tables

List of Figures

Abstract

The advance in human genome sequencing technology has significantly reduced the cost of data generation and overwhelms the computing capability of sequence analysis. Efficiency, efficacy and scalability remain challenging in sequence alignment, which is an important and foundational operation for genome data analysis. In this dissemination, I propose a two stage approach to tackle this problem. In the preprocessing step, I match blocks of reference and target genome sequences based on the similarities between their empirical transition probability distributions using belief propagation. I then conduct a refined match using our recently published SCoBeP technique. I extract features from neighbors of an input nucleotide (a genome sequence of neighboring nucleotides that the input nucleotide is its middle nucleotide) and leverage sparse coding to find a set of candidate nucleotides, followed by using Belief Propagation (BP) to rank these candidates. Our experimental results demonstrated robustness in nucleotide sequence alignment and our results are competitive to those of the SOAP aligner and the BWA algorithm .

In addition, Most genomic datasets are not publicly accessible, due to privacy concerns. Patients genomic data contains identifiable markers and can be used to determine the presence of an individual in a dataset. Prior research shows that the re-identification can be possible when a very small set of genomic data is released. To protect patients, the data owners impose an application and evaluation procedure which often takes months to complete and limits the researchers. One solution to the problem is to let each data owner publish a set of pilot data to help data users

choose the right datasets based on their needs. The data owners release these pilot data with the noise parameters and the mechanism that they used. A data user can run any kind of association tests and compare the outcomes with the other datasets outputs to get an idea which datasets can be useful. I present a privacy preserving genomic data dissemination algorithm based on the compressed sensing. In my proposed method, I am adding the noise into the sparse representation of the input vector to make it differentially private. It means I find the sparse representation using using the SubSpace Pursuit and then disturb it with sufficient Laplasian noise. I compare my method with state-of-the-art compressed sensing privacy protection method.

CHAPTER 1

INTRODUCTION

Advances in DNA information extraction techniques have led to huge sequenced genomes from organisms spanning the tree of life. This increasing volume of genomic information requires Algorithms that can accurately compare multiple genome sequences to aid in the study of populations, pan-genomes, and genome evolution [5,6]. For a particular research, many individual genomes may be sequenced to investigate genetic diversity. For example, the Cancer Genome Atlas [10] and 1000 Genomes Project [11] will generate genome sequences from several thousand people. The complete bacterial genomes in public databases are already over one thousand.

In this dissertation, I propose a novel nucleotide sequence Indexing and alignment method based on empirical transitional probability, sparse coding and belief propagation to compare the similarity of the nucleotide sequences. The alignment method of this dissemination is inspired by my recent works, 3D-SCoBeP described in chapter 2. Thanks to the sparse representation, my mechanism can handle long sequences with reduced memory footprint. I also leverage belief propagation to combine local and neighboring information of candidate nucleotides into consideration and generate matching scores to determine the best match. First, I index the reference and the read genome sequence using empirical transitional probability and pick the top score indexes from the reference genome sequence to build an over–complete dictionary. I then find a set of candidate nucleotide for each nucleotide of the test sequence using sparse coding from the constructed dictionary.

In addition, Most genomic datasets are not publicly accessible, due to privacy concerns. Prior research shows that the re-identification can be possible when a very small set of genomic data is released. To protect patients, the data owners release these pilot data with the noise parameters and the mechanism that they used. I present a privacy preserving genomic data dissemination algorithm based on the compressed sensing. In my proposed method, I am adding the noise into the sparse representation of the input vector to make it differentially private. In this chapter, I will briefly review four core techniques need in this dissemination: Sparse Coding(SC), Compressed Sensing(CS), Belief Propagation(BP) and Differential Privacy(DP).

1.1 Belief Propagation (BP)

Belief Propagation (BP) is an efficient inference method used on graphical models such as factor graphs [12], Bayesian networks [13] and Markov random fields [14]. It was performed by passing messages through the factor graph of my problem. A factor graph is a graph that represent a function of multiple variables factors into a product of multiple functions with few variables. For example, a function $h(x_1, x_2, x_3, x_4)$ can be written as

$$h(x_1, x_2, x_3, x_4) = f_a(x_2) f_b(x_1, x_2, x_3) f_c(x_2, x_4) f_d(x_4) \tag{1.1}$$

where f_a, f_b, f_c ,and f_d are factor functions and the corresponding graph is shown in Fig. 1.1.

Define $N(i)$ and $N(a)$ as two sets of neighbors of a variable node i and a factor node a, respectively, and denote $m_{i\to a}$ and $m_{a\to i}$ as the forward and backward messages from node i to node a, respectively. A message itself is a vector containing current beliefs of a node mapping to all candidate pixels in the reference image.

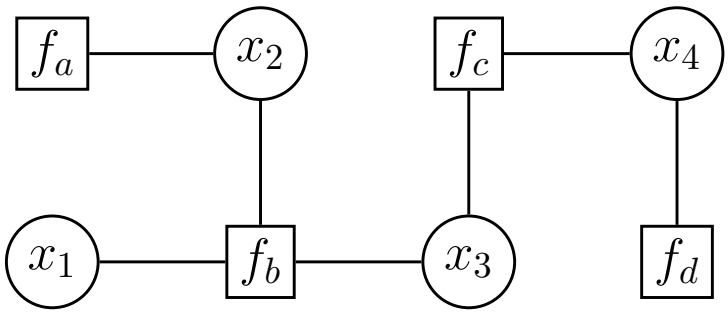


Figure 1.1: Corresponding factor graph of the equation (1). x_1, x_2, x_3, and x_4 are the variable nodes and f_a, f_b, f_c, and f_d are the factor nodes.

For example, $m_{a\to i}(g_i)$ can be interpreted as the belief of node a of how probable that the pixel of node i in the test image should map to location x_i in the reference image. Message updates for $m_{i\to a}$ and $m_{a\to i}$ are based on the messages received by the incoming messages towards nodes i and a, respectively. More precisely, in my factor graph, the message update rules are given by [12]

$$m_{i\to a}(x_i) = \prod_{b\in N(i)\backslash a} m_{b\to i}(x_i), \tag{1.2}$$

$$m_{a\to i}(x_i) = \sum_{x_a\backslash x_i} f(x_a) m_{j\to a}(x_j), \tag{1.3}$$

where $N(a)\backslash i$ means all neighbors of node a excluding node i; the factor node x_a is located between variable nodes x_i and x_j. Also, I model $f(x_a)$ as follows:

$$f(x_a) = \tilde{f}(x_i, x_j) = e^{-\frac{||\mathcal{L}_i - \mathcal{L}_j||_2}{\sigma^2}} \tag{1.4}$$

where σ^2 is a parameter to control the relative strength of the geometric constraint imposed by a neighboring node. If I increase the value of σ^2, the belief of each variable node will have less effect on its neighbors.

1.2 Sparse Coding

Consider a signal $y \in \mathbb{R}^M$ and a fat matrix $D \in \mathbb{R}^{M \times N}$, where I say the matrix is "**fat**" since $M \ll N$. I am interested in representing y with the column space of $D \in \mathbb{R}^{M \times N}$, i.e., finding $\alpha \in \mathbb{R}^N$ such that $y = D\alpha$. Since D is fat, α is not unique. However, if I also restrict α to be the sparsest vector to satisfy $y = D\alpha$ (i.e., α that has fewest number of non-zero elements), then in theory there is a unique solution. Sparse coding precisely considers the aforementioned problem of finding a sparse α such that $y = D\alpha$ is satisfied.

Mathematically, I can write the problem as

$$\hat{\alpha} = arg\ min||\alpha||_0 \ \ subject \ \ to \ \ y = D\alpha. \tag{1.5}$$

However, this l^0 optimization problem is NP-complete [15] and thus several alternative methods have been proposed to solve it [16]. For example, when a sufficiently sparse solution actually exists, substituting the l^1 norm for the l^0 pseudo-norm in (1.5) as below

$$\hat{\alpha} = arg\ min||\alpha||_1 \ \ subject \ \ to \ \ y = D\alpha \tag{1.6}$$

will still result in the same solution [15] . Moreover, solving this modified problem is much easier since it can be readily transformed into a linear programming problem. Besides linear programming, many other suboptimal techniques have been proposed to solve (1.6), including orthogonal matching pursuit [17], gradient projection [18] and subspace pursuit [19].

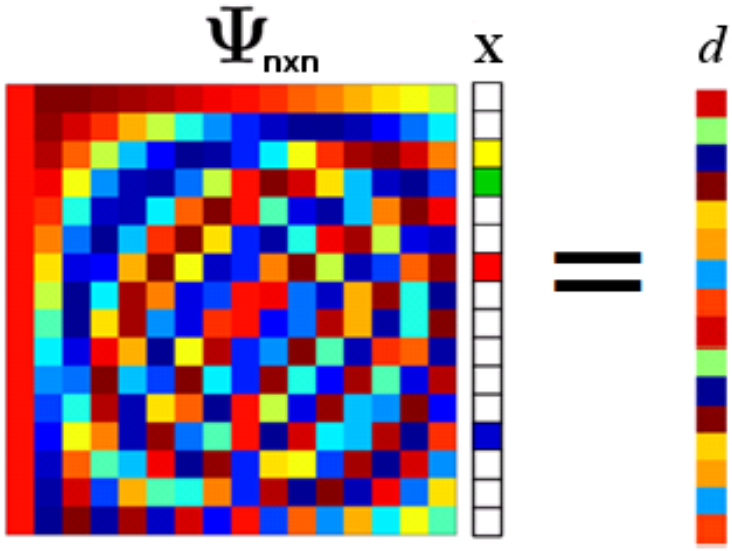


Figure 1.2: The sparse representation of the natural signals; d is a one dimensional signal that can be represented in a transformation domain by a sparse vector x where the transformation basis are the columns of the matrix $\Psi \in \mathbb{R}^{n\times n}$. Note that a vector is sparse if most of its elements are equal to zero. Here, the white squares are representative of the zero elements.

1.3 Compressed Sensing

In this section, I briefly review the theory of the compressed sensing and its major processes and elements[1]. Consider an input vector $d \in \mathbb{R}^n$ that I want to represent it by a vector $x \in \mathbb{R}^n$ using an orthonormal basis (a transformation matrix) $\Psi \in \mathbb{R}^{n\times n}$ where $d = \Psi x \in \mathbb{R}^n$ and x is a s-sparse vector ($s < n$) which means x has at most s nonzero entries (see Figure 1.2).

Note that if x is not a sparse vector, by zeroing the very small coefficients of x, I can make it sparse and this new vector still keeps the most amount of information of original vector [4, 20, 22]. Furthermore, the orthonormal basis Ψ can be a standard transformation basis like wavelet basis or discrete cosine transform basis. The vector d is the input of the compressed sensing method.

The compressed sensing is divide into two processes: a "**sampling process**" and

[1] for more information, please read [20, 21]

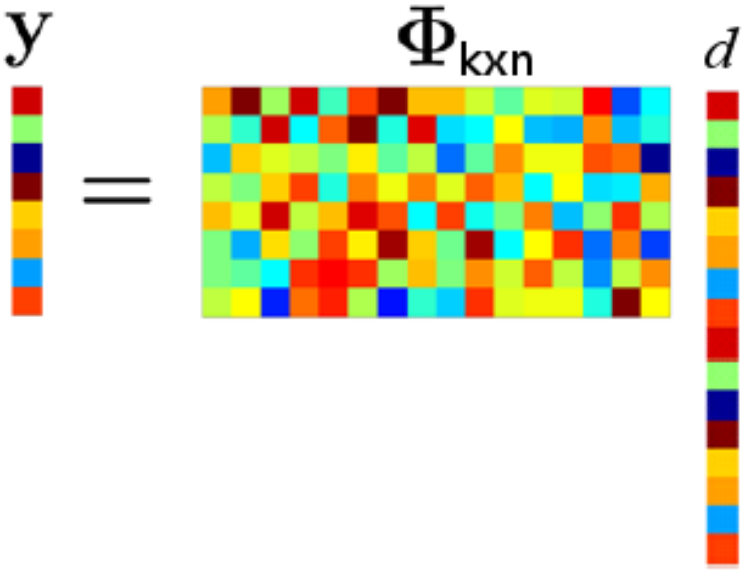


Figure 1.3: The sampling process of the compressed sensing method; d is an input signal and a random matrix $\Phi \in \mathbb{R}^{k \times n}$ maps d to a measurement vector $y \in \mathbb{R}^k$ which reduces the size of the input signal from n to $k = O\big(s \log(n/s)\big)$.

a "**reconstruction process**". The sampling process of the compressed sensing is a probabilistic compassion process using a random matrix $\Phi \in \mathbb{R}^{k \times n}$ that reduces the size of input from n to $k = O\big(s \log(n/s)\big)$. This step is modeled as a linear mapping of the input vector d into its random projection y ($y = \Phi d \in \mathbb{R}^k$). The random matrix Φ is formed by independent and identically distributed (i.i.d.) entries from a symmetric Bernoulli distribution (see Figure 1.3).

The reconstruction process of the compressed sensing exactly or approximately reconstructs the original data from the compressed samples. In this step, the vector d is recovered from its random projection y using the sparse representation $\hat{x}$. I consider a matrix $A = \Phi\Psi$ which both Φ (random matrix) and Ψ (Orthogonal transformation matrix) are known from the sampling process. Using a l_1-Norm minimization method like Orthogonal Matching Pursuit (OMP) [23] or SubSpace Pursuit (SSP) [19], the recovered answer $\hat{x}$ is close enough to the original x even in the presence of noise (see Figure 1.4).

Mathematically, the l_1-Norm minimizer selects the smallest set of columns from

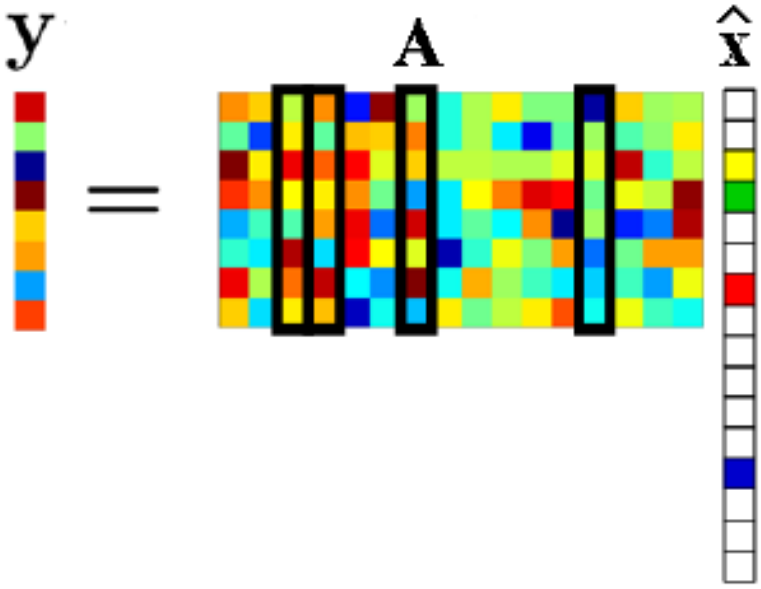


Figure 1.4: The reconstruction process of the compressed sensing method; y is a measurement vector and $A = \Phi\Psi \in \mathbb{R}^{k\times n}$. The l_1-minimizer select the smallest set of A's columns as the solution which is sparse and its error is less than the threshold ϵ. Note that the columns marked by black are the selected ones.

A such that

$$\hat{x} = \arg\ \min\ \|x'\|_0,\ \text{subject to}\ \|y - Ax'\|_2 < \epsilon \tag{1.7}$$

where $\|x\|_0 := |\{i : x_i \neq 0\}|$[2] and ϵ is a error threshold that determines how close is the $\hat{x}$ to the original vector x; The smaller ϵ forces the $\hat{x}$ to be closer to the x. If I want to have a s-sparse $\hat{x}$, then the l_1-Norm minimization method solves the following problem:

$$\hat{x} = \arg \min_{\|x'\|_0 \leq s} \|y - Ax'\|_2. \tag{1.8}$$

1.4 SubSpace Pursuit

Subspace pursuit (SSP) is a l_1-Norm minimization method which has a reconstruction capability compared to the Linear Programming (LP) methods, and has very low reconstruction complexity of matching pursuit techniques for very sparse signals.

[2] $\|x\|_1 := \sum_{i=1}^{n} |x_i|$ and $\|x\|_p := \left(\sum_{i=1}^{n} x_i^p\right)^{1/p}$ where $p > 1$

For any sampling matrix A satisfying the restricted isometry property (RIP) [24] with a constant parameter independent of K, the Subspace pursuit algorithm can recover arbitrary K-sparse signals exactly from its noiseless measurements.

When the measurements are inaccurate and/or the signal is not exactly sparse, the reconstruction distortion is of order a constant multiple of the measurement and/or signal perturbation energy. More precisely, for very sparse signals with $K = O(\sqrt{N})$ where N the number of columns of A, which, for instance, the computational complexity of the Subspace pursuit algorithm is upper bounded by $O(mNK)$, but can be further reduced to $O(mNlogK)$ when the nonzero entries of the sparse signal decay slowly [3].

1.5 Differential Privacy

The current privacy protection techniques attempt to add noise to the allele frequencies of the case group in a way that the absent or present of any individual in the output result data be impossible.

A randomized algorithm f, called ϵ–Differential Private, if for all adjacent datasets $\mathcal{D}$ and $\mathcal{D}'$, and any possible output $\mathcal{D}$ in the output space of f:

$$\frac{Pr[f(\mathcal{D}) = \hat{\mathcal{D}}]}{Pr[f(\mathcal{D}') = \hat{\mathcal{D}}]} \leq e^{\epsilon}. \tag{1.9}$$

Note that $\hat{\mathcal{D}}$ is any dataset or a numerical value depends. The Laplacian mechanism [25] is commonly used in data disturbing methods to achieve differential privacy, which adds noises generated from a Laplacian distribution, Laplace(0, $\Delta f/\epsilon$), to the output of a computation on the dataset. The amount of noises will be calculated based on the sensitivity of the computed data. The sensitivity represents the maximum change of the output when a single modification happens to a dataset.

[3] For more information about the details of the Subspace pursuit, please read [19]

Note that for any $f : \mathcal{D} \to \mathbb{R}^d$, and all adjacent datasets $\mathcal{D}$ and $\mathcal{D}'$, the sensitivity of f can be calculated as follows:

$$\Delta f = \max_{\mathcal{D},\mathcal{D}'} \|f(\mathcal{D}) - f(\mathcal{D})\|_1 \tag{1.10}$$

Another popular differential privacy mechanism is called Exponential mechanism [26]. This mechanism will one output $t \in T$ that has optimum utility function and preserving the differential privacy. The inputs of the exponential mechanism are a data set $\mathcal{D}$, a range T, a privacy parameter ϵ , and a utility function $u(\cdot)$ where $u : (\mathcal{D} \times T) \to \mathbb{R}$. Then it that assigns a real value number to the output $t \in T$, where the higher value number shows the better utility. The mechanism induces a probability distribution over the range of T and

$$t \propto \exp\left(\frac{\epsilon u(\mathcal{D}, t)}{2\Delta(u)}\right), \tag{1.11}$$

where $\Delta u = \max_{\mathcal{D},\mathcal{D}'} \|u(\mathcal{D}) - u(\mathcal{D})\|_1$ is the sensitivity of the utility function $u(\cdot)$.

CHAPTER 2

3D MEDICAL IMAGE REGISTRATION USING SPARSE CODING AND BELIEF PROPAGATION

There are various medical imaging methods which have been used broadly in clinical and medical research. Consequently, the interests in registering and finding similarities of different images for diagnosis, treatment, and the sake of basic science are increasing. As images are typically captured at different times, angles, and often by different modalities, registering (or aligning) one image with another is challenging. In general, The accuracy of registration techniques will affect the performance and robustness of all subsequent analysis. I propose an efficient 3D medical image registration method based on sparse coding and belief propagation for Computed Tomography (CT) and Magnetic Resonance (MR) imaging. I used 3D image blocks as the input features and then I employed sparse coding with a dictionary of the features to find a set of the candidate voxels. To select optimum matches, belief propagation was subsequently applied on a factor graph of voxels generated by these candidate voxels. The outcome of belief propagation was interpreted as a probabilistic map of aligning the candidate voxels to the source voxels. I compared my proposed method (3D-SCoBeP) with the state-of-the-art medical image registration, MIRT [2] and GP-Registration algorithm [3]. My objective results based on Root Mean Square Error (RMSE) are smaller than those from MIRT and GP-Registration. My results prove the effectiveness of my algorithm in registering

the reference image to the source image.

2.1 Introduction

Image registration refers to the process of aligning two or more images obtained from different capturing modules and/or angles, and/or at different times into the same coordination system [27]. Registration is essential in many clinical applications including diagnosis [28], simulating and surgical planning [29]. For example, registration techniques have been used to align a Magnetic Resonance (MR) image to a Computer Tomography (CT) image [30,31]. In surgery, radiotherapy, or radiological intervention, preoperative medical data are used to diagnose, plan, simulate, guide, or otherwise assist a surgeon, or possibly a robot [32]. While the surgical procedure is performed in the coordinate system relative to the patient, the surgical plan is constructed in the coordinate system relative to the preoperative data. The spatial transformation between the plan and the preoperative data is formed by registration. Registration as a central step of processing images in the treatments, allows any voxel defined in the preoperative image to be precisely located in the patient coordinate system. This can aid the surgeon by delineating the position of the surgical instruments relative to the ultimate target.

Images of similar or different modalities need to be aligned for navigation, detection, data-fusion and visualization in medical applications [33]. Medical image registration still presents many challenges. For example, finding a one-to-one correspondence between several scans of the patient is difficult, because the body of the patient can be subject to sudden changes or the modality of the scans can be different. The first one makes the transformations between scans highly non-rigid and the last one creates significantly different images in overall appearance and resolution [34].

Many medical image registration methods have been developed in the last two

decades [2,3,27–40]. These can be divided into two major categories, namely, direct and feature-based matching. Direct methods use all available image data, and they result in very accurate registration if initialization points are close to target points at the beginning of the registration procedure [35]. For instance, in [3], a general-purpose registration algorithm for the medical images has been developed which incorporates both geometric and intensity transformation. The authors modeled the transformation with a local affine model and a global smoothness constraint. Intensity variations are also modeled with local changes in brightness, contrast and a global smoothness constraint. Moreover, Myronenko and Song used the definition of the similarity measure to propose a registration method [2]. They derived the similarity measure by analytically solving for the intensity correction field and its adaptive regularization. The final measure was interpreted as one that favors a registration with minimum compression complexity of the residual image between the two registered images.

Feature-based registration methods, utilize invariant features (specially around Harris corners) to ensure reliable matching. As a result, feature-based methods are independent from an initialization point [37]. Also, feature-based registration methods obtain the transformation parameters from the set of extracted features. For example, Glocker *et al.* [36] used different levels of smoothness in modeling medical images and then used Markov Random Fields (MRFs) to formulate image deformations. Liu *et al.* [41, 42] and Elbakary *et al.* [43] registered multi-modal medical images using banks of local Gabor and Gaussian filters to evaluate the frequencies. The number and characteristics of filters in those works were selected empirically. Staring *et al.* [44] incorporated multiple image features including the intensity gradients and Hessians. They combined parametric cubic B-splines, and an iterative stochastic gradient ascent optimization [45, 46] to solve the registration problem.

Image-registration techniques based on type of deformation have been divided into two categories: "rigid" and "non-rigid". In the rigid techniques, like [38], images are assumed to have rotation and translation only but in the non-rigid techniques, like [47], images can have restricted localized stretching. For example, in brain image registration with different modalities, a rigid body approximation is sufficient due to relatively little changes in brain shape over a short period between scans. In [38], authors formulated the rigid registration problem based on general image acquisition model and cast the problem of finding a similarity measure into their maximum likelihood problem. Then, they derived similarity measures for different modeling assumptions. Their experimental results concentrated on the multi-modal images of the brain. Sabuncu *et al.*, in [48] introduced an entropy-based algorithm for registering rigid multi-modal images that incorporates spatial information. Spatial feature vectors obtained from the images and a minimum spanning-tree approach were used to estimate the conditional higher-dimensional entropy. They minimized the Jensen-Renyi divergence between the learned and new joint intensity distributions with a gradient descent method.

As an example for non-rigid registration techniques, Likar *et al.* [47] proposed a hierarchical image subdivision strategy to perform a non-rigid registration method based on mutual information. The non-rigid matching problem was decomposed into a Thin-Plate-Spline-based (TPS) elastic interpolation of multiple local rigid registrations of sub-images. One of the sub-categories of the non-rigid image registration is topology-preserving registration. In these kind of methods, the existing structures are kept, no new structures are allowed to be added, and neighborhood relationships between the structures are preserved. For example, Musse *et al.* [39] proposed a parametric topology-preserving deformable image registration using the Gauss–Seidel optimization method. The Jacobian of the mapping was controlled over the domain of the transformation to ensure topology preservation. The authors

derived the necessary and sufficient conditions for the determinant of the Jacobian of such transformations to be continuously positive everywhere and applied their method to the 2D images.

Based on the dimensions of input data, the registration method can be categorized as 3D or 2D registration techniques. The 3D registration method normally applies to the registration of two tomographic datasets but the 2D registration may apply to the separate slices from tomographic data. Also, a 3D to 2D registration may help to transfer the acquired 3D data to the 2D data, to facilitate treatment planning. In [49–51], the authors developed automated intensity-based algorithms for updating a 3D position of an interventional instrument using a single-plane angiogram registered to a 3D volume. In [49], Penney *et al.* aligned preoperative CT and intraoperative fluoroscopy images where the surface-target registration errors were of the order of 12 mm. In [50], Hipwell *et al.* expanded the former method to registering 3D cerebral Magnetic Resonance Angiography (MRA) with 2D X-Ray angiograms [50] where their RMSE were 1.5 ± 0.9 mm for 85% of the clinical images. Byrne *et al.*, in [51], extended Penney *et al.* work and registered 3D X-ray Digital Subtraction Angiography (3D-DSA) images. Their registration method accuracy was 1.3 ± 0.6 mm in the clinical study of the two images with the same modality.

Many researchers incorporate smoothness (or spatial coherence) conditions by reformulating matching into an optimization problem [52, 53]. For example, Tang and Chung [53] assigned a vector displacement label indicating the position in the test image to each pixel in the reference image. They used a smoothness constraint based on the first derivative to penalize sharp changes in displacement labels across pixels. Then they employed a graph-cuts method to solve that labeling problem. Moreover, Liu *et al.* [52] used belief propagation to optimize cost function incorporated with smoothness constraints which encourage similar displacements of near-by pixels.

In this chapter, I propose a dense registration technique by aligning two CT or MR images using sparse coding and belief propagation. First, I build an overcomplete dictionary out of all 3D features of a reference image [54]. Note that since the dictionary is constructed by padding the features directly, I only need to normalize each columns. I then find a set of the candidate voxels for each voxel of the source image using sparse coding out of the constructed dictionary. The match score of each candidate voxel will be evaluated taking both local and neighboring information into account using belief propagation [12]. The best match will be selected as the candidate with the highest score. For those voxels with belief less than the threshold θ, I use graph-cuts algorithm [55] to find the proper matches. In comparing to the state-of-the-art belief propagation based registration methods, the key innovation of the proposed approach (3D-SCoBeP) is the inclusion of a preprocessing step to preselect good candidate registration points for each voxel. Belief propagation is very powerful optimization technique, but if the size of the problem increases, it is more difficult to obtain a good local optimum. This restricts the size of the search range for each voxel. In prior approaches such as SIFT-flow [52], the search range is simply chosen as a patch containing neighboring voxels around each target voxel. In contrast, a preprocessing step is used to carefully preselect candidate registration points for each voxel in the 3D-SCoBeP. Since these candidate points are selected from any voxel in the image, the search range of the 3D-SCoBeP is much larger than prior approaches and essentially covers the entire image. This is a main reason for the improvement of the 3D-SCoBeP over the prior works.

A naïve approach computes the Mean Square Error (MSE) of the input patch with each possible patch of the reference image and selects patches that have the smallest MSEs. These kind of approaches have poor diversity which means the candidate patches are concentrated in a small region. In this case, a small shift from the most similar patch generally does not decrease similarities sharply except for very

high frequency patches. Consequently, this approach results in patches with very low diversity. Furthermore, the naïve approach may fail to find the true corresponding match points. It is possible that the naïve approach returns a set of candidates where all of them concentrate around a wrong point. Instead, I propose to find candidate match points using sparse coding. The intuition is that if these candidate patches are similar enough to the source patch, I should be able to construct a source patch out of good candidate patches (so they correspond to a sparse coding solution). With my technique, the sparse coding outputs the patches that can reconstruct the original patch through a linear combination. The resulting patches of sparse coding are likely to be complementary to each other and so provide a better diversity than the naïve solution.

The proposed method described here is inspired by my recent works, SCoBeP [56] to answer the the current challenges in the medical data alignment which are:

1) the patient body movements while capturing the data,

2) the patient body tissue changes due to progress of disease or treatment and

3) different sampling rates of the data because of different sampling rates along the directions.

A preliminary version of this work has been reported in [57]. Since then, much research has been done and the novel components in addition to the aforementioned works are summarized as follows:

- Using 3D feature of input data which makes the proposed method more accurate than the other state-of-the-art methods;

- Using 3D feature of input data which makes the proposed method more accurate than the other state-of-the-art methods;

- Using a template and interpolating the voxels of input data to map onto the template where the captured voxel coordination systems are different which

increases the robustness of registration method;

- Employing a graph-cuts method as a post-processing method which further refines the matches obtained from the belief propagation step.

The rest of the chapter is organized as follows. In the next section, I will introduce the concept of my 3D-SCoBeP and the inference algorithm and then, in Section 2.3, I will show my simulation results, followed by a brief conclusion in Section 2.4.

2.2 3D–SCoBeP

As mentioned in Section 2.1, in the medical image applications I need dense registration so that for each point of the *source data* a corresponding match point will be found in the **reference data**. This section describes the implementation details of my proposed registration method for the 3D medical data which is based on sparse coding and belief propagation.

The proposed method described here is inspired by my recent works, SCoBeP [56]. First, I extract the features from the 3D reference data $\{\mathcal{X}_s\}_{s=1}^{k_x} \in \mathbb{R}^{M\times N\times K}$ and the 3D source data $\{\mathcal{Y}_s\}_{s=1}^{k_y} \in \mathbb{R}^{M\times N\times K}$ where k_x and k_y are the numbers of the reference and the source image slices, respectively. I focus on only using 3D block features even though the proposed approach can generally be applied to other features (such as SIFT-features [58] or Gabor-features [59]). The 3D feature block is a rectangular cube neighbor around each voxel of the 3D data which I reorder as a 1D vector. Thus, each feature considered here is essentially a vectorized 3D block centered around a voxel in a 3D data.

Second, I create a dictionary $\mathcal{D}$ which contains all extracted feature vectors of the reference data to match to the corresponding extracted features of the source data. The dictionary includes all vectorized 1D features as its columns where all of them have been normalized. I then apply sparse coding to each extracted feature of

the source data. Sparse coding will reconstruct a 3D source patch at voxel $[i, j, k]$ as a linear combination of the reference 3D patches. Denote α_{qlv} as the weight vector where each element corresponds to a coefficient in this combination. Note that the representation coefficients α_{ijk} should be sparse, i.e., it should be 0 for most coefficients. To select the n candidate voxels, I simply pick those corresponding to n largest coefficients in the sparse coefficient vector. I denote a set as an $n \times 3$ matrix storing the locations of these candidate voxels and a probability vector ρ as a length-n vector storing the corresponding probabilities of the sparse coefficient vector. Each coefficient in the probability vector ρ_{ijk} serves as a prior probability of matching the 3D source patch centered around voxel $[i, j, k]$ to a 3D patch of the reference data. This probability vector is taking only local characteristics into account but ignoring geometric characteristics of the matches.

Finally, to incorporate these geometric characteristics, I model the problem by a factor graph and apply BP to identify the best matches similar to [60]. I consider a 3D lattice factor graph as follows: For each voxel in the source 3D data, one variable node was assigned and then I connect each variable node to its six neighbors by a factor node (see Fig. 2.1). Also, I consider one extra factor node for each variable node to take care of prior probabilities of the candidate points. In my model, the factor function $f(x_i, x_j)$ which can be interpreted as the local belief mapping nodes i and j to x_i and x_j can be used to impose the geometric constraint described earlier. Intuitively, since x_i and x_j are the corresponding mapped match points in the reference image of two neighboring voxels in the source image, I expect the probability of getting x_i and x_j to decrease as their distance apart increases. Therefore, I model the function of the factor node between two particular variable nodes x_i and x_j with a Gaussian kernal as [12]

$$f(x_i, x_j) = e^{-\frac{||x_i - x_j||_2}{\sigma^2}} \tag{2.1}$$

where σ^2 is a parameter to control the relative strength of the geometric constraint imposed by a neighboring node. If I increase the value of σ^2, the belief of each variable node will have less effect on its neighbors.

To synthesize the source image, I replaced each voxel of the source image with the selected candidate voxel from the reference image where its probability is more than the threshold θ. If the final maximum belief of the selected point was less than the threshold θ, I employ graph-cuts algorithm [55, 61] to find the correspond voxel. First, for voxels with maximum belief more than θ, I calculate the movement of each voxel of reference data in comparison to the source data and create the displacement matrix β. Then, I feed the displacement matrix β to graph-cuts algorithm to estimate the disparity of voxels with unsatisfactory beliefs. In the graph-cuts algorithm, I initialize the label of each voxel by its displacement value if its input belief is small then threshold θ and by average displacement value of all voxels otherwise. The data term is defined by a quadratic function of the distance between the current label and the desired label, and the smoothness term is defined by a linear function of the distance between the current label and its neighborhood label. The neighborhoods are the same as which I used in the BP step and the swap algorithm of graph-cuts is applied to label voxel with beliefs less than the threshold θ.

2.2.1 Implementation

This section describes the implementation's details of my proposed registration method. The main procedure for my proposed registration method is summarized in Algorithm 1.

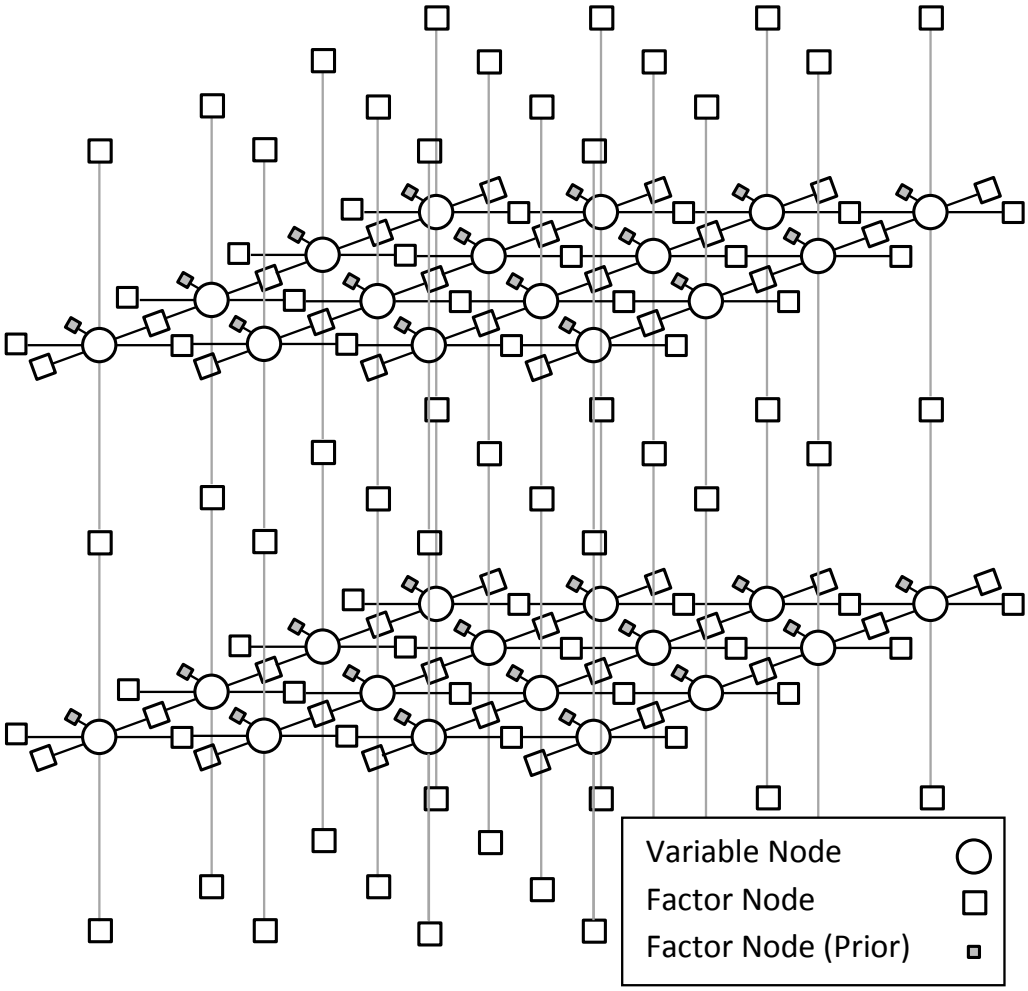


Figure 2.1: Three dimensional factor graph of medical data used in Belief Propagation: for each voxel in the source 3D data, one variable node was assigned to incorporate these geometric characteristics. I connect each variable node to its six neighbors by a factor node and incorporate one extra factor node to store initial probabilities. A part of two slices of medical data corresponding factor graph is shown. This factor graph can be extended on X-axis, Y-axis and Z-axis.

Algorithm 1 3D-SCoBeP for the medical image registration- estimate version of the registered image $\mathcal{Z}$

Inputs: a reference data $\{\mathcal{X}_s\}_{s=1}^{k_x} \in \mathbb{R}^{M\times N\times K}$, a source data $\{\mathcal{Y}_s\}_{s=1}^{k_y} \in \mathbb{R}^{M\times N\times K}$, a threshold θ , the number of the candidate points n

Extract 3D dense feature and construct dictionary:

- $\mathbf{Y} = ExtractDenseFeature\big(\{\mathcal{Y}_s\}_{s=1}^{k_y}\big)$
- $\mathbf{X} = ExtractDenseFeature\big(\{\mathcal{X}_s\}_{s=1}^{k_x}\big)$
- $\mathcal{D} = MakeDic\big(\mathbf{X}\big)$

Find the initial estimate of the candidate voxels: For each vector $y_{i,j,k} \in \mathbf{Y}$ perform:

- $\hat{\alpha}_{i,j,k} = FindSCV\big(\mathcal{D}, y_{i,j,k}\big)$
- $[\mathcal{L}_{i,j,k}, \rho_{i,j,k}] = FindTopSCV\big(n, \hat{\alpha}_{i,j,k}\big)$

Refine the candidate voxels:

- $\hat{\rho} = BP\big(\mathcal{L}, \rho\big)$

Find the correspond voxels:

- $[\mathcal{Z}, \beta] = Warp\big(\mathbf{X}, \hat{\rho}, \mathcal{L}, \theta\big)$
- if there is a voxel with probability less than θ then $\mathcal{Z} = Graphcuts\big(\mathbf{X}, \beta, \theta\big)$

Output: the estimated version of the registered image $\mathcal{Z}$

Implementation Details:

- $\mathbf{Y} = ExtractDenseFeature\big(\{\mathcal{Y}_s\}_{s=1}^{k_y}\big)$ presents a 3D block extractor algorithm using $\{\mathcal{Y}_s\}_{s=1}^{k_y}$ as a source data. More precisely, I consider a 3D block of size $S = (2a+1) \times (2b+1) \times (2c+1)$ containing neighboring voxels around each voxel on a 3D data, where a, b and c are positive integers. For each voxel $p_{i,j,k}$ in the source data $\{\mathcal{Y}_s\}_{s=1}^{k_y}$, I vectorized the 3D block centered around the voxel $p_{i,j,k}$ to a feature vector $y_{i,j,k} \in \mathbb{R}^{S\times 1}$. A source feature $\mathbf{Y} \in \mathbb{R}^{M\times N\times K\times S}$ is then constructed from $y_{i,j,k,t}$ as follows:

$$\begin{aligned}\mathbf{Y} = \{y_{i,j,k,t} \mid \quad & 1 \le i \le M, \quad 1 \le j \le N, \\ & 1 \le k \le K, \quad t \in S\}. \end{aligned} \tag{2.2}$$

  Note that $\mathbf{X}$ is created in the same manner as $\mathbf{Y}$ but instead from the reference data $\{\mathcal{X}_s\}_{s=1}^{k_y}$.

- $\mathcal{D} = MakeDic\big(\mathbf{X}\big)$ creates a dictionary $\mathcal{D}$ using the vectors of $\mathbf{X}$. Later, the dictionary $\mathcal{D}$ is used to match the extracted features of the source data to the corresponding extracted features of the reference data. I can write $\mathcal{D}$ as

$$\mathcal{D} = \left[x_{1,1,1} \ ... \ x_{1,1,K} \ x_{1,2,K} ... \ x_{1,N,K} ... \ x_{M,N,K}\right], \tag{2.3}$$

  where $x_{i,j,k}$ is a feature vector of $\mathbf{X}$. Note that I normalize dictionary $\mathcal{D}$ to guarantee the norm of each feature vector to be 1.

- $\hat{\alpha}_{i,j,k} = FindSCV\big(\mathcal{D}, y_{i,j,k}\big)$ finds the candidate match voxels using the sparse coding algorithm, where $\hat{\alpha}_{i,j,k}$ is a sparse vector. Mathematically, I try to solve the following sparse coding problem to find the most sparse coefficient vector

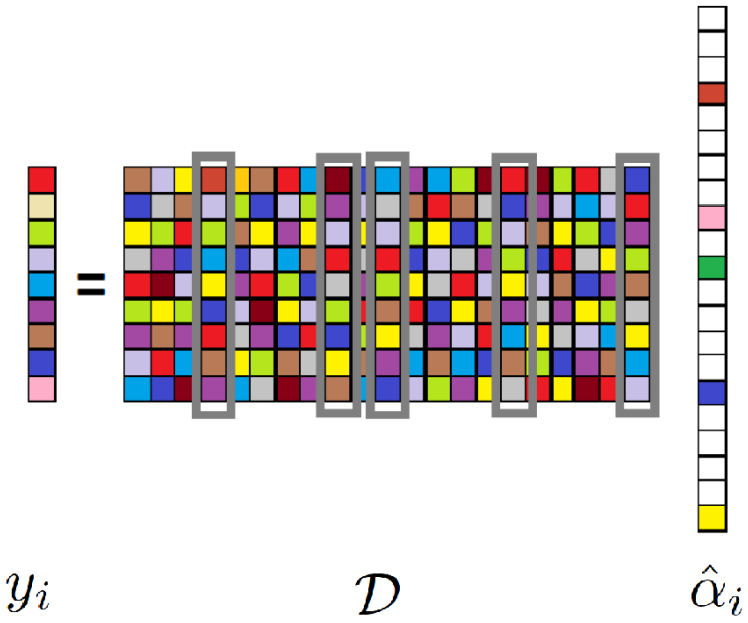


Figure 2.2: Sparse representation of a feature vector $y_{i,j,k}$ with a dictionary $\mathcal{D}$: $\hat{\alpha}_{i,j,k}$ as a sparse vector constructs the feature vector $y_{i,j,k}$ using a few columns (highlighted in gray) of dictionary $\mathcal{D}$.

$\hat{\alpha}_{i,j,k}$ (see Fig. 2.2) such that

$$y_{i,j,k} = \mathcal{D}\hat{\alpha}_{i,j,k}. \tag{2.4}$$

Although there are several methods to solve (2.4) [17–19], in my work, I employ Subspace Pursuit (SP) [19] because of its computational efficiency.

- $[\mathcal{L}_{i,j,k}, \rho_{i,j,k}] = FindTopSCV\left(n, \hat{\alpha}_{i,j,k}\right)$ picks up the n largest coefficients of $\hat{\alpha}_{i,j,k}$ as n candidates. $\mathcal{L}_{i,j,k}$ as an $n \times 3$ matrix stores the locations of these candidate voxels and $\rho_{i,j,k}$ as a length-n vector stores the corresponding values of $\mathcal{L}_{i,j,k}$. Each coefficient in $\rho_{i,j,k}$ serves as a prior probability of matching the source patch at $[i, j, k]$ to a patch centered around the voxel $x_{i,j,k}$. After finding the candidate locations $\mathcal{L}_{i,j,k}$ and their initial probabilities $\rho_{i,j,k}$ for each voxel,

I concatenate the results and construct following matrices:

$$\mathcal{L} = [\mathcal{L}_{1,1,1} \ ... \ \mathcal{L}_{1,1,K} \ \mathcal{L}_{1,2,K} ... \ \mathcal{L}_{1,N,K} ... \ \mathcal{L}_{M,N,K}], \tag{2.5}$$

$$\rho = [\rho_{1,1,1} \ ... \ \rho_{1,1,K} \ \rho_{1,2,K} ... \ \rho_{1,N,K} ... \ \rho_{M,N,K}] \tag{2.6}$$

which I will use to apply belief propagation at the next step.

- $\hat{\rho} = BP\big(\mathcal{L}, \rho\big)$ models the problem by a factor graph and applies belief propagation [12] to update probability ρ. The updated probability $\hat{\rho}$ can be used to register the reference data onto the source data. In my case, I assign a variable node for each voxel in the source data and connect each pair of neighboring voxels with a factor node. Also, I introduce one extra factor node to take care of the prior knowledge obtained in the sparse coding step for each voxel of the source data (for more details, see [56]).

- $[\mathcal{Z}, \beta] = Warp\big(\mathbf{X}, \hat{\rho}, \mathcal{L}, \theta\big)$ returns the registered image $\mathcal{Z}$ and a displacement matrix β which contains the movement of each voxel of reference data. This matrix can be used to refine the result of the voxels with a probability less than θ.

- $\mathcal{Z} = Graphcuts\big(\mathbf{X}, \beta, \theta\big)$ applies the graph-cuts method to find the displacement of the voxels with a probability less than θ and returns the registered image $\mathcal{Z}$. The displacement matrix β keeps the movement of the voxels and marks the area with a probability less than θ. I feed the matrix β to graph-cuts algorithm and initialize the label of each voxel by the average displacement value of all voxels if it is marked in the matrix β and by its displacement value otherwise. I use a quadratic function of the distance between the current label and the desired label as the data term and a linear function of the distance between the current label and the neighboring label as the smoothness term. The

graph-cuts algorithm updates the matrix β and I use the new displacement matrix to find the corresponding voxels.

- $\hat{\mathcal{Z}} = Warp\big(\{\mathcal{X}_s\}_{s=1}^{k_y}\ ,\ \hat{\rho}\ ,\ \mathcal{L}\big)$ displays the registered CT image $\hat{\mathcal{Z}}$ using the updated probabilities $\hat{\rho}$, the candidate voxels location $\mathcal{L}$, and the reference CT image $\{\mathcal{X}_s\}_{s=1}^{k_x}$. In my work, I select the most probable point after the BP step as the best match point. I assume that my registration method successfully finds a match for an input point if the most probable candidate has belief larger than a threshold θ. Otherwise, I assume no best match is found.

2.3 Experimental Results

The utility and novelty of my medical image registration algorithm lies in the fact that it can handle images captured not just from a single plane but also from different planes. Hence, in this section, I will study the performance of my method for both cases and compare it with the different registration methods.

In a brief statement, I will present two experiments in this section: *the 3D CT image registration taken along a same direction* in Section 2.3.1 where I consider the problem of registering two lung CT images of one person from two different times, and *3D MR image registration taken along different directions* in Section 2.3.2 where two brain MR images were captured along the X–Z (sagittal) and the X–Y (transverse) planes. I implemented the 3D-SCoBeP algorithm in Matlab and tested it on a Pentium 3 GHz (11-GB RAM) machine.

2.3.1 3D CT Image Registration Taken In Same Directions

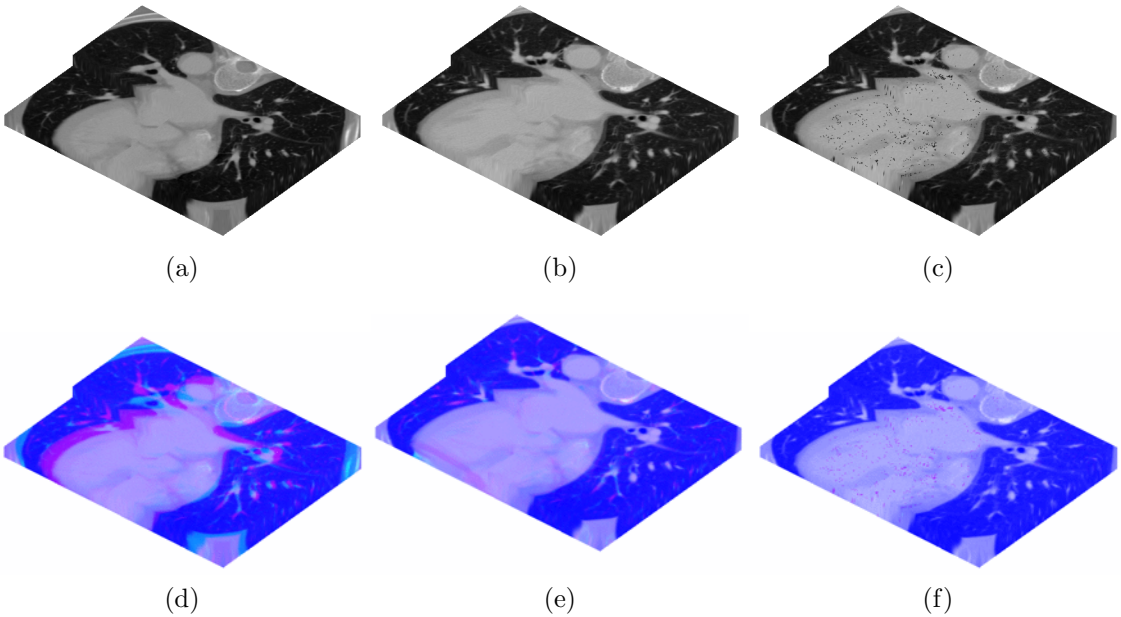


Figure 2.3: Result of 3D-SCoBeP on Lung CT images. (a) The reference CT image; (b) The source CT image; (c) The 3D-SCoBeP result; (d) The comparison between corespondent voxel between the source and the reference; (e) The comparison between corespondent voxel between the source and the MIRT result; (f) The comparison between corespondent voxel between the source and the 3D-SCoBeP result; In (d)-(e) I used a RGB image where the first channel of the image was assigned to the source image intensity and the second channel to the reference, MIRT, 3D-SCoBep results, respectively.

To evaluate the performance of my approach, I conducted tests on the data sets LIDC-IDRI [62] where the size of each slice of the CT images is 512×512 voxels. Throughout the experiments, the following parameters were used: the number of the candidate voxels n is set to be 4, $a = b = 3$ and $c = 2$. To synthesize the source image, I replaced each voxel of the source CT image with the selected candidate voxel from the reference CT image. In other words, I map the reference CT image onto the source image using the updated probabilities and the candidate voxels location. In my work, I select the most probable voxel after the BP step as the best match voxel. I assume that my registration method successfully finds a match for an input voxel if the most probable candidate has belief larger than a threshold $\theta = 0.3$. Otherwise, I assume no "best match" is found. The threshold θ can be chosen empirically which was the way that I chose in Fig. 2.4 and 2.5 to express the results.

Fig. 2.3 shows the result of the 3D-SCoBeP and MIRT [2] with a 3D perspective. In this figure, I decided to show only a part of the CT images because the inside details of the lung are more important than the tissue around it. Figs. 2.3(a) and 2.3(b) are the reference and the source CT image, respectively. Fig. 2.3(c) shows the result of 3D-SCoBeP where I used the voxel of the reference data to synthesize the source data. I created one RGB image where its first channel was assigned to the intensity of source CT image and its third channel was equal to 255. I assigned the reference CT image intensity, the MIRT result and the 3D-SCoBeP result to the second channel, respectively. Therefore, Fig. 2.3(d) corresponds to the initial state and Fig. 2.3(e) and 2.3(f) are final state of the MIRT and the 3D-SCoBeP. Note that in Fig. 2.3(c), I display a pure result of the proposed method which only the voxels with a probability more than the threshold θ was shown, therefore there are some dark voxels in this figure. For those voxels with probability less than θ, the analysis of motion fields of neighborhood voxels could be used to estimate their motions. It

means, one can extract the motions of voxels which have probability more than θ in each direction and apply the graph-cuts method [55], median filter, or moving average to estimate the motion of voxels with probability less than threshold θ.

I now proceed to compare the 3D-SCoBeP with other approaches; Figs. 2.4 and 2.5 show the output of my proposed method compared to two of the state-of-the-art methods: the MIRT [2] and GP-Registration [3]. In these figures, I select only one slice of CT image to show the weaknesses and strengths of each technique. Figs. 2.4(a) and 2.5(a) correspond to the reference CT image and Figs. 2.4(b) and 2.5(b) correspond to the source CT image. Figs. 2.4(c)-(e) and 2.5(c)-(e) show results using MIRT [2], GP-Registration [3] and 3D-SCoBeP. The warped images using MIRT with highlighted artifacts are shown in Figs. 2.4(h) and 2.5(h) and the warped images using GP-Registration with highlighted artifacts are shown in Figs. 2.4(i) and 2.5(i). The estimated images generated from the 3D-SCoBeP with highlighted areas are shown in 2.4(j) and 2.5(j).

To quantify my registration performance, I used the root mean square error (RMSE) measure between the true and estimated transformations:

$$\varepsilon_{RMSE} = \sqrt{(1/N)\sum \|\tau - \hat{\tau}\|^2}, \tag{2.7}$$

where N is the number of voxels in the reference and τ and $\hat{\tau}$ are the source image and the estimated transformation respectively.However, the RMSE is insufficient to qualify the accuracy of the registration methods. It can only give a rough estimation of similarity between the estimated image and the source image [63,64]. In the term of the RMSE, I compare the source image with the output of the 3D-SCoBeP, MIRT and GP-Registration and the results are shown under (c), (d) and (e) of Figs. 2.4 and 2.5. However, in Fig. 2.4, the RMSE values for the MIRT and GP-Registration are 24.31 and 28.78, respectively. Although, The MIRT generates the result with the lower RMSE value, the GP-Registration preserves the structures better. My

proposed technique shows an improved qualitative and quantitative result. The 3D-SCoBeP preserves the structure as precise as GP-Registration and also has the less RMSE value, 21.73, in comparison to two other methods. In Fig. 2.5, all three methods show the same structure preservation but the RMSE value of the 3D-SCoBeP is much less than the RMSE value of MIRT and GP-Registration.

2.3.2 3D MR Image Registration Taken In Different Directions

In this section, I am trying to align the medical data shown in Figs. 2.6 and 2.7 where the brain MR images captured in parallel to the X–Z (sagittal) and the X–Y (transverse) planes, respectively. The X-axis is from left to right along the column direction in Fig. 2.6 and is from anterior to posterior along each slice in Fig. 2.7. The Z-axis and Y-axis are from the first slice to the last slice along the plane direction in Figs. 2.6 and 2.7, respectively.

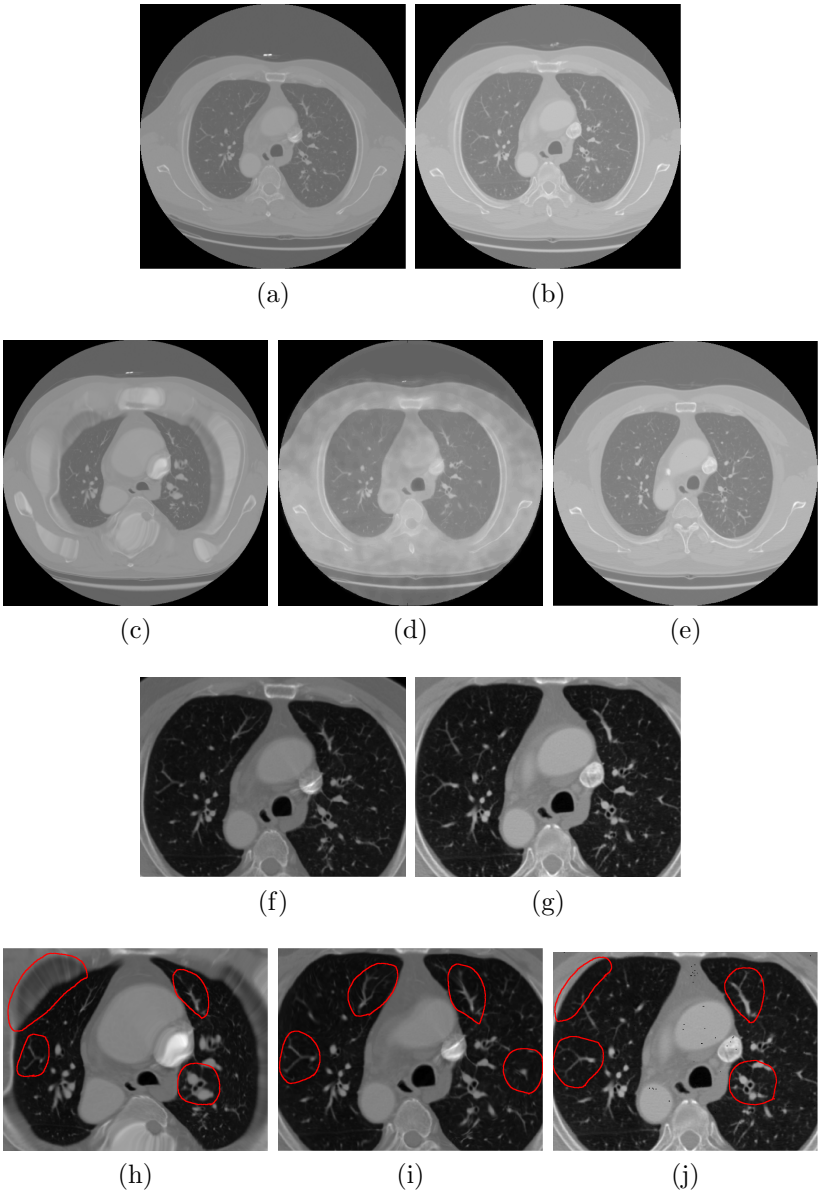


Figure 2.4: Registration result of the lung CT images that were captured with six months gap. (a) Source image; (b) Reference image; (c) MIRT [2] [RMSE: 24.31]; (d) GP-Registration [3] [RMSE: 28.78]; (e) 3D-SCoBeP [RMSE: 21.73]; (f) Source image (zoom in); (g) Reference image (zoom in); (h) MIRT [2] (zoom in); (i) GP-Registration [3] (zoom in); (j) 3D-SCoBeP (zoom in).

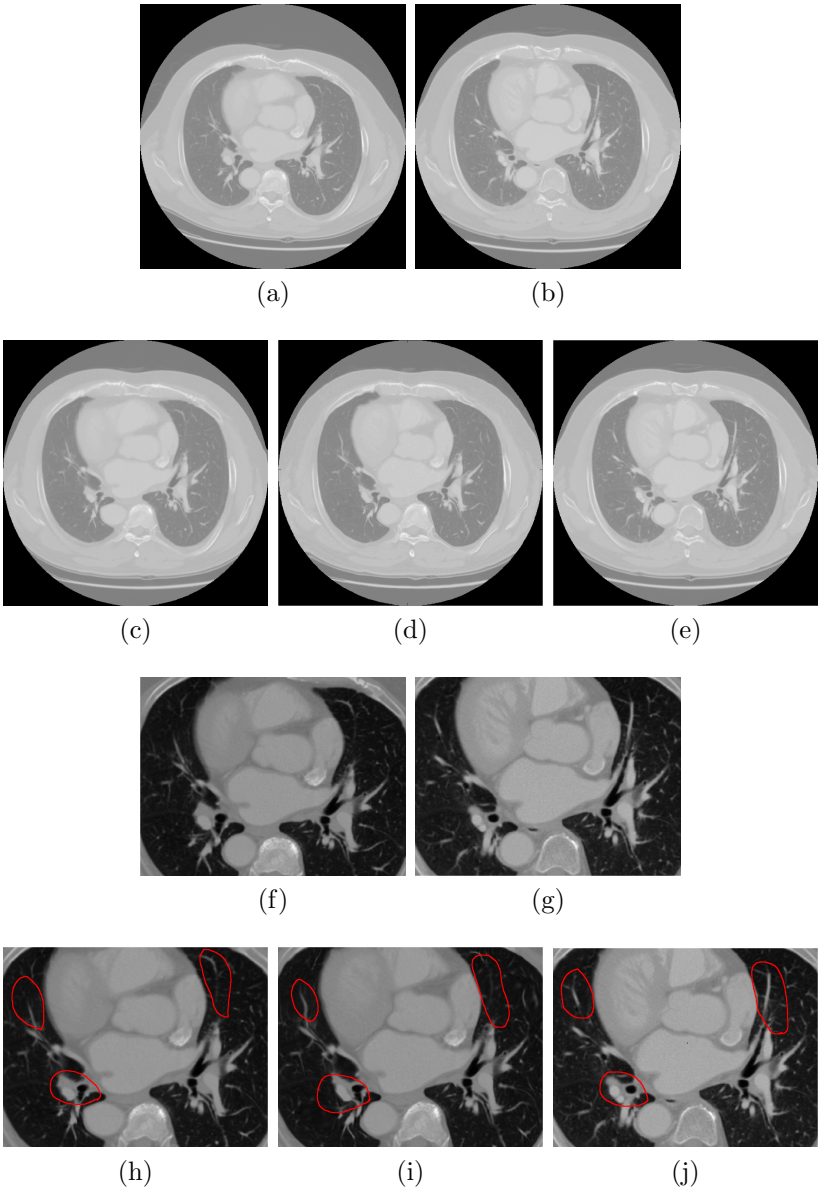


Figure 2.5: Registration result of the lung CT images that were captured with three months gap. (a) Source image; (b) Reference image; (c) MIRT [2] [RMSE: 7.71]; (d) GP-Registration [3] [RMSE: 7.38]; (e) 3D-SCoBeP [RMSE: 4.18]; (f) Source image (zoom in); (g) Reference image (zoom in); (h) MIRT [2] (zoom in); (i) GP-Registration [3] (zoom in); (j) 3D-SCoBeP (zoom in).

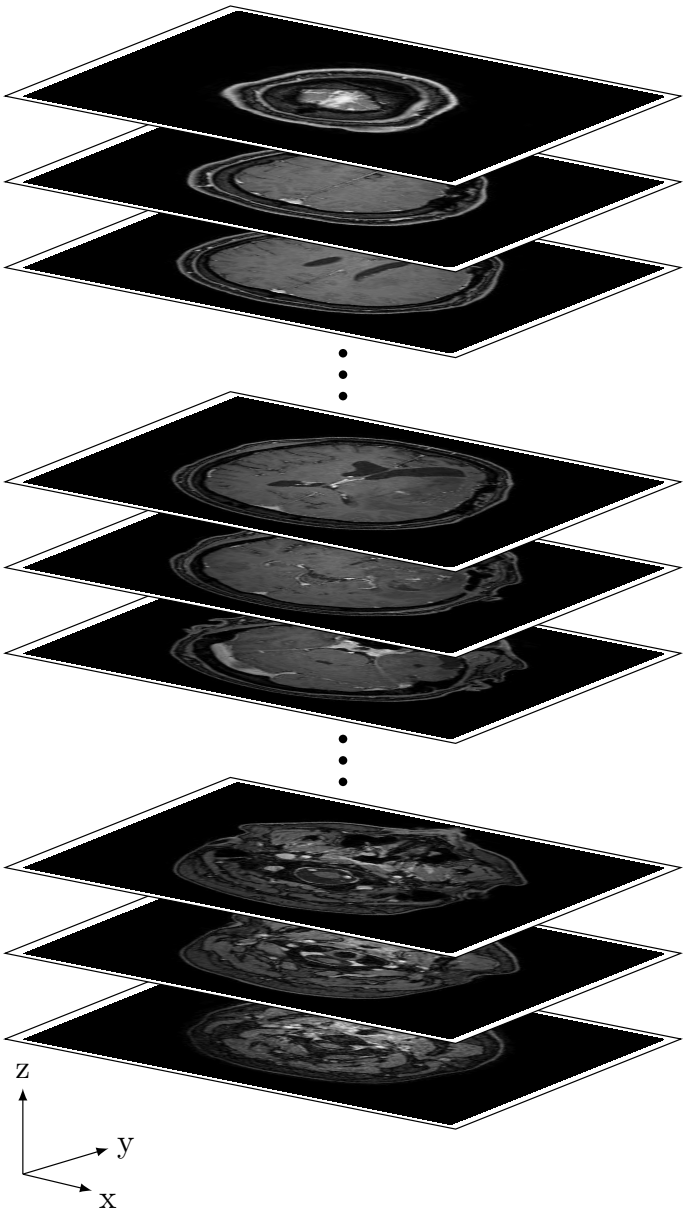


Figure 2.6: Brain MR images captured in parallel to X–Y (transverse) plane.

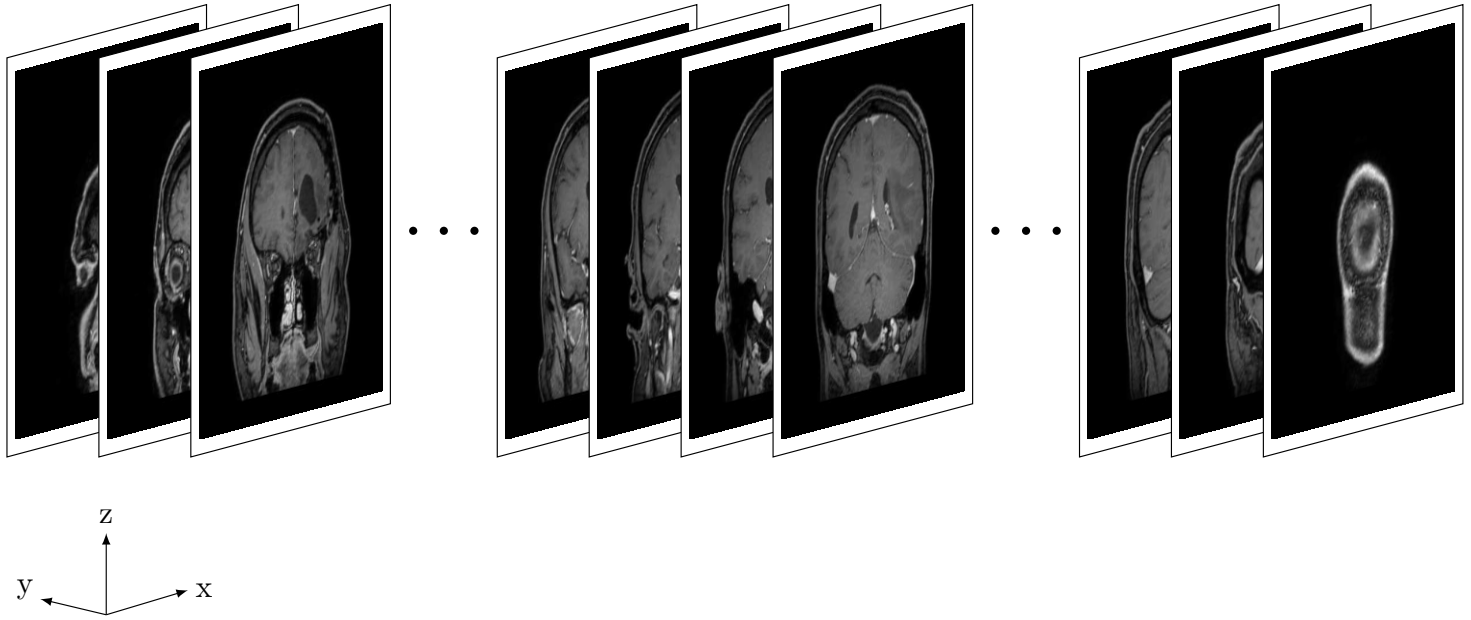
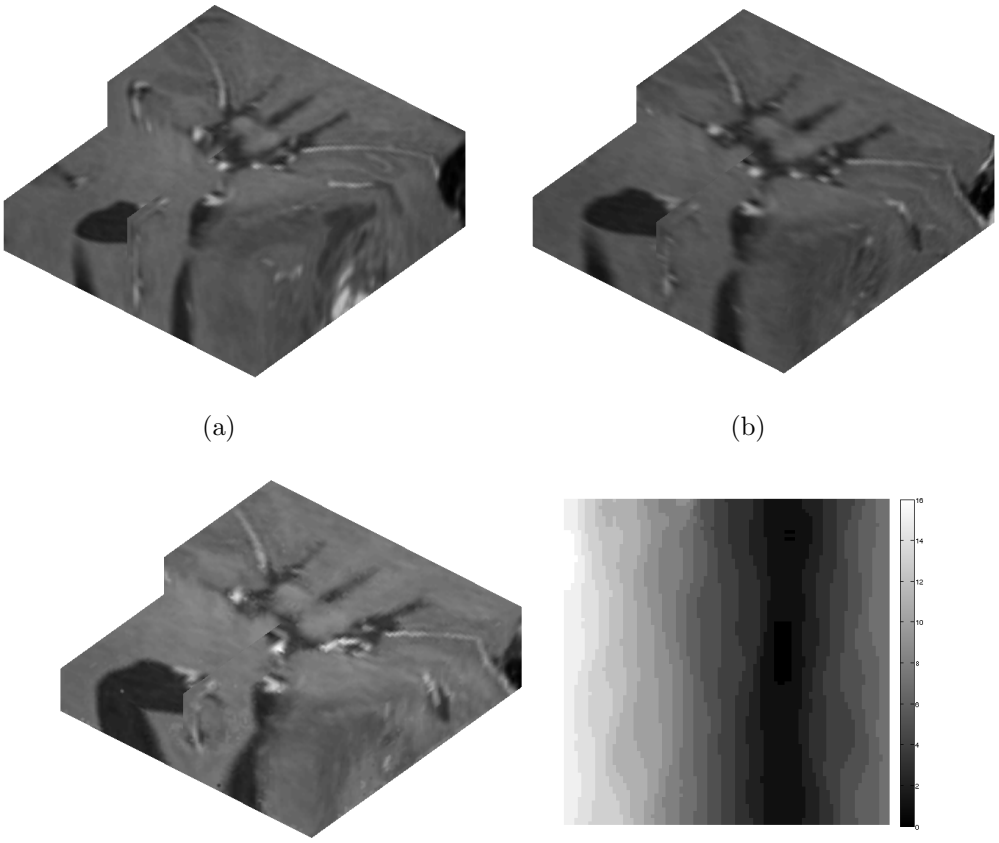


Figure 2.7: Brain MR images captured in parallel to X–Z (sagittal) plane.

Here, the captured data are slices of brain image, where each one has 320×320 voxels. In this case, I use 200 slices which were captured in parallel to the X–Y (sagittal) plane as the reference data and 180 slices in parallel to the X–Z (transverse) plane as the source data. For registering this kind of 3D data, I first map each reference and source to a template data in size of $320 \times 320 \times 320$ using B-Spline interpolation [65]. Since resampling could lead to an image to a new set of coordinates often provides a loss in image quality, the interpolating should be implemented with great care. Although there will be some expenses in computing time, the image quality can be improved by resampling using the B-Spline interpolation function.

Note that I set $a = b = c = 7$ and keep all the parameters as the same as previous section. Then, I apply the 3D-SCoBeP on the interpolated data. Fig. 2.8(a) and 2.8(b) show the reference and the source data, respectively. Fig. 2.8(c) shows the output of 3D-SCoBeP. In Fig. 2.8(d), I present the motion field in one of the slices in parallel to the X–Y (sagittal) plane which the darker points have less movement (minimum three pixels movement) and the lighter points are the area with more displacement (maximum sixteen pixels movement). In the other words, the selected area has three pixels translation in compared to the reference data. Also, the distance between the slices in Y direction is not uniform. These slices are closer to each other in the area that is darker in Fig. 2.8(d). Note that all selected voxels' probabilities are bigger than the threshold θ.

Fig. 2.9 shows the output of 3D-SCoBeP on brain data. While in Fig. 2.9(a), I applied 3D-SCoBeP directly on the original reference and source data with no interpolation approach, Fig. 2.9(b) shows the result of 3D-SCoBeP when I map each reference and source data to a template data using B-Spline interpolation. As seen in this figure, the registration with no interpolation performs poorly with significant misalignment where the interpolation approach brings the accurate performance in registration images.



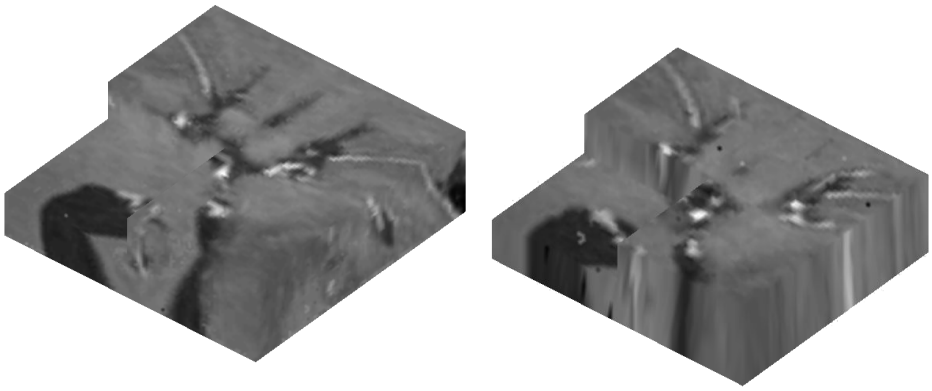
(a) (b)

(c) (d)

Figure 2.8: Result of 3D-SCoBeP on Brain CT images. (a) The reference CT image where the CT slices were taken in parallel to X–Z (sagittal) plane; (b) The source CT image where the CT slices were taken in parallel to X–Y (transverse) plane; (c) The 3D-SCoBeP result; (d) motion field in the X–Y (transverse) plane for one selected slice.

2.4 Discussion and Conclusion

In conclusion, I have proposed an effective registration method based on a sparse coding and belief propagation. The proposed method can be used for both rigid and non-rigid registration. My technique executes registration by first running sparse coding over an overcomplete dictionary out of 3D features of the reference image to gather possible match candidates. Belief propagation is then applied to eliminate bad candidates and to select optimum matches. The experimental result illustrates



(a) (b)

Figure 2.9: Result of 3D-SCoBeP on Brain CT images. (a) The 3D-SCoBeP result with B-Spline interpolation; (b) The 3D-SCoBeP result without B-Spline interpolation;

that my proposed algorithm compares with the high accuracy MIRT method by Myronenko and Song [2] and the state-of-the-art GP-Registration by Periaswamy and Farid [3] over the CT and MR images. The key advantage of my proposed method lies in the fact that it is applicable to both global and local registration [35]. Since the voxels act independently in my proposed method, by changing the size of the dictionary and using a part of data to make it, one can employ 3D-SCoBeP for medical image registration with higher speed locally. In addition, I used a 3D factor graph for the entire 3D data instead of using a 2D factor graph per slice. This new strategy along with using graph-cut method that refines the disparity of each voxel enable us to register 3D data captured parallel to the sagittal plane into 3D data captured parallel to the transverse plane.

CHAPTER 3

GENOME SEQUENCE ALIGNMENT USING EMPIRICAL TRANSITION PROBABILITY, SPARSE CODING AND BELIEF PROPAGATION

The latest sequencing technologies have generated numerous sequenced genomes for various species. This increasing volume of data requires tools that can accurately compare multiple genome sequences to aid in the study of populations, pan-genomes, and genome evolution [5, 6]. For a particular study, many individual genomes may be sequenced to investigate genetic diversity. For example, the Cancer Genome Atlas [10] and 1000 Genomes Project [11] will generate genome sequences from several thousand people. The complete bacterial genomes in public databases are already over one thousand. To better utilize this huge amount of sequenced genome information, many tools have been developed that are capable of efficiently finding similar sequences from whole genomes.

3.1 Introduction

In bioinformatics, sequence alignment is an important way to identify similar regions that might be associated with similar functional and structural relationship between sequences. With the quick growth of genomic data, it is important to develop effective sequence alignment techniques that are scalable. The past decade has witnessed the development of many sequence alignment technologies. Cancers are

caused by the collection of genomic sequence changes [66]. Therefore, alignment and analyses of cancer genome sequences provide basics to understand cancer biology, diagnosis and therapy.

In general, pairwise sequence alignment methods can be classified into local and global approaches. The global alignment attempts to find the best match between two strings with similar lengths through global optimization. In contrast, the local alignment is usually used to identify regions of similarity between a short query and a longer sequence. Global alignments [67–70] are less prone to demonstrating false homology as each letter of one sequence is constrained to being aligned to only one letter of the other. Local alignments [71–74], on the other hand, can cope with rearrangements between non-syntenic, orthologous sequences by identifying similar regions in sequences; this, however, comes at the expense of a higher false positive rate due to the inability of local aligners to take into account overall conservation maps [75].

A lot of efforts have been made to improve the efficiency and efficacy of sequence alignments. The ClustalW program proposed by Thompson and Larkin [76,77] uses a multi-stage mechanism to weight and to align sub-sequences based on sequence divergences. In addition, sequence annealing technique incrementally builds sequence alignment one at a time by checking whether a single match is consistent with a partial multiple alignments [78]. Darling *et al.* proposed a hidden Markov model that uses a sum-of-pairs breakpoint score to facilitate the detection of rearrangement breakpoints, when genomes have unequal gene content [79]. Mummer is a highly efficient suffix tree based matching tool for whole genome alignment, as well as incomplete genomes [80].

Researchers also proposed heuristics to accelerate sequence alignment. For example, the bounded sparse dynamic programming (BSDP) is used to support rapid approximation of exhaustive alignment in [81]. Another heuristic-driven approach,

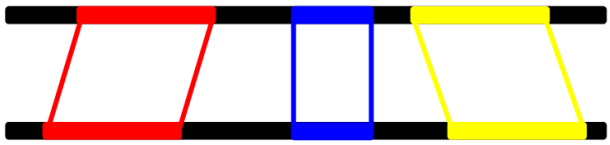
namely FastTree, is a tree-based method that stores profiles of internal nodes in a tree, such that candidate joins can be quickly identified. FastTree is also scalable for handling alignments over 10,000 sequences [82, 83].

Maximum-likelihood based approaches like PhyML and RAxML-VI-HPC have been developed as well. PhyML [84] used a hill-climbing algorithm that adjusts tree topology and branch length at each tree modification iteration. RAxML-VI-HPC [85], which stands for randomized accelerated maximum likelihood for high performance computing, takes advantages of a parallel program to support large-scale genome alignment.

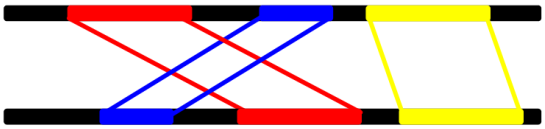
Whole genome sequence alignment are used for studying genome evolution and genetic diversity [86,87]. For example, Blanchette *et al.*, defined a Threaded Blockset Aligner (TBA) and built a threaded blockset under the assumption that all matching segments occur in the same order and orientation in a given sequence [88]. TBA was designed for aligning megabase-sized regions of multiple mammalian genomes. Darling *et al.* [89] implemented a method for identification and alignment of conserved genomic DNA in the presence of rearrangements and horizontal transfer called Mauve. Mauve has been applied to align nine enterobacterial genomes and to determine global rearrangement structure in three mammalian genomes. There are other whole-genome alignment tools that can align multiple whole genomes such as [90–92].

Whole-genome alignment tools are classified from collinear multiple sequence alignment tools, such as tools in [76, 93, 94] where they can align very long sequences and detect the presence of rearrangements, duplications, and large-scale sequence gains and losses. For example, Bradley *et al.*, in [93] proposed a program for the alignment of multiple biological sequences that is statistically motivated and fast for practical size problems. It was based on pair hidden Markov models which approximate an insertion/deletion process on a tree and used a sequence annealing algorithm

to combine the posterior probabilities estimated from these models into a multiple alignment. Edgar *et al.* proposed another alignment tool named MUSCLE which is a program for creating multiple alignments of protein sequences [94]. Elements of the algorithm include fast distance estimation using *k-mer* counting, progressive alignment using a log-expectation score, and refinement using tree-dependent restricted partitioning. In spite of collinear alignment technologies, non-collinear alignment such as [95, 96] contains the elements that are arranged in some non-linear order (see Fig. 3.1).



(a) Collinear



(b) Non-collinear

Figure 3.1: Collinear vs. Non–collinear nucleotide sequence alignment.

In this chapter, I propose a novel alignment method that uses sparse coding [97] and empirical transition probability to tackle the scalability challenge. Thanks to the sparse representation, my mechanism can handle long sequences with reduced memory footprint. I also leverage belief propagation to combine local and neighboring information of candidate nucleotides into consideration and generate matching scores to determine the best match. First, I index the reference and the read genome sequence using empirical transional probability and pick the top score indexes from the reference genome sequence to build an over-complete dictionary. I then find a set of candidate nucleotide for each nucleotide of the test sequence using sparse coding from the constructed dictionary. The match score of each candidate nucleotide will be evaluated taking both local and neighboring information into account us-

ing belief propagation. The rest of this chapter is structured as follows. Section 3.2 introduces my proposed method. Section 3.3 presents my results, including the comparison against SOAP aligner [6] and BWA [5]. Finally, I draw my conclusions in Section 3.4.

3.2 Proposed Method

In this section, I present my genome indexing and alignment framework in detail, where the proposed method includes three steps: indexing, index matching, and sequence matching. In this work, I refer to "*reference sequence*" as the base-line sequence and try to align a "*read sequence*" against the base-line sequence.

3.2.1 Indexing

The current genome indexing methods generate huge indices before performing the actual alignment to decrease the alignment time [98, 99]. The indexing process can be very time-consuming. In contrast, my proposed indexing technique provides a faster and light-weight alternative for index generation, which is similar to the big data retrieval systems that were proposed [100–102]. These indices can reduce the search space and provide an estimation of the read sequence locations in the reference sequence. The proposed genome indexing technique models a nucleotide sequence as a graph by counting the transitions between each pair of nucleotides. To be more specific, as shown in Fig. 3.2, I consider a graph with four states according to the different types of nucleotides and sixteen vertices according to all possible transitions between nucleotides. I read the first nucleotide of the sequence and treat it as the initial state. Then, I move from one state to the other state by scanning the next nucleotide repeatedly till the end of the sequence. Afterwards, I calculate the number of nucleotide transitions where I count how many times I pass one vertex

in the graph and store them in a 4×4 matrix. Finally, I normalize the resulting matrix as follows:

$$I = \begin{array}{c} \\ A \\ C \\ G \\ T \end{array} \begin{array}{c} \begin{array}{cccc} A & C & G & T \end{array} \\ \begin{pmatrix} k_{aa} & k_{ac} & k_{ag} & k_{at} \\ k_{ca} & k_{cc} & k_{cg} & k_{ct} \\ k_{ga} & k_{gc} & k_{gg} & k_{gt} \\ k_{ta} & k_{tc} & k_{tg} & k_{tt} \end{pmatrix} \end{array} \times \frac{1}{\sum_{s,w \in \{a,c,g,t\}} k_{sw}} \tag{3.1}$$

where k_{sw} is the number that has the S-type nucleotide immediately before the W-type nucleotide.

If the length of a sequence is larger than a given threshold i.e., h, I divide it into subsequences with maximum length of h, where each subsequence will have o nucleotides overlap with their neighbors. I set $o \geq \frac{h}{2}$ so that each pair of nucleotides can be counted at least twice. For each subsequence i, I count the transition of the nucleotides from the start of the subsequence till its end to reveal the number of different nucleotides that reside beside each other. In Fig. 3.3, an input sequence with $h = 250$ is used to demonstrate the proposed indexing process, where I_i is the calculated index for the input sequence based on the transition graph shown on the left hand side. Finally, I normalize the transition matrix, which will be used to find the approximate location of each subsequence in the next step.

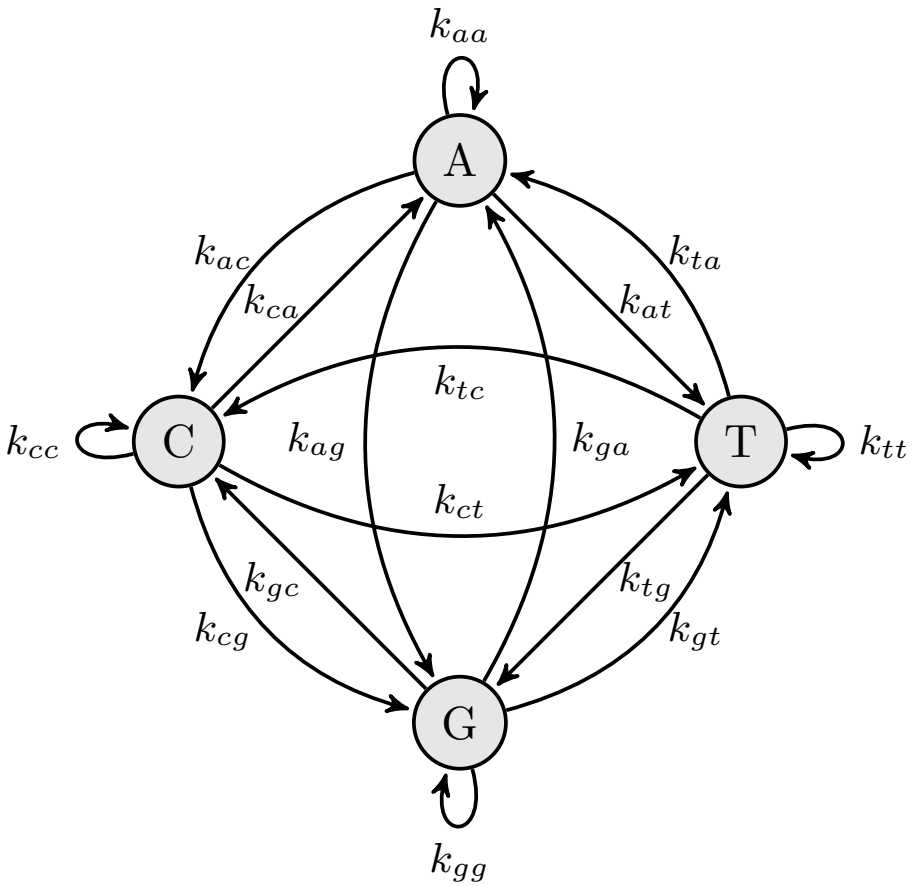
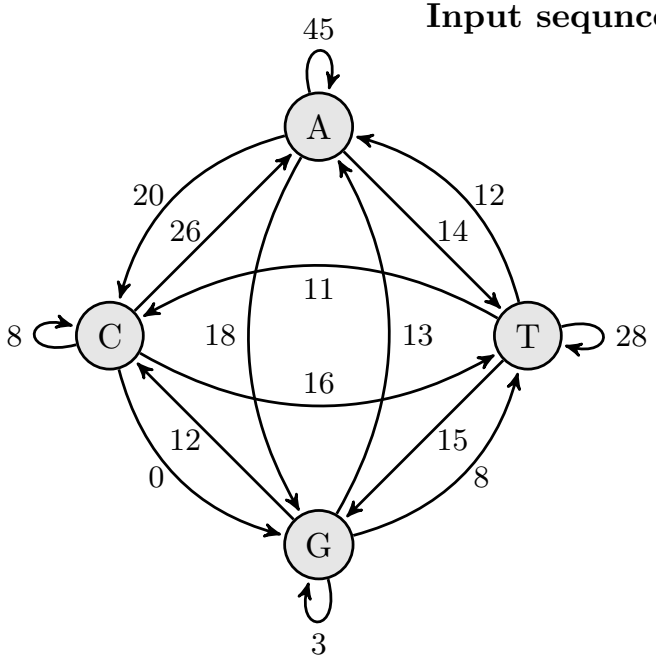


Figure 3.2: The transition diagram between nucleotides. k_{sw} is the number of appearance of the W-type nucleotide immediately after the S-type nucleotide where $s, w \in \{a, c, g, t\}$.



Input sequnce={aaaccaagaggcagaggttgcagtgagccaagatcatgccattgcactccagcc
ttagcaacagagtgagactccatctcagaacaacaacaacaacaaaaaaaaaaaa
acacaaaaaaacaaaaaattctgcaactaattaatttgttgtaactcttaaagcag
gaaccttatatagaaaatgttgtgatcctattaatttttttttctttctatgtaagca
acttcacttttgactttgcagcactgac}

$$I_i = \begin{pmatrix} 45 & 20 & 18 & 14 \\ 26 & 8 & 0 & 16 \\ 13 & 12 & 3 & 8 \\ 12 & 11 & 15 & 28 \end{pmatrix} \times \frac{1}{249}$$

$$= \begin{pmatrix} 0.1807 & 0.0803 & 0.0723 & 0.0562 \\ 0.1044 & 0.0321 & 0.0 & 0.0643 \\ 0.0522 & 0.0482 & 0.0120 & 0.0321 \\ 0.0482 & 0.0442 & 0.0602 & 0.1124 \end{pmatrix}$$

Figure 3.3: An example of the indexing procedure for a small sample subsequence.

3.2.2 Index Matching

The index matching step is designed to find similar indices based on global information of the sequence. I define a symmetric distance function between two index matrices I and J as follows: $D_{MSE}(I, J) = \|I - J\|_f$, where $\|\cdot\|_f$ is the Frobenius norm of the matrix.

After generating the indices of the reference sequence and the read sequence, the D_{MSE} distances to all reference sequence indices are calculated, where the top t most similar indices in terms of D_{MSE} are chosen as candidate indices. To find the best matched index, I resort to belief propagation (BP) on a factor graph. In this dissemination, I provided a concise review about the BP algorithm on factor graph on Section 1.1. Interested readers can check my earlier publications in [4, 22, 57] for more details about the factor graph design and the BP algorithm.

For each index in the read sequence, I consider a variable node and a factor node which connects each two neighbor variable nodes. An extra factor node of each variable node stores the prior probabilities which are calculated based on D_{MSE} distance (see Fig. 3.4).

I apply BP to the factor graph of the test sequence with n candidate nucleotides as the prior knowledge. BP updates the probability of candidate nucleotides based on the probabilities of their neighbors.

Then, the candidate index numbers are fed to a factor graph and the corresponding D_{MSE} of each of candidates is employed to calculate the initial probability (prior probability) of each candidate. Then, message passing (i.e., forward and backward) algorithm is applied to calculate the best match indices. The correspond subsequences of these indices is used in the next step.

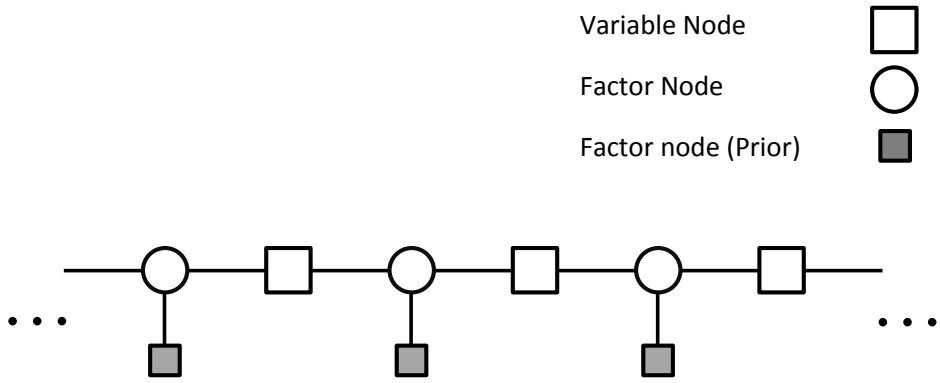


Figure 3.4: Nucleotides model: One dimensional factor graph used in Belief Propagation.

3.2.3 Sequence Matching

The sequence matching step is based on sparse coding and BP algorithm. In this step, I use the subsequences that were selected in the previous step to generate an over-complete dictionary. Then, for each nucleotide in the read sequence, I pick n candidate nucleotides using sparse coding. To choose the best candidate, a factor graph is employed where for each nucleotide in the read sequence, a variable node was assigned. The relation between neighborhood variable nodes is taking care of by a factor node. Also one extra factor node for each variable node keeps the prior probability of candidates. By applying belief propagation to a factor graph, I can obtain the best match for each nucleotide in the read sequence. A detailed description about the sequence matching can be found in my recent publications [4, 56]. A summary of the main procedure for my proposed indexing method is shown in Algorithm 2.

Algorithm 2 Proposed nucleotide sequence alignment algorithm for estimating the location of the input sequence

Inputs: a reference sequence $\mathcal{X} \in \mathbb{R}^M$, a test sequence $\mathcal{Y} \in \mathbb{R}^N$, number of the candidate state matrix k, number of the candidate points n

Initialize: a 4×4 state matrix I storing the numbers of nucleotide states (3.1), nucleotide overlap v

Iterate: while the length of the sequence $\mathcal{X}$ is not reached

Fill the reference state matrix I: For each subsequence $x_i \in \mathcal{X}$ with v nucleotide overlap in each direction perform:

- $I_i = MakeIndex(x_i)$

Fill the test state matrix J: For each subsequence $y_j \in \mathcal{Y}$ with v nucleotide overlap in each direction perform:

- $J_j = MakeIndex(y_j)$
- $[c_j, \rho_j] = FindCandidates(J_j, I, k)$

Refine the candidate state matrix:

- $\hat{\rho} = BP(c, \rho)$

Find the correspond nucleotide in the reference sequence $\mathcal{X}$: For each subsequence $y_j \in \mathcal{Y}$ with v nucleotide overlap in each direction perform:

- $z_j = FindBestSubsequence(\mathcal{X}, y_j, \theta, n)$ (see Algorithm 3)

Output: the estimated version of aligned sequence $\mathcal{Z}$

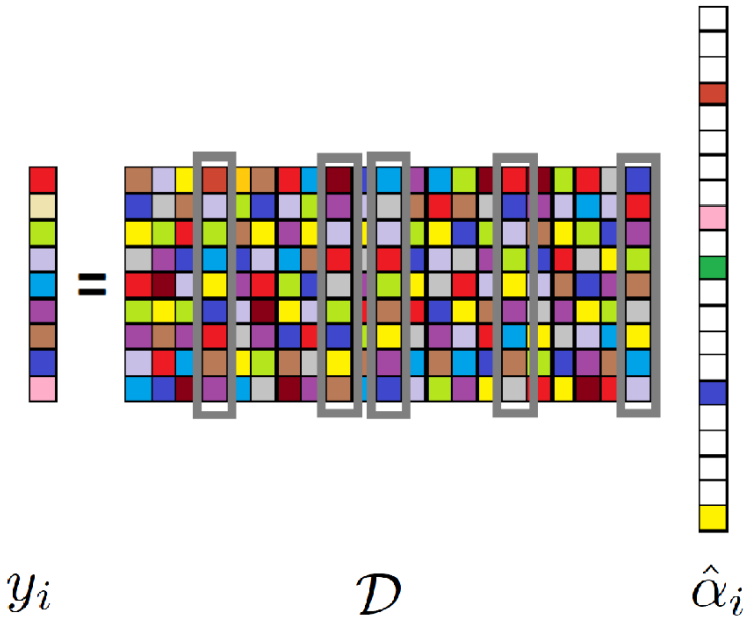


Figure 3.5: Sparse representation of a feature vector y_i with a dictionary $\mathcal{D}$: $\hat{\alpha}_i$ as a sparse vector constructs the feature vector y_i using a few columns (highlighted in gray) of dictionary $\mathcal{D}$.

This step is inspired by my recent work, SCoBeP [56]. First, I map the reference nucleotide sequence $\mathcal{X}$ of size N and the test nucleotide sequence $\mathcal{Y}$ of size M into the two integer sequences, $\mathbf{X}$ and $\mathbf{Y}$, respectively. I then extract the features from the reference sequence $\mathbf{X} \in \mathbb{Z}^{N\times 2}$ and the test sequence $\mathbf{Y} \in \mathbb{Z}^{M\times 2}$.

Second, I create a dictionary $\mathcal{D}$ containing all the extracted feature vectors of the reference sequence $\mathbf{X}$ which match the corresponding extracted features of the test sequence $\mathbf{Y}$. The dictionary includes all vectorized one dimensional features as its columns where all of them have been normalized. I then apply sparse coding to each extracted feature of the test sequence. Sparse coding will reconstruct a nucleotide vector at each nucleotide g_i as a linear combination of the reference sequences. Note that the obtained representation coefficients α_i should be sparse, i.e., it should be 0 for most coefficients. The non-zero coefficients of α_i indicate the corresponding nucleotides on the reference sequence (see Fig. 3.5).

To select n candidate nucleotides, I simply pick those corresponding to n which

have the largest coefficients in the sparse coefficient vector. I store the locations of these candidate nucleotides in a length-n vector $\mathcal{L}_i$ and a probability vector ρ_i as a length-n vector which stores the corresponding probabilities for the sparse coefficient vector. Each coefficient in the probability vector ρ_i serves as a prior probability of matching the nucleotide at i to a nucleotide of a reference sequence. This probability vector takes only local characteristics into account but ignores neighborhood characteristics of the matches.

Finally, I expect that nearby nucleotides in the test sequence should also match the nucleotides that are close to each other in the reference sequence. To incorporate these neighborhood characteristics, I model the problem by a factor graph and apply the Belief Propagation (BP) algorithm to identify the best matches. I consider a one dimensional factor graph as follows: for each nucleotide in the test sequence, one variable node was assigned and each variable node was connected to its two neighbors by a factor node. Also, I consider one extra factor node for each variable node to impose the restriction of prior probabilities for the candidate nucleotides (see Fig. 3.4).

To align the test nucleotide sequence, I select the nucleotide candidate with highest probability and then calculate the displacement β between the current nucleotide and the selected candidate nucleotide. Therefore, for each nucleotide in the test sequence, I have $\mathcal{Z}_i$, β_i and ρ_i for each nucleotide g_i which are the most probable match nucleotide, the most probable displacement and the probability of the current match, respectively. A summary of the alignment method is shown in Algorithm 3.

Algorithm 3 FindBestSubsequence($\mathcal{X}, y, \theta, n$)

Inputs: a reference sequence $\mathcal{X} \in \mathbb{R}^M$, a test sequence $\mathcal{Y} \in \mathbb{R}^N$, a threshold θ, number of the candidate points n

Convert a string sequence to numeric sequence:

- $\mathbf{X} = ConvertData(\mathcal{X})$
- $\mathbf{Y} = ConvertData(\mathcal{Y})$

Extract feature and construct dictionary:

- $\hat{Y} = ExtractFeature(\mathbf{Y})$
- $\hat{X} = ExtractFeature(\mathbf{X})$
- $\mathcal{D} = MakeDic(\hat{X})$

Find the initial estimate of the match location: For each vector $y_i \in \hat{Y}$ perform:

- $\alpha_i = FindSparseVector(\mathcal{D}, y_i)$
- $[\mathcal{L}_i, \hat{\rho}_i] = FindTopScoreMatch(n, \alpha_i)$

Refine the candidate match location:

- $\rho = BP(\mathcal{L}, \hat{\rho})$

Find the correspond nucleotides:

- $[\mathcal{Z}, \beta] = Warp(\hat{X}, \rho, \mathcal{L})$

Output: the estimated version of aligned sequence $\mathcal{Z}$

3.2.4 Implementation Details

- $I_i = MakeIndex(x_i)$ fills the state matrix I_i using the relationship of nucleotides in the subsequence x_i. The subsequence x_i is scanned through all its nucleotides and the corresponding counts will be stored into the state matrix I_i. For example, k_{cg} in I_i in (3.2) shows how many times the nucleotide C will be identified, which is next to the nucleotide G in the subsequence x_i. Note that each subsequence x_i has a separate state matrix I_i, where i is the subsequence index.

$$I_i = \begin{array}{c} \\ A \\ C \\ G \\ T \end{array} \begin{array}{c} \begin{array}{cccc} A & C & G & T \end{array} \\ \begin{pmatrix} k_{aa} & k_{ac} & k_{ag} & k_{at} \\ k_{ca} & k_{cc} & k_{cg} & k_{ct} \\ k_{ga} & k_{gc} & k_{gg} & k_{gt} \\ k_{ta} & k_{tc} & k_{tg} & k_{tt} \end{pmatrix} \end{array} \times \frac{1}{\sum_{s,w \in \{a,c,g,t\}} k_{sw}} \tag{3.2}$$

- $[c_j, \rho_j] = FindCandidates(J_j, I, k)$ identifies k candidate state matrices that are highly similar to the test state matrix J_j in I and stores their indices in vector c_j and their probabilities in vector ρ_j. Note that the approach will compute the Mean Square Error (MSE) of the test state matrix J_j with each possible I_i of the reference state matrices and select I_i that has the smallest MSEs.

- $\hat{\rho} = BP(c, \rho)$ models the problem by a factor graph and applies belief propagation [12] to update probability ρ. The updated probability $\hat{\rho}$ can be used to align the reference state matrix index onto the test state matrix index. In my case, I assign a variable node for each test state matrix index and connect each pair of neighboring state matrix indices with a factor node. Also, I introduce one extra factor node to take care of the prior knowledge obtained in the MSE

step for each test state matrix index (for more details, see [56]).

- $\mathbf{X} = ConvertData(\mathcal{X})$ maps the input string of the nucleotide sequence to a sequence of integer values for the further processing. The mapping function of four nucleotides $\{A, C, G, T\}$ can be defined in a two-dimension space where $A = (1, 1)$, $C = (1, -1)$, $G = (-1, 1)$, and $T = (-1, -1)$, respectively. Therefore, $\{A, C, G, T\} \mapsto \{(1, 1), (1, -1), (-1, 1), (-1, -1)\}$.

- $\hat{Y} = ExtractFeature(\mathbf{Y})$ presents a vector extractor algorithm using $\mathbf{Y}$ as a source sequence, where the result is a two dimensional matrix containing the vectorized one dimensional sequences. To this end, I consider a vector of size $S = 2 \times (2a + 1)$ containing neighboring nucleotides on two sides of a nucleotide, where a is a positive integer and the first "2" is corresponding to the dimension of the mapping space). For each nucleotide g_i in the test sequence $\mathbf{Y}$, I vectorized a sequence centered around the nucleotide g_i to a feature vector $y_i \in \mathbb{Z}^{S\times 1}$. A two dimensional test feature data $\hat{Y} \in \mathbb{Z}^{N\times S}$ is then constructed from y_i as follows:

$$\hat{Y} = \{y_i \mid 1 \leq i \leq M\}. \tag{3.3}$$

  Note that $\hat{X}$ is created in the same manner as $\hat{Y}$ but from the reference sequence $\mathbf{X}$ instead.

- $\mathcal{D} = MakeDic(\hat{X})$ creates a dictionary $\mathcal{D}$ using the vectors of $\hat{X}$. Later, the dictionary $\mathcal{D}$ is used to match the extracted features of the source sequence to corresponding the extracted features of the reference sequence. , a dictionary which contains feature vectors of $\mathbf{X}$ is constructed. Thus, I can write $\mathcal{D} = [x_1\ x_2 ...\ x_N]$ where x_i is a feature vector of $\hat{X}$. Note that I normalize dictionary $\mathcal{D}$ to guarantee the norm of each feature vector to be 1.

- $\alpha_i = FindSparseVector(\mathcal{D}, y_i)$ finds the candidate match nucleotide using the sparse coding algorithm, where α_i is a sparse vector. Mathematically, I try to solve the following sparse coding problem to find the most sparse coefficient vector α_i (see Fig. 3.5) such that

$$y_i = \mathcal{D}\hat{\alpha}_i. \tag{3.4}$$

  Although there are several methods to solve (3.4) [17–19], in my work, I employ Subspace Pursuit (SP) [19] because of its computational efficiency.

- $[\mathcal{L}_i, \hat{\rho}_i] = FindTopScoreMatch(n, \alpha_i)$ picks up the n largest coefficients of α_i as n candidates. $\mathcal{L}_i$ is a $n \times 1$ vector that stores the locations of these candidate nucleotide and $\hat{\rho}_i$ is the length-n vector that stores the corresponding probabilities of $\mathcal{L}_i$. Each coefficient in $\hat{\rho}_i$ serves as a prior probability of matching the source sequence at i to a sub-sequence centered around the nucleotide g_i. After finding the candidates and their initial probabilities, I concatenate the result of each nucleotide and construct following matrices:

$$\mathcal{L} = [\mathcal{L}_1\ \mathcal{L}_2 ...\ \mathcal{L}_N], \hat{\rho} = [\hat{\rho}_1\ \hat{\rho}_2 ...\ \hat{\rho}_N], \tag{3.5}$$

  which I will use to apply belief propagation at the next step.

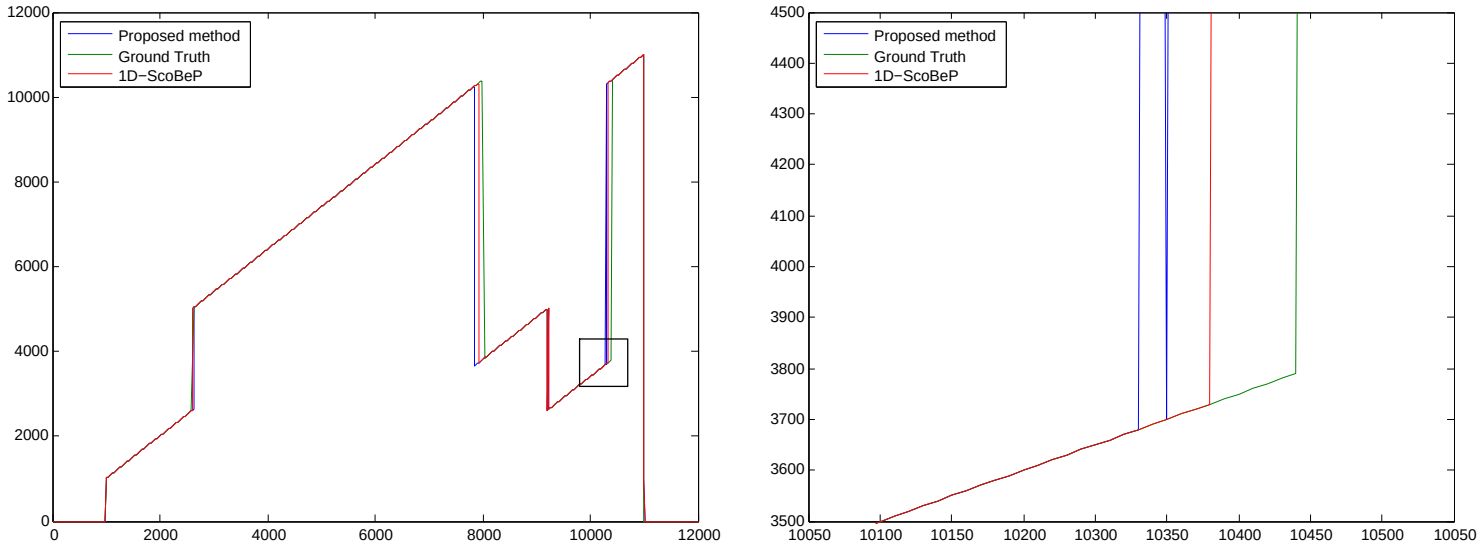
- $\rho = BP(\mathcal{L}, \hat{\rho})$ models the problem by a factor graph (see Fig. 3.6) and applies belief propagation [12] to update probability ρ. The updated probability ρ can be used to align the reference sequence onto the test sequence. In my case, I assign a variable node for each nucleotide on the source sequence and connect each pair of neighboring nucleotide with a factor node. Also, I introduce one extra factor node to take care of the prior knowledge obtained in the sparse coding step for each nucleotide of the source sequence (for more details,

see [56]).

- $[\mathcal{Z}, \beta] = Warp(\mathbf{X}, \rho, \mathcal{L}, \theta)$ returns the aligned sequence $\mathcal{Z}$ and a displacement vector β which contains the movement of each nucleotide in the reference sequence. Note that in step $BP(\cdot)$, I calculated the refined probability of each candidate nucleotide match.

- $z_j = FindBestSubsequence(\mathcal{X}, y_j, \theta, n)$ finds the corresponding location for a nucleotide $y_j \in \mathcal{Y}$. In this step, the reference nucleotide sequence $\mathcal{X}$ and the test nucleotide sequence $\mathcal{Y}$ are converted into two integer sequences. Then, an over-complete dictionary is built with all subsequences in the $\mathcal{X}$. I then apply sparse coding followed by using Belief Propagation (BP) to identify the best matches. (see [4, 56] for more details) Note that I used non-overlapped subsequences to build the dictionary. This change decreases the memory usage and the accuracy of the proposed algorithm in compare to 1D-SCoBeP [4], but it increases the speed of my alignment algorithm.

3.3 Experimental Results



(a) (b)

Figure 3.6: The results of proposed method for non-collinear nucleotide sequence alignment. a) comparison among alignment results of the ground truth, 1D-SCoBeP [4] and the proposed method. b) zoomed of the black square in figure 3.6–a to show the gap between the proposed method, ground truth and 1D-SCoBeP [4] on the jump point. The x–axis and y–axis are the index numbers of the original genome sequences and the shuffled genome sequences, respectively.

I designed my experiments based on work in [79] to evaluate the proposed method for aligning the nucleotide sequences and to compare it with SOAP aligner [6], BWA [5] and 1D-SCoBeP [4]. I considered the problem of aligning a sequence of human nucleotides from the National Center for Biotechnology Information (NCBI) [7] and Cancer Genomics Hub (CGHub) [8].

To evaluate the performance of my approach, I conducted two sets of tests on the nucleotide sequences. In the first set, I selected fifty short sub-sequences of human genomes and then used SOAP aligner, BWA, 1D-SCoBeP and the proposed method to find the location of selected sub-sequence nucleotide in the human chromosome. All of four algorithms successfully passed this test. I created twenty shuffled sub-sequences of the reference sequence as follows: for each read sequence R, I cut it into five pieces p_1, p_2, p_3, p_4 and p_5. Then I switched p_2 with p_4. Therefore, I converted a read sequence $R = [p_1, p_2, p_3, p_4, p_5]$ into a new read sequence $\hat{R} = [p_1, p_4, p_3, p_2, p_5]$.

Fig. 3.6 shows the result of the 1D-SCoBeP and the proposed method show a better performance with a gap of 100 to 120 nucleotides away from the ground truth. Since I was using non-overlapped subsequences for the dictionary generation, the gap between the proposed method and the ground truth was larger than these reported in 1D-SCoBeP [4]. In my experiments, the following parameters were used: the number of candidate points n is set to be 3, the sparsity factor $k = 3$ and the dictionary column size $a = 200$.

To evaluate the robustness of the proposed method, I generate indices for long human genome sequences (i.e. 5×10^8 nucleotides) where $h = 10000$ and $o = 5000$. Moreover, I synthesized insertion, deletion and mutation (i.e., indel) in these sequences. For indel rate, I picked 10^5 number of subsequences with size of 10^4 nucleotides. Then, I randomly modified a certain number of nucleotides (based on the indel rate) and aligned them with the references. I counted how many times the alignment location and real subsequence location (i.e., ground truth) are matched,

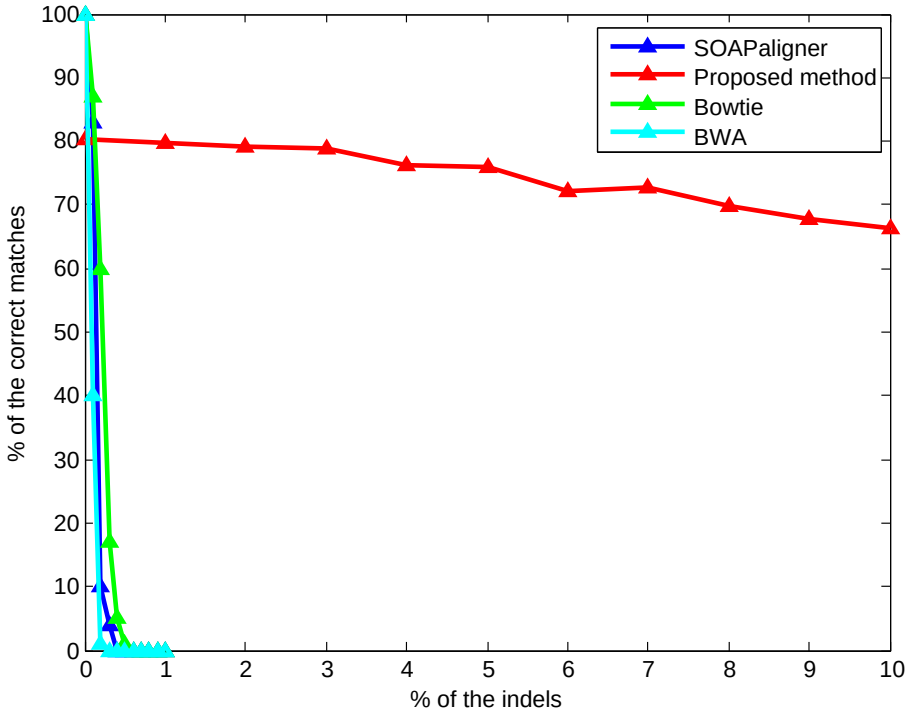


Figure 3.7: Accuracy of BWA [5], SOAP aligner [6] and the proposed method in present of different Indel rate, where the testing genome sequences were obtained from [7].

where the accuracy is defined as the count of the successfully aligned sequences over total number of the subsequences. Fig. 3.7 shows the accuracy of alignment of the proposed method, BWA and SOAP aligner in the presence of the different indel rates. The proposed method showed similar accuracies even when I increased the indel rate to 3%. Moreover, the proposed algorithm still showed more than 75% accuracy even after I modified 5% of the nucleotides in my selected subsequences. In contrast, the accuracy of the BWA and SOAP aligner decreased sharply as the indel rates increase.

I investigate the impact of small indel rate in the range from 0.5% to 1.5% in Fig. 3.8. In this figure, I showed accuracy of 1% indels in red for the data set used in 3.7

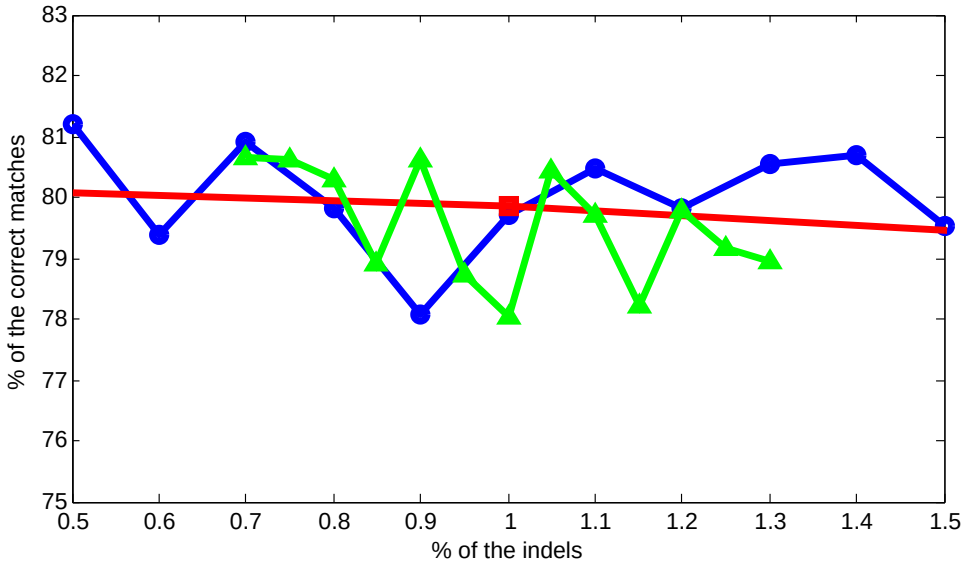


Figure 3.8: The percentage of successful alignments in present of 0.5% to 1.5% indels. The **green line** is the percentage successful alignments where the rate of the indels are changing with step equal 0.05% between 0.7% and 1.3%. The **blue line** is the percentage successful alignments where the rate of the indels are changing with step equal 0.1% between 0.5% and 1.5% and the **red line** is the same as the Fig. 3.7. Each point represents 10^5 random site selection with same indels rate. Note that the genome sequences used in this studies were obtained from [8] and [7].

as reference. To verify my result, I repeat the experiments with different indel steps and different read locations and present the results in green and blue, respectively. Note that each point in this figure was obtained from the evaluation over 10^5 read sequences. There are slight variation among the curves due to statistical deviation. The summary of the indel rate accuracy was shown in Table 3.1.

The computational complexity of proposed is mainly determined by the following three steps: 1) indexing 2) index matching 3) extracting sub-sequence nucleotides as features and constructing the dictionary, 4) finding candidate nucleotides via sparse coding, and 5) applying BP. Assume the size of the read and reference sequences are N and M nucleotides, respectively. The required time for create indexes is $O\big(M+N\big)$, because I have to scan whole read and reference sequences. The number of reference sequence indexes is $I_M = \frac{M}{h} + \frac{2oM}{h^2} = O\big(\frac{M}{h}\big)$ and similarly, $I_N = O\big(\frac{N}{h}\big)$.

Therefore, the time for the index match matching is $O\big(\frac{M}{h}\big) \times O\big(\frac{N}{h}\big) = O\big(\frac{MN}{h^2}\big)$. After index matching step, the size of search space reduces from M to $\bar{M} = I_s \times h$ where I_s is the number selected indexes I_s and h is the size of each index. The required time of feature extraction will be $O\big(a(\bar{M} + N)\big)$, where a is the size of the vector of extracted features for each nucleotide. The dictionary construction step involves the normalization of each column, which requires $O(a\bar{M})$ amount of time. Thus the total time complexity of the first step is $O\big(a(\bar{M} + N)\big)$. In the next step, the time complexity of Subspace Pursuit (SP) is $O\big(\log(f)a\bar{M}\big)$ [103], where f is the number of iterations for searching the sparse vector. Since I have to repeat the process to find candidate points for all N feature vectors, the time complexity of finding candidate points by SP is $O\big(\log(f)a\bar{M}N\big)$. Then, the time complexity of Belief Propagation in my factor graph is $O(vn^2\bar{M})$, where v is the number iterations before converging and n is the number of candidates in each variable node. Finally, the time complexity of proposed method will be $O\Big(MN + \log(f)a\bar{M}N + vn^2\bar{M}\Big)$.

3.4 Conclusion

In this chapter, I proposed a sparse coding and BP based method for indexing and alignment genome sequences. The proposed method builds a transition matrix based on the neighboring nucleotides of an input sequence and then reduces the search space by selecting the top K most similar subsequences based on their distances. The proposed algorithm selects candidate nucleotides by using sparse coding with an over-completed dictionary, which was constructed from the nucleotides of reference sequence in the indexing step. BP algorithm is then applied to select the best matches. Through experimental results, I showed that the proposed algorithm are comparable to SOAP aligner [6], BWA [5] and 1D-SCoBeP [4] in terms of the alignment accuracy. In addition, the proposed method is robust to insertions, deletions, and mutations in the genome sequences when comparing with SOAP aligner

and BWA. Finally, the proposed method is able to process much longer sequences then our previous 1D-SCoBeP approach.

Table 3.1: Percentage of successfull alignments

% of the Indels	Accuracy of **red** line	Accuracy of **blue** line	Accuracy of **green** line
0.00	80.33	–	–
0.50	–	81.19	–
0.60	–	79.38	–
0.70	–	80.90	80.64
0.75	–	–	80.63
0.80	–	79.82	80.29
0.85	–	–	78.90
0.90	–	78.09	80.63
0.95	–	–	78.74
1.00	79.85	79.71	78.04
1.05	–	–	80.43
1.10	–	80.49	79.70
1.15	–	–	78.22
1.20	–	79.81	79.78
1.25	–	–	79.16
1.30	–	80.54	78.94
1.40	–	80.70	–
1.50	–	79.52	–
2.00	79.09	–	–
3.00	78.90	–	–
4.00	76.33	–	–
5.00	75.90	–	–
6.00	72.09	–	–
7.00	72.86	–	–
8.00	69.79	–	–
9.00	67.87	–	–
10.00	66.42	–	–

CHAPTER 4

GENOME SEQUENCE PRIVACY PROTECTION USING COMPRESSED SENSING

In this chapter, I present a privacy preserving genomic data dissemination algorithm based on the compressed sensing. I participated in the challenge at the iDASH on March 24, 2014 in La Jolla, California which the result of the challage are available on $http://www.humangenomeprivacy.org$. In my proposed method, I am adding the noise into the sparse representation of the input vector to make it differentially private. It means I find the sparse representation using using the SubSpace Pursuit and then disturb it with sufficient Laplasian noise. I compare my method with state-of-the-art compressed sensing privacy protection method [1].

4.1 Introduction

Most genomic datasets are not publicly accessible, due to privacy concerns. Patients' genomic data contains identifiable markers and can be used to determine the presence of an individual in a dataset. Prior research shows that the re-identification can be possible when: There is a very small set of SNPs, Or The genomic data is aggregated; like releasing the frequencies of different SNPs across a population. To protect patients, the data owners impose an application and evaluation procedure. Then an agreement (like IRB approval) needs to be signed before the use of data is permitted. This process often takes months to complete and limits the researchers.

One solution to the problem can be to let each data owner publish a set of pilot data to help data users choose the right datasets based on their needs. Such pilot data comes from adding noise to the original genomic data to ensure that individuals information is protected. The data owners release these pilot data with the noise parameters and the mechanism that they used. A data user can download, run any kind of association tests and compare the outcomes with the other datasets outputs to get an idea which datasets can be useful. With such information, the researchers can approach the owners of the most relevant datasets for further research with proper agreements.

I reviewed the basics of compressed sensing in section 1.3 that is the inspiration of current research. Now, I explain my proposed genome privacy protection mechanism and conclude by demonstrating the results of proposed method on human genome sequences.

4.2 Privacy Protection Based On Compressed Sensing

The proposed privacy protection method is based on the compressed sensing mechanism. The input sequences to the proposed method are the genomic nucleotide (A, C, G, T). First, I process the input genomic sequences to find the location of the Single Nucleotide Polymorphisms (SNPs). I check the same location of the all genome sequences and look for the locations that their nucleotides are different which I mark them as SNPs. Then I keep only the SNPs location as representatives of the original genome sequences and calculate the frequency of changes in the same SNP. The nucleotide which appears less in the SNP location called "minor" and the nucleotide with most appearance in the SNP location called "major". These majors and minors with their frequencies will be used in the proposed privacy protection method.

To have a differentially private data, I am using the compressed sensing technique

and adding the noise to the genomic data representation in the transformation domain. As it was explained in the section 1.3, I need to specify an input vector y, a random matrix Φ and a transformation matrix Ψ to be used in the compressed sensing process. The input vector y is the frequency of the minors SNPs which I calculate it from pre-processing step. The random matrix ϕ is a binary independent and identically distributed (i.i.d.) of the Bernoulli distribution matrix which in average, half of the coefficient in each row and each column are "1"s and the rest are equal to "0"s. The transformation that I choose is the Haar wavelet transform (HWT) [104, 105] Because I know from [1, 106, 107] that if I have an ϵ budget how to use it on the wavelet domain that keeps the differential privacy.

In my proposed method, I am adding the noise into the sparse representation of the input vector to make it differentially private. It means I calculate the matrix $A = \Phi\Psi$ and then using the SubSpace Pursuit [19], I reach to the sparse representation of the y which I will call it $\hat{x}$. To making sure that the amount of the noise that I am adding to sparse representation is enough to protect the differential privacy, I add Laplasian noise λ to all element of the sparse representation according to [107]. Note that the element in the sparse representation are correspond to HWT tree (see Figure 4.1), therefore based on [106], I double the amount of noise when I move up one step in the wavelet tree from the leaves to the root. The result of this result of the adding Laplasian noise to $\hat{x}$ generates x^* which I use it to generate the differential private data to publish. To get the punishable data y^*, I need to apply inverse transformation on x^* as follows:

$$y^* = \Psi x^* \tag{4.1}$$

where the Ψ is the Haar wavelet transformation matrix. Algorithm 4. shows the summary of the proposed method.

Algorithm 4 : Proposed differentially private Protection using Compressed Sensing algorithm for Genomic Data

Inputs: a set genome sequences $\mathcal{X}$
Pre-processing: Converting the genome sequence to SNP sequences
Initialization: Set the initial parameters:

- Find the frequencies of major and minor for each SNP location
- Generate sampling matrix Φ from a Bernoulli i.i.d. distribution
- Generate Haar wavelet transformation matrix Ψ
- Calculate matrix $A = \Phi\Psi$
- Consider an array contains the frequencies of minors as input vector $y[c_j, \rho_j] = FindCandidates(J_j, I, k)$

Adding Noise: Add Laplasian noise:

- Finding $\hat{x}$, a sparse representation of the input vector y using SSP [19] where $y = A\hat{x}$
- Add sufficient Laplasian noise according to $[1, 106, 107]$

Post-processing:

- $y^* = \Psi x^*$

Output: the noise-added differentially private version of SNP frequencies y^*

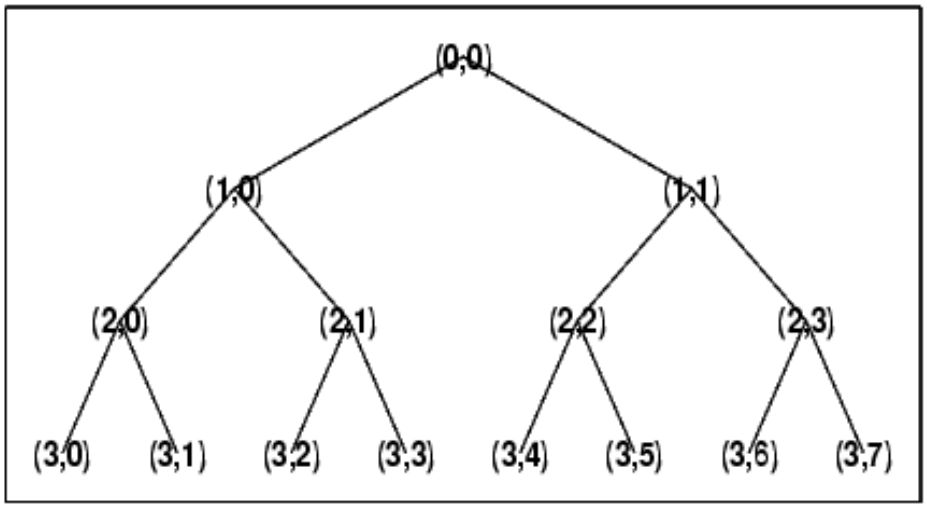


Figure 4.1: The Haar wavelet tree structure of [$(0,0)$, $(1,0)$, $(1,1)$, $(2,0)$, $(2,1)$, $(2,2)$, $(2,3)$, $(3,0)$, $(3,1)$, $(3,2)$, $(3,3)$, $(3,4)$, $(3,5)$, $(3,6)$, $(3,7)$]. Note that each (x,y) shows one entry in the array. The node $(0,0)$ is the root of the wavelet transformation tree and the nodes $(3,t)$, where t is a number between 0 and 7, are the leaves of the wavelet transformation tree.

4.3 Experimental Results

To evaluate the proposed method, I participated in the challenge at the iDASH (UCSD–based National Center for Biomedical Computing: NIH U54HL108460) workshop [1] on March 24, 2014 in La Jolla, California and the results are available online at [9]. The challenge was about sharing aggregate human genomic data (i.e., allele frequencies) to preserve the privacy of the data and to maximize the utility of the data for Genome–Wide Association Studies (GWAS). I applied the proposed method on the two datasets from the case and control groups of individuals: the case group includes 411 individuals from the Personal Genome Project [108] [2], and the control data includes 174 participants from the CEU population in HapMap [109] [3]. There are two test sets: the first one consists of 311 SNP sites of the human

[1] http://www.humangenomeprivacy.org
[2] http://www.personalgenomes.org/
[3] http://hapmap.ncbi.nlm.nih.gov/index.html.en

chromosome 2 and The second one consists of 600 SNP sites of human chromosome 10. Table 4.1 shows the result of proposed method on these two test sets in compare to the SNP–Based baseline of the challenge organizers.

Data Set		SNP–Based baseline		Proposed Method		Number of the significant SNPs
Dataset 1	Power	0.05		0.61		
	Cutoffs	TPR	FPR	TPR	FPR	
	5×10^{-2}	0.864	0.844	1.0	0.941	22
	10^{-3}	0.632	0.774	1.0	0.884	19
	10^{-5}	0.642	0.700	1.0	0.879	14
Dataset 2	Power	0.4		0.005		
	Cutoffs	TPR	FPR	TPR	FPR	
	5×10^{-2}	0.933	0.924	1.0	0.958	45
	10^{-3}	0.800	0.862	1.0	0.909	15
	10^{-5}	0.625	0.788	1.0	0.876	8

Table 4.1: Results of the proposed method on the challenge datasets. The Dataset 1 refers to 200 participants with 311 SNPs on chromosome 2 and Dataset 2 refers to 200 participants with 610 SNPs on chromosome 10. The "Power" row is the ratio of identifiable individuals using the likelihood ratio test in the case group. The false positive rate (FPR) and true positive rate based on χ^2 test are listed per different cutoff threshold. In addition, the last column corresponds to the number of significant SNPs.

Figures 4.2 and 4.3 are the result of the proposed method disturbed data for identification of an individual in the data base. Figures 4.4 and 4.5 show the utility evaluation of the proposed method results on the online evaluation tool WIDGET [9].

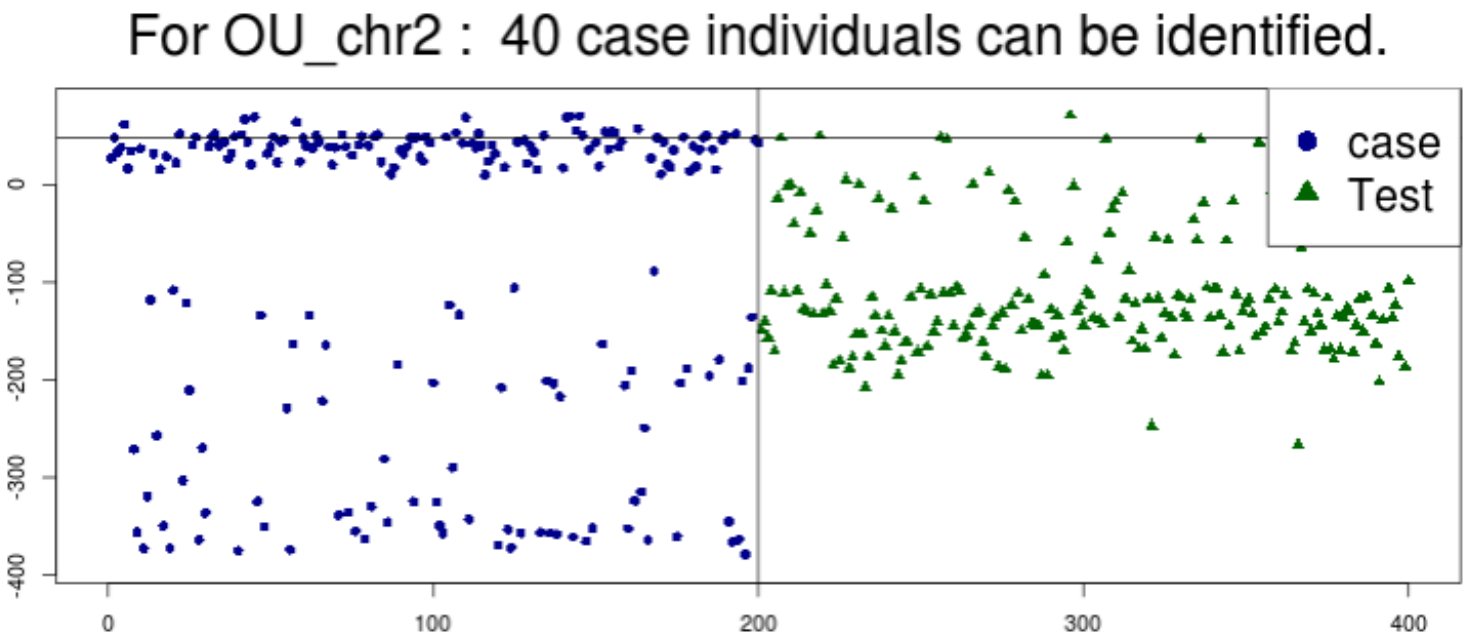


Figure 4.2: The online privacy evaluation [9] of the proposed method on chromosome 2 with p-value 0.01

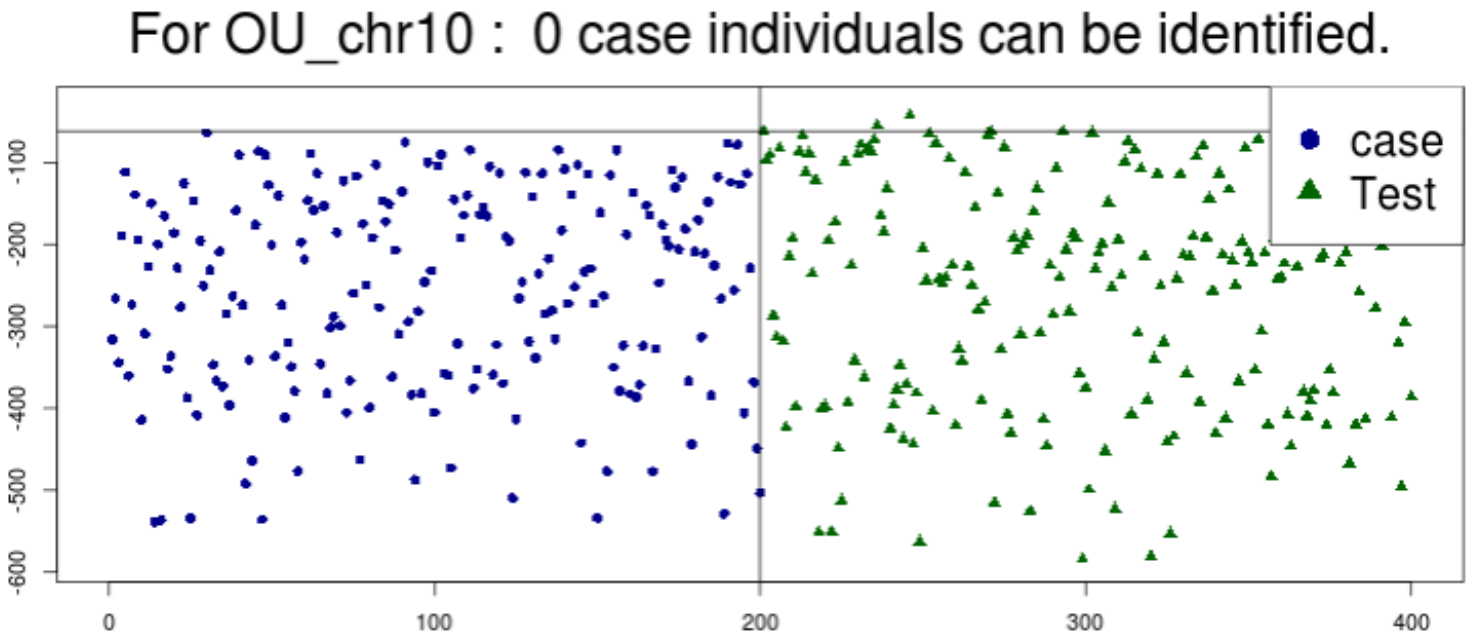
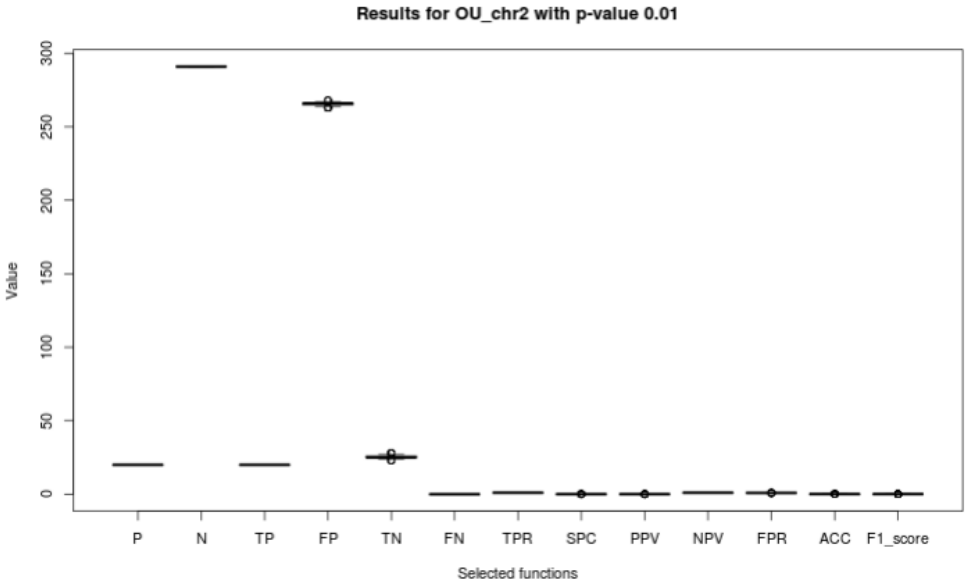


Figure 4.3: The online privacy evaluation [9] of the proposed method on chromosome 10 with p-value 0.01

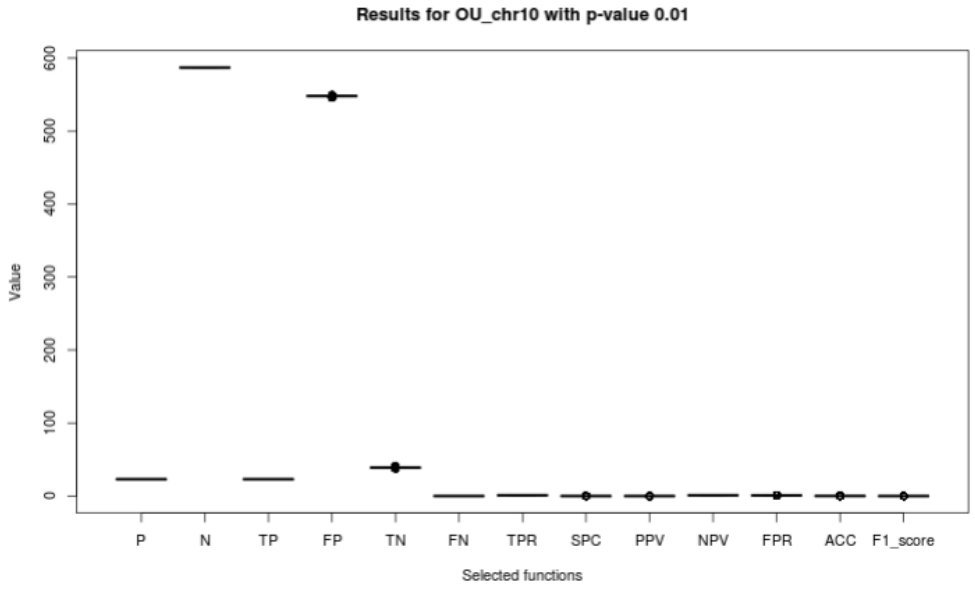



```
Notations:   P - Positive;                          N - Negative;
            TP - True Positive;                    FP - False Positive;
            TN - True Negative;                    FN - False Negative;
           SPC - True Negative Rate (Specificity); PPV - positive predictive value (Precision)
           NPV - Negative Predictive Value;        FPR - False Positive Rate
           TPR - True Positive Rate (Sensitivity); ACC - Accuracy;
Summarize results:
  1 result was selected.
    Result "OU_chr2" has 311 SNPs over 1000 runs with p-value:0.01

==============Statistics for results: OU_chr2=========================
For p-value < 0.01, number of significant SNPs in Case Group is 20
After adding noise:
              mean    median       min       max        sd    n
P         20.00000  20.00000  20.00000  20.00000 0.0000000 1000
N        291.00000 291.00000 291.00000 291.00000 0.0000000 1000
TP        20.00000  20.00000  20.00000  20.00000 0.0000000 1000
FP       265.59300 266.00000 263.00000 268.00000 0.7320430 1000
TN        25.40700  25.00000  23.00000  28.00000 0.7320430 1000
FN         0.00000   0.00000   0.00000   0.00000 0.0000000 1000
TPR        1.00000   1.00000   1.00000   1.00000 0.0000000 1000
SPC        0.08731   0.08591   0.07904   0.09622 0.0025156 1000
PPV        0.07003   0.06993   0.06944   0.07067 0.0001794 1000
NPV        1.00000   1.00000   1.00000   1.00000 0.0000000 1000
FPR        0.91269   0.91409   0.90378   0.92096 0.0025156 1000
ACC        0.14600   0.14469   0.13826   0.15434 0.0023538 1000
F1_score   0.13089   0.13072   0.12987   0.13201 0.0003134 1000
```

Figure 4.4: The online utility evaluation [9] of the proposed method on chromosome 2 with p-value 0.01

The power of the likelihood ratio test is a privacy risk measure which the number of case individuals who can be recognized with confidence level more than a threshold. The lower likelihood ratio power level shows that the perturbed data has less risk of reidentification. In addition the utility of the privacy protection method




```
Notations:   P - Positive;                          N - Negative;
            TP - True Positive;                    FP - False Positive;
            TN - True Negative;                    FN - False Negative;
           SPC - True Negative Rate (Specificity);  PPV - positive predictive value (Precision)
           NPV - Negative Predictive Value;         FPR - False Positive Rate
           TPR - True Positive Rate (Sensitivity);  ACC - Accuracy;
Summarize results:
  1 result was selected.
    Result "OU_chr10" has 610 SNPs over 1000 runs with p-value:0.01

==============Statistics for results: OU_chr10========================
For p-value < 0.01, number of significant SNPs in Case Group is 23
After adding noise:
              mean    median       min       max         sd    n
P         23.00000  23.00000  23.00000  23.00000 0.00000000 1000
N        587.00000 587.00000 587.00000 587.00000 0.00000000 1000
TP        23.00000  23.00000  23.00000  23.00000 0.00000000 1000
FP       548.05400 548.00000 546.00000 550.00000 0.45308861 1000
TN        38.94600  39.00000  37.00000  41.00000 0.45308861 1000
FN         0.00000   0.00000   0.00000   0.00000 0.00000000 1000
TPR        1.00000   1.00000   1.00000   1.00000 0.00000000 1000
SPC        0.06635   0.06644   0.06303   0.06985 0.00077187 1000
PPV        0.04028   0.04028   0.04014   0.04042 0.00003199 1000
NPV        1.00000   1.00000   1.00000   1.00000 0.00000000 1000
FPR        0.93365   0.93356   0.93015   0.93697 0.00077187 1000
ACC        0.10155   0.10164   0.09836   0.10492 0.00074277 1000
F1_score   0.07743   0.07744   0.07718   0.07770 0.00005912 1000
```

Figure 4.5: The online utility evaluation [9] of the proposed method on chromosome 10 with p-value 0.01

can be measured based on the χ^2 test to detect the significants SNPs with different cutoff p–value. As Table 4.1 shows the proposed method has the higher true positive significant detected on the perturbed data.

In addition to the challenge, I select 180 SNPs of the Personal Genome Project

[108] and partition it into two subsets. The first subset used as control group. I modified the frequencies of randomly selected SNPs on second subset and generated 3 new datasets with different levels of utility called A,B,C with 27,9 and 4 significant. After adding noise, the p-value of a SNP reported by a association test, χ^2, of these three datasets can be compared with each other to select the best dataset as "Best Pick" which is A. Also, I check the results to find the "Correct Order" of datasets A better than B better than C. The results are shown in Table 4.2. The numbers in the table are the percentage of successful detection of best dataset and the correct order of them. As this results shows the proposed method dominates the other methods.

	Correct Order %	Best Pick %
SNP–Based baseline	19.34	34.54
Reference [1]	24.85	47.52
Proposed Method	25.02	75.12

Table 4.2: Results of the SNP–Based baseline, Reference [1] and the proposed method for best pick and correct order.

4.4 Conclusion

In this chapter, I present a privacy preserving genomic data dissemination algorithm based on the compressed sensing. The input sequences to the proposed method are the genomic nucleotide and I process the input genomic sequences to find the location of the SNPs. Then I specify an input vector of the SNPs frequencies, a random matrix and the wavelet transformation matrix. To making sure that the amount of the noise that I am adding to sparse representation is enough to protect the differential privacy, I add Laplasian noise λ to all element of the sparse representation according to [107]. My experimental results shows that the proposed method outperform the other methods.

CHAPTER 5

CONCLUSION

In this dissemination, I propose a novel nucleotide sequence Indexing and alignment method based on empirical transitional probability, sparse coding and belief propagation to compare the similarity of the nucleotide sequences inspired by my recent works, 3D-SCoBeP described in chapter 2. The proposed method builds a transition matrix based on the neighboring nucleotides of an input sequence and then reduces the search space by selecting the top K most similar subsequences based on their distances. The proposed algorithm selects candidate nucleotides by using sparse coding with an over-completed dictionary, which was constructed from the nucleotides of reference sequence in the indexing step. BP algorithm is then applied to select the best matches. The proposed method is robust to insertions, deletions, and mutations in the genome sequences when comparing with SOAP aligner and BWA. Finally, the proposed method is able to process much longer sequences then our previous 1D-SCoBeP approach. Through experimental results, I showed that the proposed algorithm are comparable to SOAP aligner [6], BWA [5] and 1D-SCoBeP [4] in terms of the alignment accuracy. In addition, I present a privacy preserving genomic data dissemination algorithm based on the compressed sensing. The input sequences to the proposed method are the genomic nucleotide and I process the input genomic sequences to find the location of the SNPs. Then I specify an input vector of the SNPs frequencies, a random matrix and the wavelet transformation matrix. Then I add Laplasian noise λ to all element of the sparse rep-

resentation. My experimental results shows that the proposed method outperform the other methods.

Bibliography

[1] Y. D. Li, Z. Zhang, M. Winslett, and Y. Yang, "Compressive mechanism: Utilizing sparse representation in differential privacy," in *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*. ACM, 2011, pp. 177–182.

[2] A. Myronenko and X. Song, "Intensity-based image registration by minimizing residual complexity," *Medical Imaging, IEEE Transactions on*, vol. 29, no. 11, pp. 1882–1891, 2010.

[3] S. Periaswamy and H. Farid, "Elastic registration in the presence of intensity variations," *Medical Imaging, IEEE Transactions on*, vol. 22, no. 7, pp. 865–874, 2003.

[4] A. Roozgard, N. Barzigar, S. Wang, X. Jiang, L. Ohno-Machado, and S. Cheng, "Nucleotide sequence alignment using sparse coding and belief propagation," in *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*. IEEE, 2013, pp. 588–591.

[5] H. Li and R. Durbin, "Fast and accurate short read alignment with burrows–wheeler transform," *Bioinformatics*, vol. 25, no. 14, pp. 1754–1760, 2009.

[6] R. Li, C. Yu, Y. Li, T. Lam, S. Yiu, K. Kristiansen, and J. Wang, "Soap2: an improved ultrafast tool for short read alignment," *Bioinformatics*, vol. 25, no. 15, pp. 1966–1967, 2009.

[7] N. C. Institute, "The national center for biotechnology information," http://www.ncbi.nlm.nih.gov/genome?db=genome, January 2013. [Online]. Available: http://www.ncbi.nlm.nih.gov/genome?db=genome

[8] S. C. University of California, "Cancer genomics hub," https://cghub.ucsc.edu/datasets/benchmark-download.html, January 2013. [Online]. Available: https://cghub.ucsc.edu/datasets/benchmark-download.html

[9] S. Wang, W. Wei, Z. Ji, Y. Zhao, X. Jiang, X. Wang, H. Tang, and L. Ohno-Machado. Widget: a web interface for dynamic genome-privacy evaluation. [Online]. Available: https://humangenomeprivacy.ucsd-dbmi.org/

[10] T. Hampton, "Cancer genome atlas," *JAMA: The Journal of the American Medical Association*, vol. 296, no. 16, pp. 1958–1958, 2006.

[11] N. Siva, "1000 genomes project," *Nature biotechnology*, vol. 26, no. 3, pp. 256–256, 2008.

[12] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 498–519, 2001.

[13] F. V. Jensen, *An introduction to Bayesian networks*. UCL press London, 1996, vol. 210.

[14] Y. A. Rozanov, *Markov random fields*. Springer, 1982.

[15] R. Baraniuk, "Compressive sensing," *IEEE Signal Processing Magazine*, vol. 24, no. 4, pp. 118–120, 2007.

[16] Y. Pang, X. Li, and Y. Yuan, "Robust tensor analysis with l1-norm," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 20, no. 2, pp. 172–178, 2010.

[17] A. Yang, S. Sastry, A. Ganesh, and Y. Ma, "Fast 1-minimization algorithms and an application in robust face recognition: A review," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*. IEEE, 2010, pp. 1849–1852.

[18] R. Maleh, A. Gilbert, and M. Strauss, "Sparse gradient image reconstruction done faster," in *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, vol. 2. IEEE, 2007, pp. II–77.

[19] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *Information Theory, IEEE Transactions on*, vol. 55, no. 5, pp. 2230–2249, 2009.

[20] E. J. Candes, "The restricted isometry property and its implications for compressed sensing," *Comptes Rendus Mathematique*, vol. 346, no. 9, pp. 589–592, 2008.

[21] N. Barzigar, A. Roozgard, P. Verma, and S. Cheng, "A video super resolution framework using scobep," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2013.

[22] N. Barzigar, A. Roozgard, S. Cheng, and P. Verma, "Scobep: Dense image registration using sparse coding and belief propagation," *Journal of Visual Communication and Image Representation*, vol. 24, no. 2, pp. 137 – 147, 2013, sparse Representations for Image and Video Analysis. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1047320312001319

[23] Y. C. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*. IEEE, 1993, pp. 40–44.

[24] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin, "A simple proof of the restricted isometry property for random matrices," *Constructive Approximation*, vol. 28, no. 3, pp. 253–263, 2008.

[25] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography*. Springer, 2006, pp. 265–284.

[26] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*. IEEE, 2007, pp. 94–103.

[27] B. Zitova and J. Flusser, "Image registration methods: a survey," *Image and vision computing*, vol. 21, no. 11, pp. 977–1000, 2003.

[28] A. Roozgard, S. Cheng, and H. Liu, "Malignant nodule detection on lung ct scan images with kernel rx-algorithm," in *IEEE-EMBS International Conference on Biomedical and Health Informatics*, Hong Kong, Shenzhen, 2012.

[29] M. Bro-Nielsen, "Medical image registration and surgery simulation," *IMM-DTU PhD thesis*, 1996.

[30] J. Schnabel, D. Rueckert, *et al.*, "A generic framework for non-rigid registration based on non-uniform multi-level free-form deformations," in *Medical Image Computing and Computer-Assisted Intervention*. Springer, 2001, pp. 573–581.

[31] D. Hill, D. Hawkes, J. Crossman, M. Gleeson, T. Cox, E. Bracey, A. Strong, and P. Graves, "Registration of mr and ct images for skull base surgery using point-like anatomical features," *British journal of radiology*, vol. 64, no. 767, pp. 1030–1035, 1991.

[32] A. Lanfranco, A. Castellanos, J. Desai, and W. Meyers, "Robotic surgery: a current perspective," *Annals of Surgery*, vol. 239, no. 1, p. 14, 2004.

[33] J. Maintz and M. Viergever, "A survey of medical image registration," *Medical image analysis*, vol. 2, no. 1, pp. 1–36, 1998.

[34] S. Periaswamy and H. Farid, "Medical image registration with partial data," *Medical Image Analysis*, vol. 10, no. 3, pp. 452–464, 2006.

[35] H. Shum and R. Szeliski, "Systems and experiment paper: Construction of panoramic image mosaics with global and local alignment," *International Journal of Computer Vision*, vol. 36, no. 2, pp. 101–130, 2000.

[36] B. Glocker, N. Komodakis, G. Tziritas, N. Navab, and N. Paragios, "Dense image registration through mrfs and efficient linear programming," *Medical Image Analysis*, vol. 12, no. 6, pp. 731–741, 2008.

[37] M. Brown and D. Lowe, "Automatic panoramic image stitching using invariant features," *International Journal of Computer Vision*, vol. 74, no. 1, pp. 59–73, 2007.

[38] A. Roche, G. Malandain, N. Ayache, *et al.*, "Unifying maximum likelihood approaches in medical image registration," *Inria*, 1999.

[39] O. Musse, F. Heitz, and J. Armspach, "Topology preserving deformable image matching using constrained hierarchical parametric models," *Image Processing, IEEE Transactions on*, vol. 10, no. 7, pp. 1081–1093, 2001.

[40] S. Cain, M. Hayat, and E. Armstrong, "Projection-based image registration in the presence of fixed-pattern noise," *Image Processing, IEEE Transactions on*, vol. 10, no. 12, pp. 1860–1872, 2001.

[41] J. Liu, B. Vemuri, and J. Marroquin, "Local frequency representations for robust multimodal image registration," *Medical Imaging, IEEE Transactions on*, vol. 21, no. 5, pp. 462–469, 2002.

[42] A. Collignon, D. Vandermeulen, P. Suetens, and G. Marchal, "3d multi-modality medical image registration using feature space clustering," in *Computer Vision, Virtual Reality and Robotics in Medicine*. Springer, 1995, pp. 193–204.

[43] M. Elbakary and M. Sundareshan, "Accurate representation of local frequency using a computationally efficient gabor filter fusion approach with application to image registration," *Pattern recognition letters*, vol. 26, no. 14, pp. 2164–2173, 2005.

[44] M. Staring, U. van der Heide, S. Klein, M. Viergever, and J. Pluim, "Registration of cervical mri using multifeature mutual information," *Medical Imaging, IEEE Transactions on*, vol. 28, no. 9, pp. 1412–1421, 2009.

[45] S. Klein, M. Staring, and J. Pluim, "Evaluation of optimization methods for nonrigid medical image registration using mutual information and b-splines," *Image Processing, IEEE Transactions on*, vol. 16, no. 12, pp. 2879–2890, 2007.

[46] S. Klein, J. Pluim, M. Staring, and M. Viergever, "Adaptive stochastic gradient descent optimisation for image registration," *International journal of computer vision*, vol. 81, no. 3, pp. 227–239, 2009.

[47] B. Likar and F. Pernuš, "A hierarchical approach to elastic registration based on mutual information," *Image and Vision Computing*, vol. 19, no. 1, pp. 33–44, 2001.

[48] M. Sabuncu and P. Ramadge, "Using spanning graphs for efficient image registration," *Image Processing, IEEE Transactions on*, vol. 17, no. 5, pp. 788–797, 2008.

[49] G. Penney, P. Batchelor, D. Hill, D. Hawkes, and J. Weese, "Validation of a two-to three-dimensional registration algorithm for aligning preoperative ct images and intraoperative fluoroscopy images," *Medical physics*, vol. 28, p. 1024, 2001.

[50] J. Hipwell, G. Penney, R. McLaughlin, K. Rhode, P. Summers, T. Cox, J. Byrne, J. Noble, and D. Hawkes, "Intensity-based 2-d-3-d registration of cerebral angiograms," *Medical Imaging, IEEE Transactions on*, vol. 22, no. 11, pp. 1417–1426, 2003.

[51] J. Byrne, C. Colominas, J. Hipwell, T. Cox, J. Noble, G. Penney, and D. Hawkes, "Assessment of a technique for 2d–3d registration of cerebral intra-arterial angiography," *British journal of radiology*, vol. 77, no. 914, pp. 123–128, 2004.

[52] C. Liu, J. Yuen, and A. Torralba, "Sift flow: Dense correspondence across scenes and its applications," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 5, pp. 978–994, 2011.

[53] T. Tang and A. Chung, "Non-rigid image registration using graph-cuts," in *Proceedings of the 10th international conference on Medical image computing and computer-assisted intervention-Volume Part I.* Springer-Verlag, 2007, pp. 916–924.

[54] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *The Journal of Machine Learning Research*, vol. 11, pp. 19–60, 2010.

[55] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 11, pp. 1222–1239, 2001.

[56] N. Barzigar, A. Roozgard, S. Cheng, and P. Verma, "Scobep: Dense image registration using sparse coding and belief propagation," *Journal of Visual Communication and Image Representation*, 2012.

[57] A. Roozgard, N. Barzigar, S. Cheng, and P. Verma, "Medical image registration using sparse coding and belief propagation," in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, Aug 2012, pp. 1141–1144.

[58] D. Lowe, "Object recognition from local scale-invariant features," in *International Conference on Computer Vision*, vol. 2, 1999, pp. 1150–1157.

[59] H. G. Feichtinger, *Gabor analysis and algorithms: Theory and applications.* Birkhauser, 1998.

[60] S. Cheng, V. Stankovic, and L. Stankovic, "Improved sift-based image registration using belief propagation," in *Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing-Volume 00*. IEEE Computer Society, 2009, pp. 2909–2912.

[61] B. Fulkerson, A. Vedaldi, and S. Soatto, "Class segmentation and object localization with superpixel neighborhoods," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 670–677.

[62] N. C. Institute, "The cancer imaging archive," https://wiki.cancerimagingarchive.net/display/Public/LIDC-IDRI, September 2011. [Online]. Available: https://wiki.cancerimagingarchive.net/display/Public/LIDC-IDRI

[63] H. Li, B. Manjunath, and S. Mitra, "A contour-based approach to multisensor image registration," *Image Processing, IEEE Transactions on*, vol. 4, no. 3, pp. 320–334, 1995.

[64] M. Haque, M. Biswas, M. Pickering, and M. Frater, "A low-complexity image registration algorithm for global motion estimation," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 3, pp. 426–433, 2012.

[65] T. M. Lehmann, C. Gonner, and K. Spitzer, "Addendum: B-spline interpolation in medical image processing," *Medical Imaging, IEEE Transactions on*, vol. 20, no. 7, pp. 660–665, 2001.

[66] M. Meyerson, S. Gabriel, and G. Getz, "Advances in understanding cancer genomes through second-generation sequencing," *Nature Reviews Genetics*, vol. 11, no. 10, pp. 685–696, 2010.

[67] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of molecular biology*, vol. 48, no. 3, pp. 443–453, 1970.

[68] B. Morgenstern, "Dialign 2: improvement of the segment-to-segment approach to multiple sequence alignment." *Bioinformatics*, vol. 15, no. 3, pp. 211–218, 1999.

[69] N. Bray, I. Dubchak, and L. Pachter, "Avid: A global alignment program," *Genome research*, vol. 13, no. 1, pp. 97–102, 2003.

[70] M. Brudno, C. B. Do, G. M. Cooper, M. F. Kim, E. Davydov, E. D. Green, A. Sidow, S. Batzoglou, *et al.*, "Lagan and multi-lagan: efficient tools for large-scale multiple alignment of genomic dna," *Genome research*, vol. 13, no. 4, pp. 721–731, 2003.

[71] T. F. Smith and M. S. Waterman, "Comparison of biosequences," *Advances in Applied Mathematics*, vol. 2, no. 4, pp. 482–489, 1981.

[72] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of molecular biology*, vol. 215, no. 3, pp. 403–410, 1990.

[73] M. Brudno and B. Morgenstern, "Fast and sensitive alignment of large genomic sequences," in *Bioinformatics Conference, 2002. Proceedings. IEEE Computer Society.* IEEE, 2002, pp. 138–147.

[74] S. Schwartz, W. J. Kent, A. Smit, Z. Zhang, R. Baertsch, R. C. Hardison, D. Haussler, and W. Miller, "Human–mouse alignments with blastz," *Genome research*, vol. 13, no. 1, pp. 103–107, 2003.

[75] M. Brudno, S. Malde, A. Poliakov, C. B. Do, O. Couronne, I. Dubchak, and S. Batzoglou, "Glocal alignment: finding rearrangements during alignment," *Bioinformatics*, vol. 19, no. suppl 1, pp. i54–i62, 2003.

[76] J. D. Thompson, D. G. Higgins, and T. J. Gibson, "Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice," *Nucleic acids research*, vol. 22, no. 22, pp. 4673–4680, 1994.

[77] M. Larkin, G. Blackshields, N. Brown, R. Chenna, P. McGettigan, H. McWilliam, F. Valentin, I. Wallace, A. Wilm, R. Lopez, *et al.*, "Clustal w and clustal x version 2.0," *Bioinformatics*, vol. 23, no. 21, pp. 2947–2948, 2007.

[78] A. S. Schwartz and L. Pachter, "Multiple alignment by sequence annealing," *Bioinformatics*, vol. 23, no. 2, pp. e24–e29, 2007.

[79] A. E. Darling, B. Mau, and N. T. Perna, "progressivemauve: multiple genome alignment with gene gain, loss and rearrangement," *PloS one*, vol. 5, no. 6, p. e11147, 2010.

[80] S. Kurtz, A. Phillippy, A. L. Delcher, M. Smoot, M. Shumway, C. Antonescu, S. L. Salzberg, *et al.*, "Versatile and open software for comparing large genomes," *Genome Biol*, vol. 5, no. 2, p. R12, 2004.

[81] G. S. Slater and E. Birney, "Automated generation of heuristics for biological sequence comparison," *BMC bioinformatics*, vol. 6, no. 1, p. 31, 2005.

[82] M. N. Price, P. S. Dehal, and A. P. Arkin, "Fasttree: computing large minimum evolution trees with profiles instead of a distance matrix," *Molecular biology and evolution*, vol. 26, no. 7, pp. 1641–1650, 2009.

[83] ——, "Fasttree 2–approximately maximum-likelihood trees for large alignments," *Plos one*, vol. 5, no. 3, p. e9490, 2010.

[84] S. Guindon and O. Gascuel, "A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood," *Systematic biology*, vol. 52, no. 5, pp. 696–704, 2003.

[85] A. Stamatakis, "Raxml-vi-hpc: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models," *Bioinformatics*, vol. 22, no. 21, pp. 2688–2690, 2006.

[86] S. Batzoglou, "The many faces of sequence alignment," *Briefings in bioinformatics*, vol. 6, no. 1, pp. 6–22, 2005.

[87] C. Dewey and L. Pachter, "Evolution at the nucleotide level: the problem of multiple whole-genome alignment," *Human Molecular Genetics*, vol. 15, no. suppl 1, pp. R51–R56, 2006.

[88] M. Blanchette, W. J. Kent, C. Riemer, L. Elnitski, A. F. Smit, K. M. Roskin, R. Baertsch, K. Rosenbloom, H. Clawson, E. D. Green, *et al.*, "Aligning multiple genomic sequences with the threaded blockset aligner," *Genome research*, vol. 14, no. 4, pp. 708–715, 2004.

[89] A. Darling, B. Mau, F. Blattner, and N. Perna, "Mauve: multiple alignment of conserved genomic sequence with rearrangements," *Genome research*, vol. 14, no. 7, pp. 1394–1403, 2004.

[90] I. Dubchak, A. Poliakov, A. Kislyuk, and M. Brudno, "Multiple whole-genome alignments without a reference organism," *Genome research*, vol. 19, no. 4, pp. 682–689, 2009.

[91] M. Höhl, S. Kurtz, and E. Ohlebusch, "Efficient multiple genome alignment," *Bioinformatics*, vol. 18, no. suppl 1, pp. S312–S320, 2002.

[92] B. Paten, J. Herrero, K. Beal, S. Fitzgerald, and E. Birney, "Enredo and pecan: genome-wide mammalian consistency-based multiple alignment with paralogs," *Genome research*, vol. 18, no. 11, pp. 1814–1828, 2008.

[93] R. Bradley, A. Roberts, M. Smoot, S. Juvekar, J. Do, C. Dewey, I. Holmes, and L. Pachter, "Fast statistical alignment," *PLoS computational biology*, vol. 5, no. 5, p. e1000392, 2009.

[94] R. Edgar, "Muscle: multiple sequence alignment with high accuracy and high throughput," *Nucleic acids research*, vol. 32, no. 5, pp. 1792–1797, 2004.

[95] J. Bartoš, Č. Vlček, F. Choulet, M. Džunková, K. Cviková, J. Šafář, H. Šimková, J. Pačes, H. Strnad, P. Sourdille, *et al.*, "Intraspecific sequence comparisons reveal similar rates of non-collinear gene insertion in the b and d genomes of bread wheat," *BMC Plant Biology*, vol. 12, no. 1, p. 155, 2012.

[96] R. Song and J. Messing, "Gene expression of a gene family in maize based on noncollinear haplotypes," *Proceedings of the National Academy of Sciences*, vol. 100, no. 15, pp. 9055–9060, 2003.

[97] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?" *Vision research*, vol. 37, no. 23, pp. 3311–3325, 1997.

[98] C.-M. Liu, T. Wong, E. Wu, R. Luo, S.-M. Yiu, Y. Li, B. Wang, C. Yu, X. Chu, K. Zhao, *et al.*, "Soap3: ultra-fast gpu-based parallel alignment tool for short reads," *Bioinformatics*, vol. 28, no. 6, pp. 878–879, 2012.

[99] P. D. Vouzis and N. V. Sahinidis, "Gpu-blast: using graphics processors to accelerate protein sequence alignment," *Bioinformatics*, vol. 27, no. 2, pp. 182–188, 2011.

[100] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik, "Blobworld: A system for region-based image indexing and retrieval," in *Visual Information and Information Systems*. Springer, 1999, pp. 509–517.

[101] D. Doermann, "The indexing and retrieval of document images: A survey," *Computer Vision and Image Understanding*, vol. 70, no. 3, pp. 287–298, 1998.

[102] C. Liu and H. Wechsler, "Robust coding schemes for indexing and retrieval from large face databases," *Image Processing, IEEE Transactions on*, vol. 9, no. 1, pp. 132–137, 2000.

[103] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing: Closing the gap between performance and complexity," DTIC Document, Tech. Rep., 2008.

[104] C. E. Heil and D. F. Walnut, "Continuous and discrete wavelet transforms," *SIAM review*, vol. 31, no. 4, pp. 628–666, 1989.

[105] C. Mulcahy, "Image compression using the haar wavelet transform," *Spelman Science and Mathematics Journal*, vol. 1, no. 1, pp. 22–31, 1997.

[106] X. Xiao, G. Wang, and J. Gehrke, "Differential privacy via wavelet transforms," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 23, no. 8, pp. 1200–1214, 2011.

[107] G. Cormode, M. Procopiuc, D. Srivastava, and T. T. Tran, "Differentially private publication of sparse data," *arXiv preprint arXiv:1103.0825*, 2011.

[108] G. M. Church, "The personal genome project," *Molecular Systems Biology*, vol. 1, no. 1, 2005. [Online]. Available: http://www.personalgenomes.org/

[109] R. A. Gibbs, J. W. Belmont, P. Hardenbol, T. D. Willis, F. Yu, H. Yang, L.-Y. Ch'ang, W. Huang, B. Liu, Y. Shen, *et al.*, "The international hapmap project," *Nature*, vol. 426, no. 6968, pp. 789–796, 2003.

DEDICATION

to

My father and mother
Mohammad Rouzgard and Sorayya Moradi

AND

My wife
Nafise Barzigar