3D protein structure analysis based on multi-view images from molecular visualization

March 2016

SURYANTO. Chendra Hadi

3D protein structure analysis based on multi-view images from molecular visualization

Graduate School of Systems and Information Engineering
University of Tsukuba

March 2016

SURYANTO. Chendra Hadi

Acknowledgements

I would like to thank my academic supervisor, Prof. Kazuhiro Fukui, who is very caring and always supports my study and research since I was still a research student until now. It is a privilege for me to be able to study subspace-based methods with him. It is a very great experience for me being under his guidance, not only in research but also life.

Then, I would like to thank Prof. Hideitsu Hino, for the idea, discussion, and support on my research. It was also very enjoyable when studying statistical learning and manifold learning with him.

I also would like to thank Prof. Jing-hao Xue for the discussion on my research and the research internship experience.

Next, I would like to thank Prof. Hotaka Takizawa, Prof. Itaru Kitahara, Prof. Keisuke Kameyama, and Prof. Kiyoshi Hoshino for the guidance and useful feedbacks as the examination committees of this thesis.

Also, I would like to thank Ms. Hiroko Sawabe, who works as administrative staff in the Computer Vision laboratory, for supporting paperworks during my study for six years.

Additionally, I would like to thank all members of Computer Vision laboratory, both active members and alumni for the discussion. At the same time, I would like to thank MEXT 2010 scholarship recipients, especially to those who are still keeping in touch and motivating each other.

I also would like to thank my family and friends for the moral support and encouragements.

Finally, I would like to express my gratitude to the Ministry of Education, Culture, Sports, Science and Technology Japan for sponsoring my study in Japan for six years.

Abstract

Analysis of 3D protein structures is an important task in structural biology. In the protein structural analysis, protein structures are commonly compared by applying alignment and superposition to the backbone structures and computing the root mean square deviation (RMSD). This approach, however, can be suboptimal because the results depend on the alignment techniques. Moreover, RMSD, which relies on the alignment results, does not always proportionate to the number of the aligned substructures. These lead to the proposal of many protein structural comparison methods. While most of them are based on using geometrical information directly, here we propose a new method for protein structural comparison based on the set of the multi-view molecular visualization images of the protein. In the proposed method, the set of the images is then modeled as a subspace and the similarity between two protein structures is defined by the canonical angles between the corresponding subspaces. The main advantage is that precise alignment is not needed. Besides, multiple types of protein visualization can be used to enrich the distinctive information. The effectiveness of the proposed method was evaluated through the classification experiments on seven classes of proteins, where the proposed method outperformed the conventional methods. We also designed and developed an online protein comparison system, called View-based Protein Comparison (VPC), as an implementation of the proposed method. Finally, as many methods have been proposed for measurement of protein structural similarity, we proposed a method for distance metric combination, by generalizing the concept of large margin nearest neighbor (LMNN). While LMNN was originally proposed to learn Mahalanobis-based distance metric from a set of feature vectors, we learnt a weight combination from a set of distance metrics, by introducing three loss functions to the objective function of LMNN. The validity of the proposed method was demonstrated through experiments on two public dataset of Ding Dubchak and ENZYME dataset.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Proteins have very important roles in carrying out various biological processes in every living organism. For example, hemoglobin, which is a protein that densely exists in a red blood cell, has a function as a transport agent for oxygen and carbon dioxide. Keratin is a protein that provides strength for structures such as human skin or hair. Enzymes are protein molecules that work as biological catalysts. The function of a protein is determined by its 3D structure. This points to the importance of protein structure comparison as the fundamental tool for protein analysis. In this chapter, firstly, the overview of protein in an organism is provided in Chapter 1.1. Secondly, the importance of protein analysis is elaborated in Chapter 1.2. Then, the motivation and objective of this thesis are provided in Chapter 1.3. Finally, the organization of this thesis is given in Chapter 1.4.

1.1 Overview of protein

The word "protein" was first coined by Jacob Berzelius in 1838, which was later used by Mulder in scientific literature in the same year [1]. It comes from the Greek word "proteos" that has meaning of primary importance. Protein is a very important substance in a living thing. Living thing can be defined as a unit that can conduct chemical activities including reproduction and evolution [2]. Figure 1.1 shows the general hierarchical composition of a *multicellular* living thing with human as an example. One *multicellular* organism consists of a collection of organ systems where each organ system consists of a number of organs to carry out specific functions in the organism. For example, respiratory system of human consists of organs such as nasal cavity, pharynx, larynx, trachea, bronchi, lungs, and diaphragm. Each organ contains tissues, which are

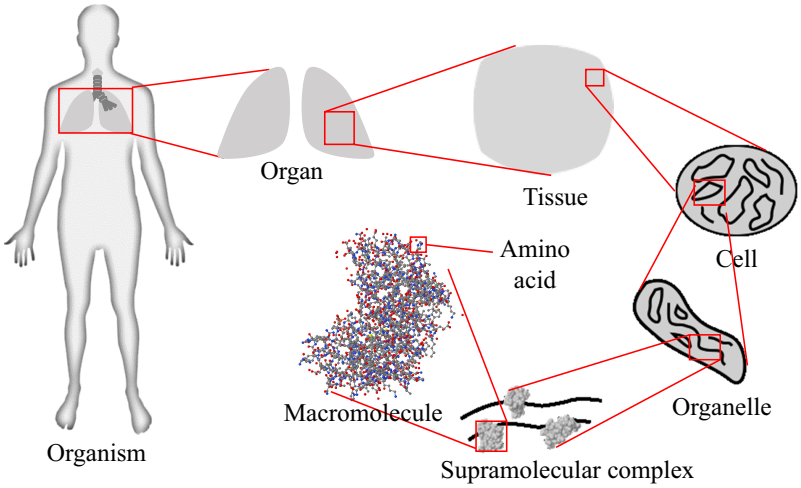

Figure 1.1: Hierarchical composition of a living thing.

groups of similar cells that perform similar functions. Tissues are composed of cells, which are the basic unit of life. Cell that has simple structure without *nucleus* is categorized as *prokaryotic*. Cell with complex structure is categorized as *eukaryotic*. Cell in the latter category is composed of units called organelles that perform specific cellular functions. Organelle contains supramolecular complexes, each of which is composed of macromolecules that are involved in mostly every cellular process [3]. Macromolecules, literally can be translated to "large molecules", are grouped into four classes, namely, carbohydrates, lipids, nucleic acids, and proteins. Here, we focus on the protein, the structure of which is composed of amino acids. There are many fields in bioinformatics that study protein and the structure, including *proteomics* and structural bioinformatics. In this thesis, we focus on the protein structure comparison which is an important tool for the protein analysis.

1.2 Importance of protein structure analysis and comparison

Protein structure analysis covers tasks such as studying the properties of the structure, structure comparison, structure prediction, and interactions between proteins. While the purpose of the protein structure analysis is to obtain a new biological insight, two main applications of protein structure analysis are as follows:

1. Structure-based drug design. Protein structures lead to the functions. By study-

ing the relationship of the structures and the functions including the interaction between proteins through computational method (termed as *in silico* method), the cost and time for drug development can be largely reduced [4].

2. Study of evolution. Proteins are encoded by genetic codes in DNA (Deoxyribonucleic Acid) which is transcribed to RNA (Ribonucleic Acid). The genetic code is then translated into the sequence of amino acids which is the building block of a protein. Since the evolution can be studied through genetic codes, studying protein and the structures can support the study of evolution, particularly the early evolution of living things [5, 6].

In this thesis, we limit the scope of our study to the protein structure comparison. The importance of protein structure comparison is summarized in terms of the following reasons [7, 8]:

1. Function determination. In the study of protein, it has been understood that the function of the protein is based on the structure. Consequently, the function of an unknown protein can be inferred by comparing the structure against the database of known structures whose functions are already known.

2. Clustering. By clustering a number of proteins based on the structural similarities, we can analyze the properties of the similar structures of the same group.

3. Assessment of structure prediction. In structural biology, predicting the 3D geometrical structure of a protein from a sequence of amino acid using computational method is one important task. Structural comparison is used to evaluate the prediction algorithm by computing the similarity between the predicted structure and the ground-truth which is determined by experimental method.

1.3 Motivations and objectives

Protein structures obtained through experimental methods are stored in a public world wide repository called protein databank (PDB) [9]. To support the study of proteins, various protein structure classification schemes have also been proposed against the structures stored in PDB (further discussion on protein structure classification schemes is given in Chapter 2.2). The most comprehensive classification scheme for the protein structure is SCOP (Structural Classification of Proteins) [10, 11]. The SCOP database is constructed by protein experts through manual inspection and utilization of various

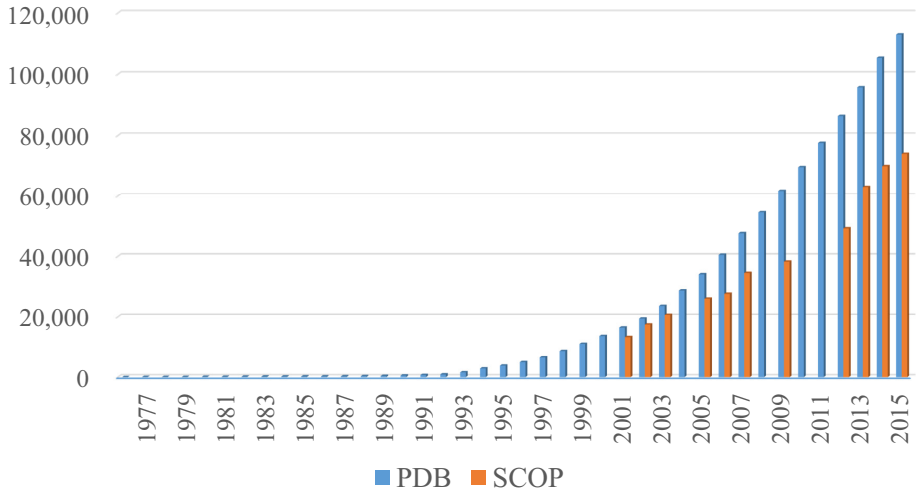

Figure 1.2: Growth of protein data stored in PDB and SCOP database.

analysis tools. As shown in Figure 1.2, due to the rapid advancement in the experimental methods for obtaining the protein structures, the growth of the protein structures stored in PDB increases exponentially, while the growth of the protein structures stored in SCOP database increases linearly. The number of the protein structures stored in PDB is more than $110,000$, while the number of the classified protein structures in SCOP database is about $73,996$ structures. This suggests that an automatic comparison and classification framework with high accuracy is needed.

Although protein structure comparison is very important, there is still no standard method for measuring the similarity between protein structures. The most common methods for comparing two protein structures are based on the structural alignment and the computation of root mean square deviation (RMSD) between the superposed structures. However, since structural alignment is a difficult problem, the alignment results are varied depending on the alignment algorithm. In this thesis, we address the issues on conventional method by the following approach:

1. We propose a new approach to protein structural comparison by casting the problem of 3D structural comparison into the comparison of multi-view molecular visualization images of the protein structures, termed as *view-based* method (Chapter 3). The validity of the proposed method is demonstrated through classification of seven classes proteins based on Astral SCOP [10, 12]. We also design and develop an online comparison system based on the proposed method.

2. We propose a method for combination of multiple distances by generalizing the concept from large margin nearest neighbor (Chapter 4). The validity of the proposed method is demonstrated through classification of 27 folds proteins of Ding Dubchak dataset [13, 14] and six classes of protein enzymes [15].

The objective of this research is to provide a robust method for comparing two protein structures. The validity of the proposed method is demonstrated through classification experiments. However, the ultimate goal of this research is to provide a general comparison tool for protein analysis, not limited only for classification of protein structures, but also for other purposes such as screening of structures in the pipeline of protein-protein interaction or for protein clustering.

1.4 Thesis organization

The rest of this thesis is organized as follows.

- Chapter 2 provides the overview of protein structure. Firstly, the composition of protein structures is provided. Secondly, several classification schemes of protein structures are described. Finally, conventional methods for protein structural comparison are reviewed.
- Chapter 3 discusses the proposed methods for protein structure comparison using multi-view molecular visualization images.
- Chapter 4 discusses the proposed methods for combination of multiple distance metrics for protein structure classification.
- Chapter 5 concludes the thesis by providing summaries and future works.
- Appendix A provides the list of protein structures used in the experimental section in Chapter 3.
- Appendix B provides the detailed technical documentation on the online comparison system based on the proposed method in Chapter 3, called View-based Protein Comparison (VPC)-system.

Chapter 2

Overview of Protein Structure Analysis

In this chapter, we first provide the review on the composition of protein structure, including how to obtain the 3D protein structures. Then, classification schemes of protein structure are reviewed. Finally, the conventional methods for protein structure comparison are described.

2.1 Protein structure composition

The building block of proteins is amino acid. The structure of one amino acid is shown in Figure 2.1. Amino acid contains amine group (NH_3^+), carboxylic acid group (COO^-), one hydrogen atom, and a residue group (side chain). In the center, there is a carbon atom which is called central alpha carbon (C_α) that connects each group and the hydrogen atom. Figure 2.2 lists 20 common amino acids available in nature including the three-letter and one-letter abbreviations for the amino acid naming. Amino acids are categorized as non polar if the side chain attracts water. Non polar amino acid, on the other hand, repels water. The property that disfavors water is also termed as *hydrophobic*. Amino acids that have either negative or positive charge of side chain are categorized as electrically charged amino acid. Amino acids with negative charged side chain are acidic, while amino acids with positive charged side chain are basic.

2.1.1 Determination of protein structure

The 3D coordinates of every atom of a protein molecule can be determined through experiments. It is still an open problem for prediction of the 3D protein structure

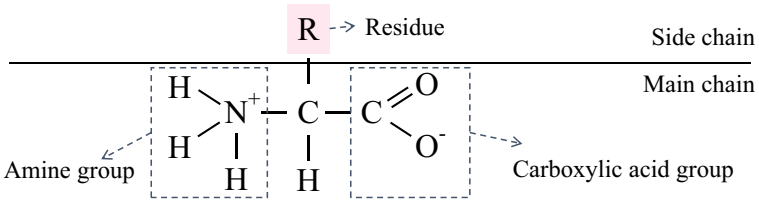

Figure 2.1: Structure of amino acid.

from amino acid sequence by using only computational method. In the following, the experimental methods to determine the protein structure using X-ray crystallography and nuclear magnetic resonance (NMR) spectroscopy are briefly described, as most structures stored in PDB are obtained from these methods.

In X-ray crystallography method, the protein has to be purified and crystalized. Then, X-rays' beams with specific wave-lengths are emitted to the crystal form of the protein. The diffraction pattern from the crystalized protein is then analyzed to generate electron density map which later be used to determine the location of the atoms. The accuracy of the obtained structure from the X-ray crystallography method is measured by resolution and *R-factor*. Resolution defines the level of detail of the diffraction pattern and the electron density map. The unit of length used is Ångström (Å), where 1 Å= 10^{-10} m. Resolution of 1 Å is high, while the resolution of 3 Å is considered low, where only the basic contours of the protein can be inferred. R-factor measures the quality of the obtained atomic model by comparing the model with the simulated diffraction pattern which is prepared beforehand. Smaller value of R-factor indicates better quality.

In NMR spectroscopy method, the protein has to be purified and in the form of liquid solution. Then, radio waves are emitted to the protein under a strong magnetic field condition. The conformation of the atoms is determined by analyzing the resonances. Since the protein is in the form of liquid solution, it is possible to study the structure of flexible protein using the NMR spectroscopy.

Besides X-ray crystallography and NMR spectroscopy, methods for determination of protein structures include electron microscopy, optical spectroscopic, vibrational spectroscopy, and electron spin resonance spectroscopy.

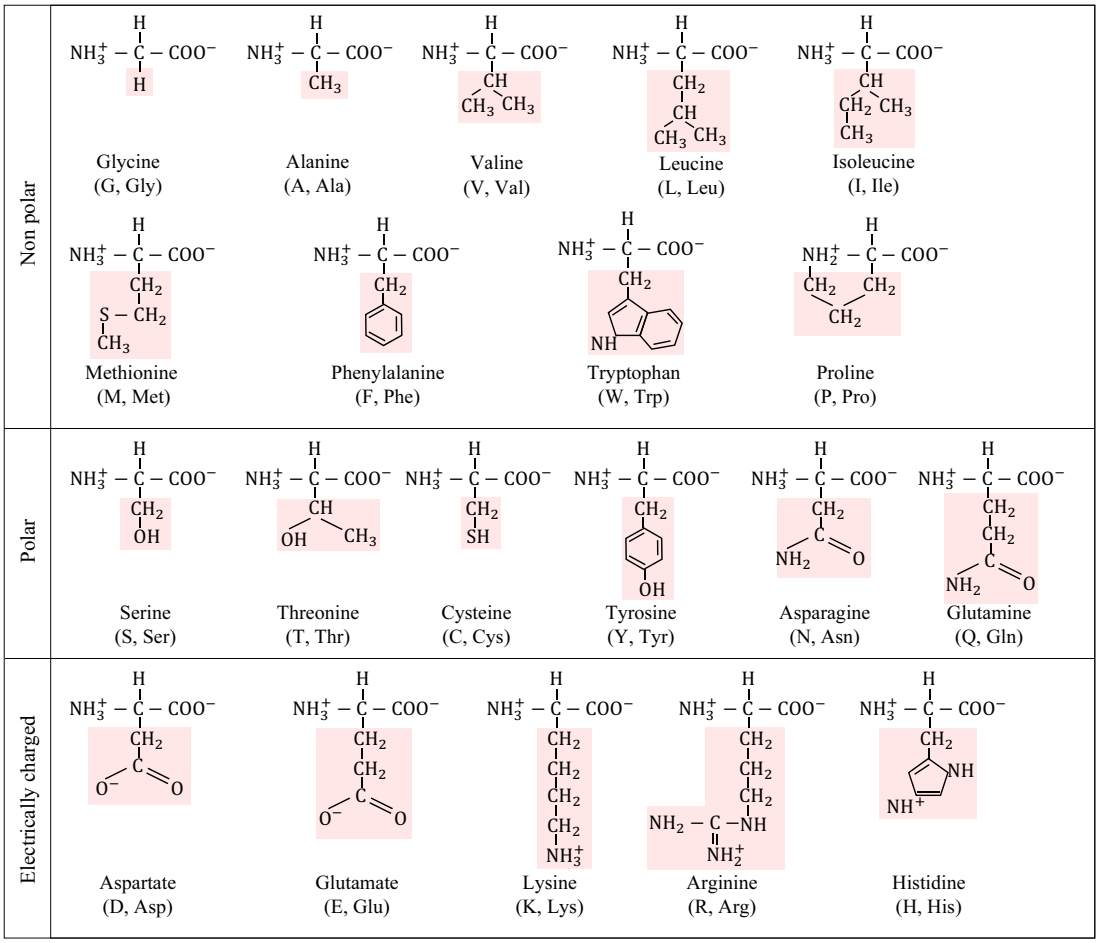

Figure 2.2: List of 20 common amino acids available in nature.

2.1.2 Hierarchical composition of protein structure

The primary structure of a protein is an ordered sequence of amino acids, where each of the amino acid is connected by *peptide bond* as shown in Figure 2.3. Peptide bond between two amino acids is formed by sharing the electron pairs in the amine group and the carboxylic acid group of both amino acids, in which a water molecule is released. From the bonding, the atoms connecting two C_α are in the same plane, forming torsion angles of ω, ϕ, and ψ. These angles are important properties to determine the *secondary structure* of the protein.

The secondary structure of a protein is common substructures which are formed due to the the interactions between the atoms, particularly the bonding of hydrogen atoms.

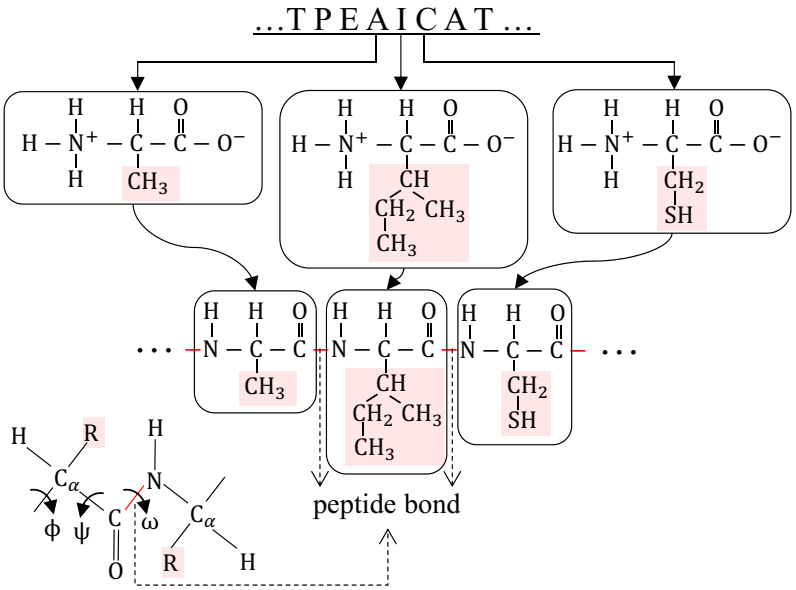

Figure 2.3: Peptide bonds between Alanine (A) and Isoleucine (I) and between Isoleucine (I) and Cysteine (C). The amino acid sequence shown in the figure belongs to the protein with PDB code 1crn.

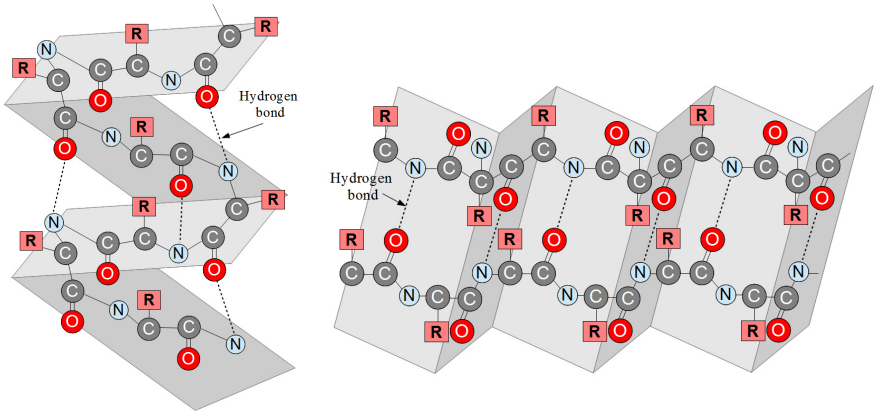

Figure 2.4: Secondary structures of protein. Left: α-helix; right: β-sheet.

Figure 2.4 shows the secondary structures of alpha helix (α) and beta strand/sheet (β). Substructures not in the categories of both α and β structures are categorized as random coils or turns. Based on the number of the residues and the torsion angles, alpha helix can be further be categorized into 3.6_{13}-helix (α-helix), 3_{10}-helix, and π-helix. The 3.6_{13}-helix has 3.6 residues per turn. The 3_{10}-helix and the π-helix have approximately

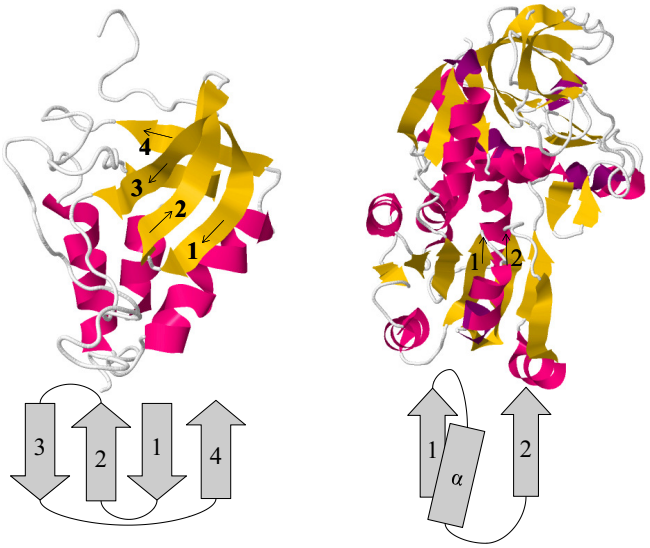

Figure 2.5: Example of common topologies of tertiary structure of protein. Left: antiparallel β-sheet; right: $\beta\alpha\beta$.

3 and 4.1 residues per turn, respectively. The *beta* structure can be categorized further into β-bridge (single β structure) and β-sheet (parallel or anti-parallel β structure).

The tertiary structure of a protein is determined based on how the secondary structures are connected. There are common motifs in the tertiary structure. For examples, antiparallel β-sheet and $\beta\alpha\beta$ motifs are two common motifs in the tertiary structure (Figure 2.5). The interaction of more than one tertiary structure of protein forms the quaternary structure. Figure 2.6 summarizes the four levels of protein structure compositions.

2.2 Protein structure classification

Recent advancement in molecular biology has enabled the collection of massive data of proteins. To take the full advantages of such abundant data, protein classification is required. The main purpose of the classification is to have meaningful categorical sets of proteins which constitute the same biological context. The protein classification scheme can be categorized into two approaches. The first approach is based on the structural information of the protein, while the second approach is based on the functionality [16].

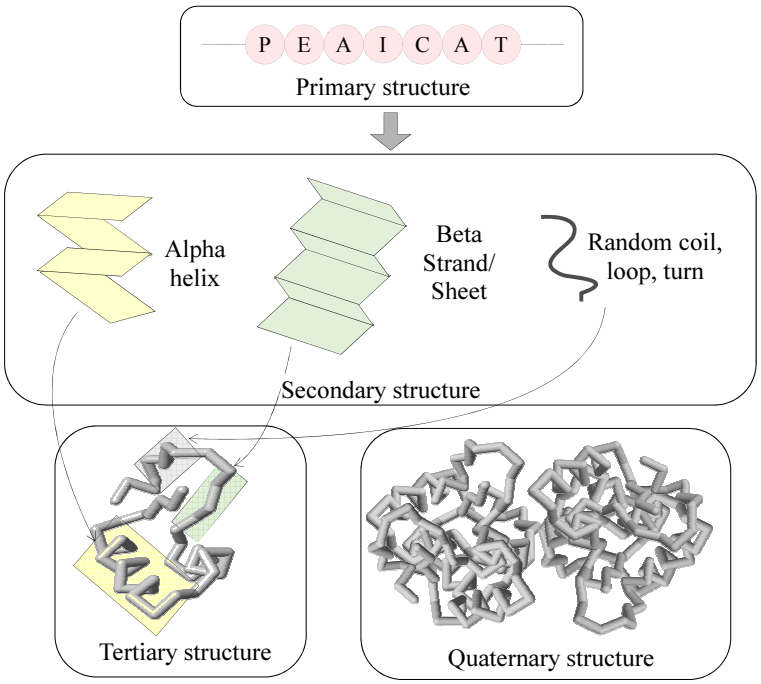

Figure 2.6: Four levels of protein structure compositions.

2.2.1 Structure-based classification

In structure-based classification, the classification scheme is based on the information of either primary, secondary, or tertiary structure of the protein. Popular classification schemes in this category include SCOP (Structural Classification of Proteins) [10, 11], CATH (Class Architecture Topology Homology) [17], FSSP (Fold classification based on Structure-Structure alignment of Proteins) [18], and DSSP (Dictionary of Secondary Structures of Proteins) [19].

SCOP provides a comprehensive classification scheme and description of known protein structures, where the structures are from the PDB. SCOP is constructed by manual inspection of protein experts with the help of some tools for the analysis. Therefore, SCOP is regarded as one of the most comprehensive and reliable protein structure classification scheme. Consequently, most classification algorithms are tested through SCOP classification scheme. In this thesis, we also used SCOP as the ground truth in the experiment. In SCOP, a protein is categorized based on a hierarchical classification scheme as shown in Figure 2.7. Prior to the classification, a protein is firstly segmented into its *domain*. A domain in a protein is defined as an area that is relatively inde-

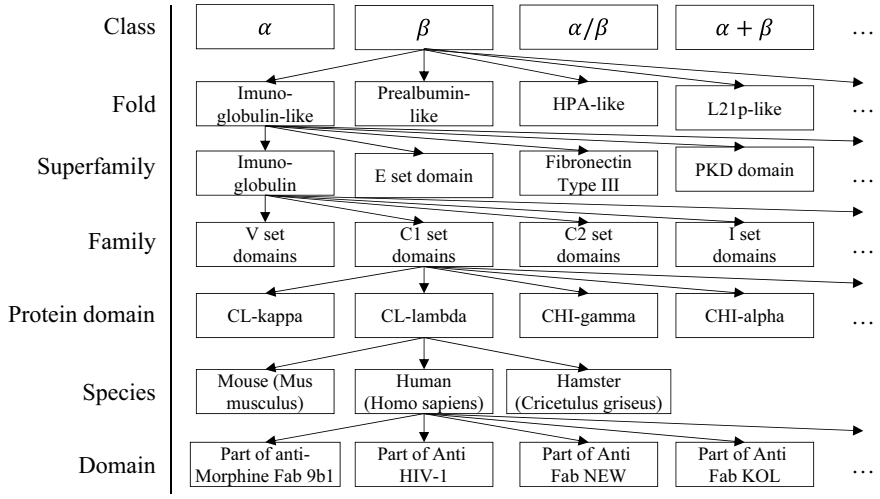

Figure 2.7: Hierarchical classification scheme of SCOP.

pendent (less interaction with the rest of the protein). Then, the classification starts by placing the domain into class level, which is based on the secondary composition. There are seven classes in SCOP with addition of four synthetic classes: α (contains mostly helical structure), β (contains mostly β structure, α/β (contains both α and β where β sheets exist in parallel), $\alpha + \beta$ (contains both α and β where β-sheets exist in anti-parallel direction), multi-domain (contains multiple domains), membrane (has membrane and surface structure), and small proteins (has few well-defined secondary structures). The next level of the hierarchy is the identification of fold, which is regarded as the most difficult stage [20]. Proteins are categorized into the same fold if they share similar topologies. The next hierarchical level is superfamily. Proteins that belong to the same superfamily have similar fold and perform similar functions, but low sequence identities. Proteins of the same family, which is one step further from superfamily, have an evolutionary relationship based on the primary structure similarity. Proteins with the same binding site are grouped into protein domain. Proteins are then categorized based on species. The bottom hierarchy in SCOP is basically the part of the segmented protein which is stored in PDB.

In CATH database, proteins are also categorized into hierarchical classification scheme which is shown in Figure 2.8. CATH is constructed by semi-automatic procedure where the intervention of human's analysis is not as much as in SCOP. At the top level of the hierarchy, a protein is categorized into class level. There are three major classes in CATH: mainly α, mainly β, and α-β which includes both α/β and $\alpha + \beta$.

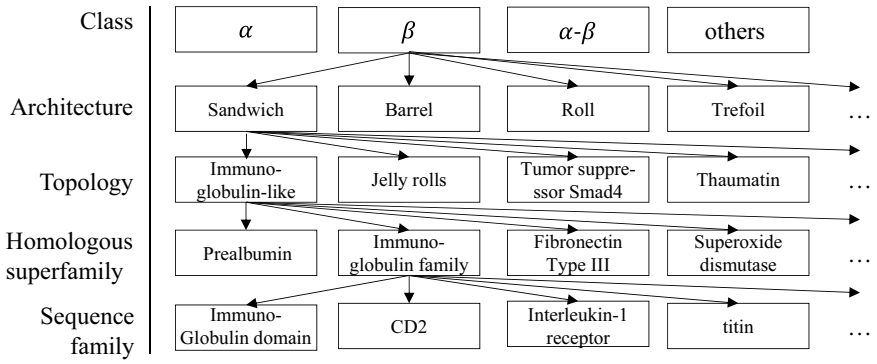

Figure 2.8: Hierarchical classification scheme of CATH.

These classes correspond to the α, β, α/β, and $\alpha+\beta$ classes in SCOP. One level below the class is called architecture. Proteins belong to the same architecture have roughly the same arrangement of secondary structure without consideration the connectivity. The next level of the hierarchy is topology. Topology in CATH corresponds to the fold level in SCOP. Finally, the last levels of hierarchy in CATH are homologous superfamily and family, which corresponds to the superfamily and family in SCOP. Just like SCOP, CATH is also widely used to benchmark protein structure comparison methods [21].

FSSP is constructed fully automatic by using DALI (Distance matrix ALIgnment) [22] against the proteins stored in PDB with criterion of having more than 30 residues and all pairwise sequence identities less than 25% (Sequence identities can be interpreted as the similarities of the amino acid sequence). By using these proteins as reference, all-against-all comparison and clustering are performed to generate the FSSP database.

DSSP contains the database of secondary structures of protein. DSSP is constructed fully automatic by using hydrogen bond estimation algorithm, which is a standard method for predicting secondary structure from the primary structure of a protein. There are eight types of outcome from DSSP: α-helix, β-bridge, β-sheet, 3_{10}-helix, π-helix, hydrogen bonded turn (helical turn), bend, and coil.

2.2.2 Function-based classification

Function-based classification does not take the structural information of the protein into account. Nevertheless, proteins with similar fold structures share the same functionality. Therefore, the distinction between function-based and structure-based approaches are not very obvious. In function-based classification, the classification scheme is based on the properties of the proteins in the experiment or other features that are com-

pletely independent of the protein structures. Most of the databases of function-based classification are constructed with low to medium automation. For example, Enzyme Commission (EC) hierarchical classification [15][1], firstly proposed in 1993, is one of the oldest and well-known classification schemes that categorize protein into six main classes based on the enzyme reaction mechanism.

2.3 Comparison between protein structures

Comparison between protein structures is an integral part of the protein structure analysis. The most common approach to comparison between protein structures is based on structural alignment. To avoid the difficulties of the alignment, different approaches apply geometrical feature extraction to the protein structure and represent it by descriptor such as graph or feature vector. The protein structures are then compared by computing the distance or similarity between the corresponding descriptors.

2.3.1 Alignment-based comparison

The principal of alignment-based method is by firstly applying alignment that considers the equivalence between pairs of amino acid residues. Next, a search is performed to determine the type of the geometrical transformation that minimizes the distance between the C_α [8]. Finally, the root mean square deviation (RMSD) between the superimposed C_α is computed as the dissimilarity measure. RMSD is written as

$$RMSD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\delta_i} \ , \tag{2.1}$$

where δ_i is the distance between ith aligned-pairs of the C_α of the superimposed protein structures and N is the length of the alignment.

Many methods of protein structural alignment have been proposed. Some of the widely used methods are DALI [22], CE (Combinatorial Extension) [23], FATCAT (Flexible structure AlignmenT by Chaining Aligned fragment pairs allowing Twists) [24], and TM-align (Template Modeling alignment) [25].

In DALI, the first step of the algorithm is the construction of a distance matrix containing the pairwise distances between the intra C_α for each protein structure. Then, overlapped sub-matrices of six residues, termed as *hexapeptide fragments*, are generated. Next, DALI searches for the most similar and compatible pairs of the hexapeptide

[1] As of 2015/11/26, EC database is accessible at http://www.expasy.ch/enzyme/

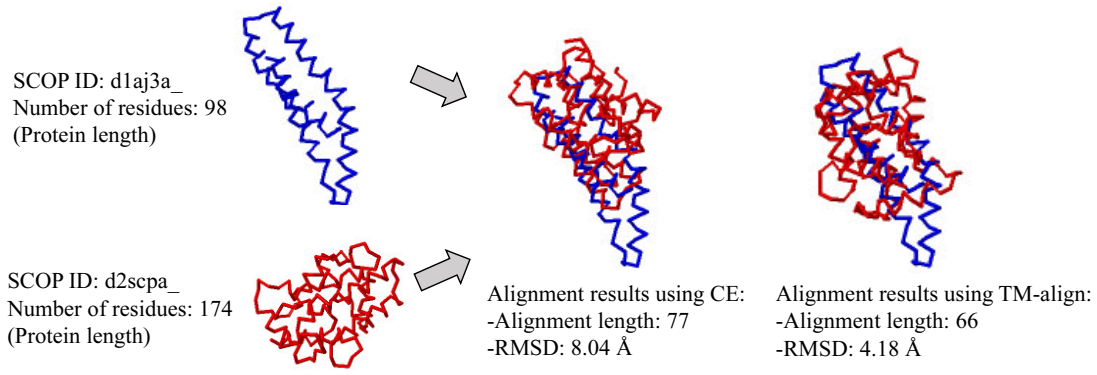

Figure 2.9: Problems of alignment-based methods: different alignment algorithms may output different alignment results.

fragments. These similar pairs are called *contact patterns*. From the set of the contact patterns pairs, a consistent large set of contact patterns is generated by using Monte Carlo algorithm, while at the same time minimizing the distance of the C_α pairs.

In CE, the protein structure is also segmented into fragments. A pair of similar fragments is called aligned fragment pair (AFP), in which one fragment contains eight residues. AFPs are detected based on the threshold of RMSD of the C_α. After set of AFPs are collected, CE searches for the longest continuous path of the AFPs by using dynamic programming.

FATCAT has the similar procedure with CE. AFPs are firstly collected. Then, by using dynamic programming AFPs are connected. The difference between FATCAT and CE is that in FATCAT, the alignment restriction is more flexible where twists are allowed in connecting two AFPs.

In TM-align, the first step of the algorithm is by applying an initial alignment. There are three types of initial alignments used in TM-align. The first type is through the detection and alignment of the secondary structures by using dynamic programming. The second type is based on the *gapless matching* of the structures, based on the threading algorithm[1]. The third type is also based on the dynamic programming but with a different penalty value. After the initial alignment, the next step is done by applying iterative heuristic algorithm to refine the alignment result.

One problem with alignment-based methods is the difficulty on determining a single optimal alignment in terms of functional similarity or tertiary structure similar-

[1]Although threading is commonly used for protein structure prediction from the amino acid sequence, threading is also equivalent to structure comparison [26].

ity [27, 28]. Consequently, different alignment algorithms produce different results. For example, Figure 2.9 shows the alignment results for protein with SCOP ID d1aj3a_ and d2scpa_ using CE and TM-align. From the results, we can see that the alignments produced quite different results. The difficulty of alignment increases when the proteins sequence identity is 20% or below because the structural differences become large [29]. Another problem arises from the use of RMSD values. RMSD depends on the length of the alignment, but a greater number of aligned positions does not always produce a smaller RMSD [24, 25]. This suggests that RMSD is too blunt to be used as dissimilarity measure. As a result, for classification and retrieval tasks, various scoring functions have also been proposed to complement the RMSD. For example, in CE, Z-Score statistics is computed by evaluating the probability of finding alignment path of the same with gap either smaller or the same in a representative dataset of proteins [23]. FATCAT also has its own score, called as FATCAT score. TM-align also provides a scoring function called TM-score [30], which uses a weight based on the length of the aligned structures.

2.3.2 Descriptor-based comparison

To avoid the alignment difficulties, other approaches use a global representation of the protein structures by applying feature extraction to the 3D geometrical structures. In the following, some of the notable protein descriptors are described.

In [31], a protein descriptor called Gauss Integral Tuning (GIT) was proposed. In GIT, the backbone structure of a protein is regarded as an oriented open curve in 3D space. Then, based on the knot theory, a series of Gauss integrals for the writhe[1] and average crossing are computed over the curve, and the protein structure is finally represented as a 31-dimensional feature vector. GIT has been used for fast clustering of protein structures [32].

In [33], principal component correlation (PCC) method was proposed. In PCC method, a protein structure is represented by a symmetric interaction matrix containing parameters of the relation between secondary structure elements. The similarity between the protein structures is defined by correlating the principal components of the matrices.

In [34, 35], tableau-based representation for protein structures was proposed. In this method, a protein structure is described as a tableau containing the encoded orientation of the secondary structures. The similarity of two protein structures are computed by

[1]Writhe is defined as the total number of negative crossing subtracted from the total number of positive crossing.

searching the most similar *subtableaux* using either quadratic integer programming, integer linear programming, or dynamic programming for speeding up the process.

In [36], 3D Zernike descriptor was proposed for representing protein structures. However, instead of extracting the features of the 3D backbone structure, this approach extracts 3D Zernike of the voxelized 3D geometrical surface of the protein structure. Similar to [36], the method of [37] applied feature extraction based on the wavelet to the voxelized 3D geometrical surface of the protein structure.

In the above methods, we note that the protein structural comparison no longer requires alignment techniques. Instead, the similarities or dissimilarities between the protein structures are defined by the comparison of the descriptors of the corresponding protein structures.

Chapter 3

Protein Structure Comparison Using Multi-view Visualization Images

As mentioned in the previous chapter, alignment based method has difficulties on comparison of two different structures. In this chapter, we propose a view-based approach for protein structure comparison in order to address the problems of conventional methods. The unique feature of our proposed method is the use of a set of 2D multi-views of 3D molecular visualization. This chapter is organized as follows. The background of the proposed method is discussed in Chapter 3.1. The detail of the proposed method is then described in Chapter 3.2. The experimental results are presented in Chapter 3.3. Further discussion on the proposed method is provided in Chapter 3.4. Then an online implementation of the proposed method is described in Chapter 3.5. Finally, the summary is given in Chapter 3.6.

3.1 Background

There are a number of 3D molecular visualization softwares which can be used for visualizing the 3D molecular structure of a protein from a set of the 3D atom coordinates of the structure [38]. The molecular visualization softwares are commonly used for manual inspection of protein structure to supplement automatic analysis tools and for educational purposes [38, 39]. In practice, 3D structure of protein is inspected manually after projection from multiple viewpoints onto the 2D plane of a computer screen using various types of structural representation. This suggests that protein visualizations

contain distinctive information for protein analysis, which leads us to propose a new approach that employs a set of multi-view images of the 3D visualization. This also points to the additional advantage that particular characteristics of a protein can be more readily extracted from different types of protein visualizations. As a result, the comparison between protein structures is casted into the comparison of the two sets of the multi-view images of them.

In computer vision field, one of widely used methods for the comparison of image sets is based on the mutual subspace method (MSM) [40]. In MSM, two sets of images to be compared are represented by subspaces which are generated by applying principal component analysis (PCA). Then, the similarity between the sets of the images is defined by the canonical angles between the two corresponding subspaces. MSM and its extensions [41, 42] have been successfully applied to various 3D object recognition tasks, such as facial recognition [43, 44, 45, 46], hand shape recognition [47, 48], and general 3D object recognition including identification of 100 apples with 99% recognition rate [49]. The simplicity and high performance of MSM based method for classification of complicated and similar 3D shapes also motivate us to adopt the subspace method.

Motivated from those two factors above, we exploit multi-view images of protein visualization for protein structure comparison. The proposed similarity measurement method takes the advantage of the multi-view images of the protein visualization and the subspace representation. To sum up, the advantages of the proposed method are as follows:

1. Precise alignment is not needed, especially for comparing very different structures.

2. By using 3D molecular visualization software, various types of protein visualizations can be used to emphasize particular properties of the protein structure.

3. The proposed method is a subspace-based method, so that it is straightforward to extend the capability of the proposed method by adapting any subspace learning methods, where in this thesis we apply Grassmann discriminant analysis (GDA) [50] in the classification framework.

In brief, the contributions of this work are as follows:

1. Introduction of GDA as a subspace learning method to the protein-structure classification framework. GDA can also be regarded as a feature extraction method that improves the discriminative power among the subspaces corresponding to the protein structures.

2. Introduction of LBP-based image-feature extraction methods [51, 52, 53] during analysis of the protein visualization images.

3. Comprehensive classification of seven classes of protein structure based on the Structural Classification of Proteins (SCOP) scheme [10].

4. Design and development of an online protein structure comparison system based on our view-based method; namely, the View-based Protein Comparison (VPC) system.

3.2 Proposed method

Figure 3.1 shows the basic idea of the proposed method. The key point is that instead of using the 3D geometrical information directly, a set of multi-view visualization images is generated by rotating the protein structures by using 3D molecular visualization software. Using the software, we can also synthesize different types of visualizations that emphasize different properties. After collecting a set of multi-view images, a feature vector is then extracted from each image, because the generated protein images can contain position and pixel variation while also having a complex visualization structure. Local binary pattern (LBP)-based [51, 52, 53] image-feature extraction methods were applied to the set of images, as LBPs can encode the micro-patterns of an image. A subspace is then generated by applying PCA to the set of feature vectors. Finally, the similarity between protein structures is measured by the canonical angles θ_i between the corresponding subspaces.

The problem of comparing protein structures becomes that of comparing sets of multi-view images. In this sense, we can regard this method as *view-based* method. In the following we describe the basic idea of using subspace representation for image set.

With the advancement of computer graphics, we can visualize the structure of proteins by using various 3D molecular graphics softwares [38]. For example, Figure 3.2 shows the backbone visualization of three small proteins and one α-protein generated using Jmol [54]. Proteins d1axha_ and d1viba_ belong to small protein class but different fold; protein d1axha_ and d1omca_ belong to the same superfamily of *omega toxin-like* [10].

To capture the whole structural view of a protein, firstly, a number of 2D projected images are synthesized by collecting the visualization images from multiple viewpoints, through rotation of the 3D structure. Then, subspace representation is used to model the pattern distribution of the image set. Let $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{n}$ be a set of feature vectors

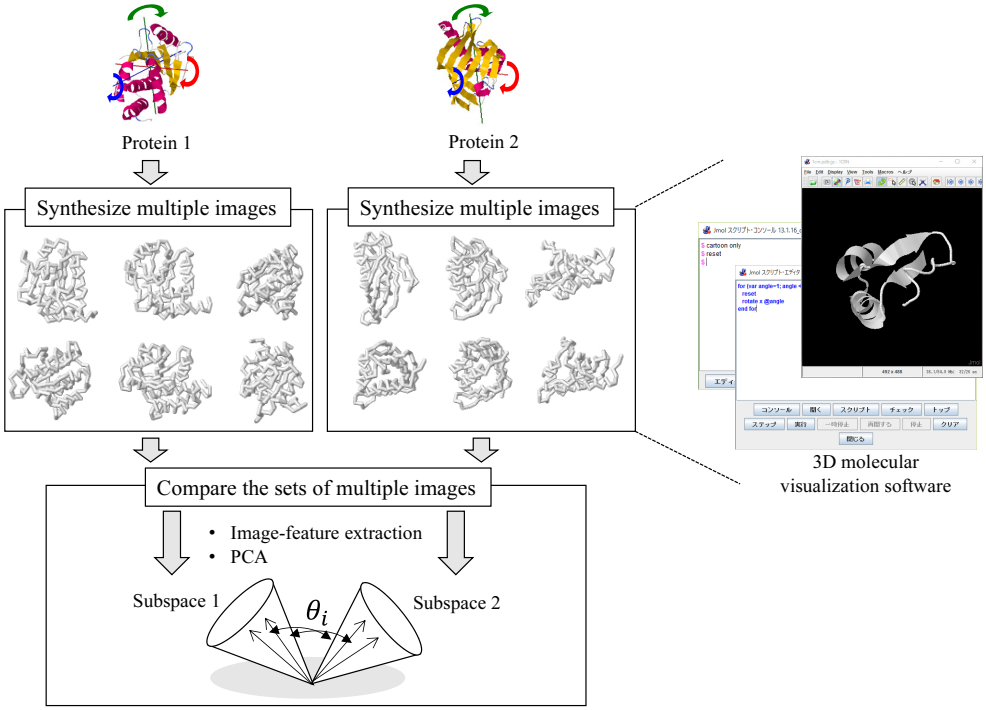

Figure 3.1: Basic idea of the proposed method.

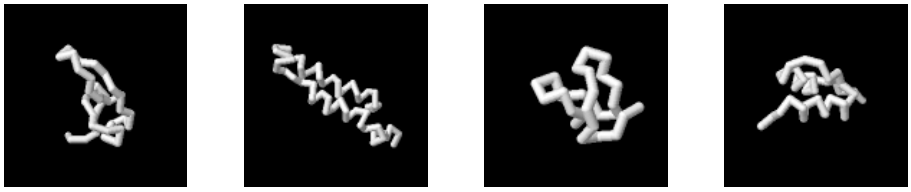

Figure 3.2: Example of backbone visualization. From left to right: backbone visualization of small proteins d1axha_, d1viba_, and d1omca_, and α-protein d1kcfa1, respectively.

of an image set from protein p, where $\mathbf{x}_i \in \mathbb{R}^f$, f is the dimensionality of one feature vector, and n is the number of the images. The autocorrelation matrix that corresponds to $\mathbf{X}$ can be computed as $\mathbf{A} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top$. A set of orthogonal basis vectors of the N-dimensional subspace $\mathcal{P}$ that represents $\mathbf{X}$ is obtained by computing the eigenvectors $[\phi_1, \ldots, \phi_N] \in \mathbb{R}^{f \times N}$ corresponding to the N highest eigenvalues of $\mathbf{A}$. Subspace $\mathcal{P}$ then contains the space for the overall possible translation and rotation of the protein shape. For example, Figure 3.3 shows the visualization of the first 30 basis vectors of

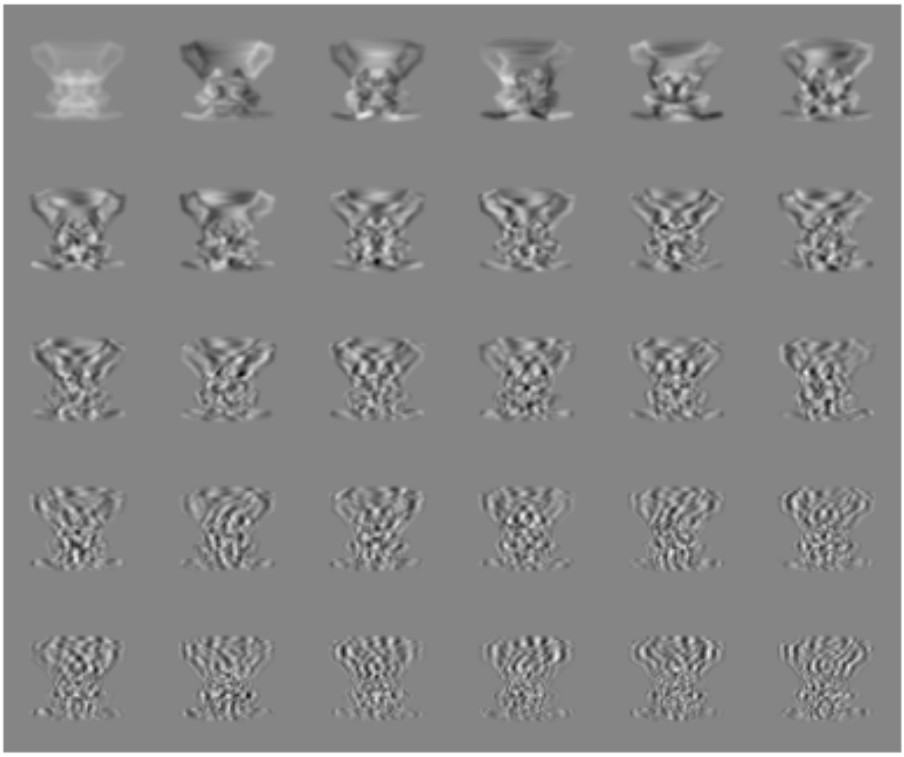

Figure 3.3: The visualization of the first 30 basis vectors of the corresponding subspace for d1axha_.

the corresponding subspace for d1axha_, where 360 images (64×64 pixels), synthesized through 359 times rotation around Y-axis of the viewing plane coordinate (360 view points around Y-axis), were used.

Unlike alignment based methods which search for a single most optimal alignment, comparing two subspaces essentially means that we search for multiple similarities in the space of the possible pose and rotation of the protein shapes within the two corresponding subspaces simultaneously. Let $\mathcal{Q}$ be another M-dimensional subspace generated from protein q. The similarity between subspaces $\mathcal{Q}$ and $\mathcal{P}$ is defined using the cosines of the canonical angles $\theta_{i=1,\ldots,\min(M,N)}$ $(\cos\theta_1 > \cdots > \cos\theta_{\min(M,N)})$ between them. The objective function to obtain the first canonical angle [55] is written as

$$\cos\theta_1 = \max_{\mathbf{u}\in\mathcal{Q}} \max_{\mathbf{v}\in\mathcal{P}} \mathbf{u}^\top \mathbf{v}, \tag{3.1}$$

where $\mathbf{u}$ and $\mathbf{v}$ are the f-dimensional unit vectors (canonical vectors) of subspaces $\mathcal{P}$ and $\mathcal{Q}$ respectively, that form the smallest angle. The next i-th canonical angles are

defined similarly as in (3.1), such that $\mathbf{u}_i^\top \mathbf{u}_i = \mathbf{v}_i^\top \mathbf{v}_i = 1$ and $\mathbf{u}_i^\top \mathbf{u}_j = \mathbf{v}_i^\top \mathbf{v}_j = 0$ for $i \neq j$. The practical solution to obtain the set of the canonical angles is by computing the singular values of $\mathbf{U}^\top \mathbf{V}$, where $\mathbf{U} = [\phi_1, \ldots, \phi_N]$ and $\mathbf{V} = [\psi_1, \ldots, \psi_M]$ are the sets of orthogonal basis vectors of subspaces $\mathcal{P}$ and $\mathcal{Q}$, respectively.

A pair of canonical vectors $\hat{\mathbf{u}}_i$ and $\hat{\mathbf{v}}_i$ that form canonical angle θ_i can be obtained as follows:

$$\hat{\mathbf{u}}_i = \mathbf{U}\mathbf{y}_i, \quad \hat{\mathbf{v}}_i = \mathbf{V}\mathbf{z}_i, \tag{3.2}$$

where $\mathbf{y}_i$ and $\mathbf{z}_i$ are, respectively, the left and right singular vectors of $\mathbf{U}^\top \mathbf{V}$.

Figures 3.4(a), 3.4(b), and 3.4(c) show the first five canonical vectors and the similarity values when comparing 100-dimensional corresponding subspaces of d1axha_ with d1viba_, d1omca_, and d1kcfa1, respectively. The first canonical angle, $cos^2\theta_1$, denotes the highest similarity value of the two proteins. The second canonical angle, $cos^2\theta_2$, denotes the second highest similarity value, and so on. The obtained similarity values based on the first canonical angle were 0.7014, 0.8281, and 0.5654, respectively. In comparison with TM-align, the average TM-scores were 0.2048, 0.4427, and 0.2005, respectively. From these, we can see that the similarity values between the protein structures can be captured through the subspaces representation, similar to that of the alignment based method. In addition to that, depending on applications, we can use multiple similarities instead of just a single best similarity. In the case of application for roughly classifying very different protein structures as preprocessing of more detail analysis, using the average of the similarities may produce better results because we can capture the overall similar sub-structures instead of just the most similar sub-structure.

The rest of this chapter describes the detail of the basic idea. We first discuss the usage of multi-view visualization images and how to generate them in Chapter 3.2.1. Then, we provide the overview of the image feature extraction methods used in our work in Chapter 3.2.2. We discuss how the comparison is conducted further with adaptation of Grassmann discriminant analysis (GDA) for classification task in Chapter 3.2.3. Finally the flow of the classification framework is presented in Chapter 3.2.4.

3.2.1 Generation of image set

In this research, we use Jmol [54] to generate the set of the visualization images. Through the 3D molecular visualization software such as Jmol, it is possible to view different type of protein visualizations which emphasize particular properties of the protein structure. Some of the protein visualizations from Jmol are shown in Figure 3.5. Backbone visualization displays the backbone structure of the protein, which

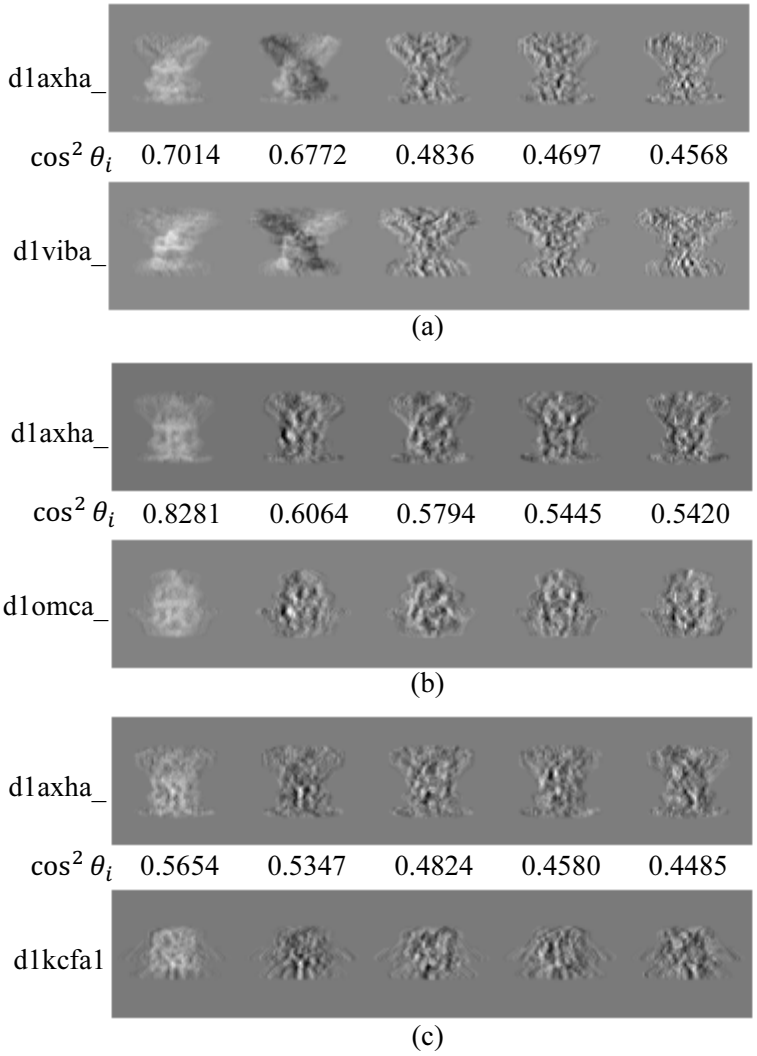

Figure 3.4: The visualization of the first five canonical vectors and the similarity values when comparing: (a) d1axha_ with d1viba_; (b) d1axha_ with d1omca_; (c) d1axha_ with d1kcfa1.

is the trace of the C_α atom. Ribbon visualization displays the backbone structure in a smooth ribbon shape, where the helical shape of the α-helix structure can be clearly visualized. Rocket visualization contains information of secondary structure of the protein, where it displays the secondary structure of α-helices and β-sheets as directed cylinders and arrows, respectively, while the random coils are visualized as strings. Cartoon visualization displays α-helices as ribbons, β-sheets as directed ribbons, and random coils as strings. Since each protein visualization has different characteristics,

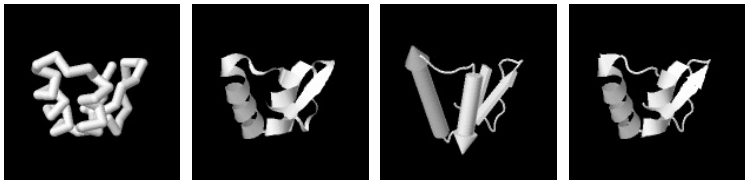

Figure 3.5: Visualization of protein structure with PDB code 1crn generated using Jmol [54]. From left to right: backbone, ribbon, rocket, cartoon.

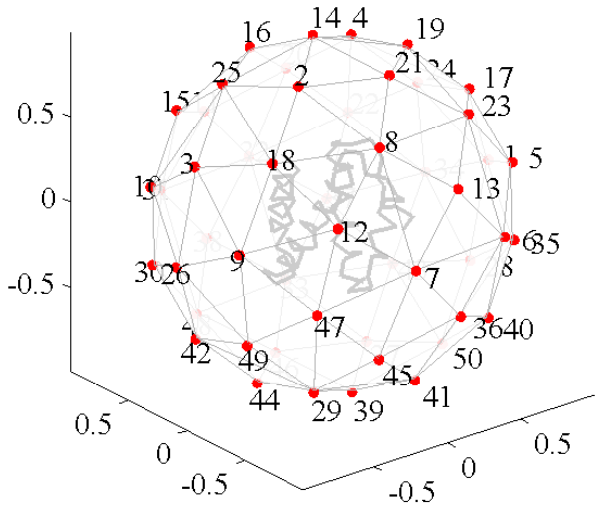

Figure 3.6: Illustration of 50 uniform viewpoints for a protein structure (50 uniform rotations). Each point on the unit sphere is the viewpoint from which the 2D visualization is synthesized.

we use multiple visualizations to build more elaborate protein descriptors.

We used simple rotational scheme (only around Y viewing axis) in describing the basic idea of the proposed method to simplify the interpretation of the visualized basis vectors and the canonical vectors. However, to capture the overall structure of a protein, multiple view protein visualization images are synthesized by uniformly rotating the protein structure around its central X, Y, and Z viewing axes. The procedure to

Figure 3.7: 50 uniform rotations of a cartoon visualization of protein ID d1a0pa1.

generate the uniform rotation angles can be regarded as the problem of producing a uniform distribution of points on the surface of a sphere, where each point is the viewpoint for the protein visualization, i.e., the position of the *virtual camera* that captures the visualization image, as illustrated in Figure 3.6. In the implementation, the protein structure is rotated according to the camera position. Note that here the

term of *the number of viewpoints* and *the number of rotations* are used interchangeably. Figure 3.7 shows an image set of cartoon visualizations generated from 50 uniform rotations (50 uniform view-points).

There are a number of methods for uniformly distributing a set of points on the surface of a sphere [56, 57, 58, 59]. One of the simplest approaches to approximate the uniform distribution of points on the surface of a sphere is by using the generalization of the Thomson problem. The Thomson problem originally aims for finding a configuration of set of electrons on the surface of a unit sphere such that the electrostatic potential energy is minimum [58, 59].

In the following, the generalization of the Thomson problem to find the configuration of uniformly distributed points on the surface of a sphere is briefly described. Let $\{\mathbf{x}\}_{i=1}^{N} \in \mathbb{R}^3$ be the set of N points to be distributed uniformly on the surface of a 3D sphere. The distances between all points are small if they are uniformly distributed. Thus, the objective function is written as follows [58]:

$$\begin{aligned} &\min \sum_{i=1}^{N} \sum_{j<i} \frac{1}{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}, \\ &\text{subject to } \mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^3, \|\mathbf{x}_i\| = \|\mathbf{x}_j\| = 1, \text{ and } 1 \leq i \leq N. \end{aligned} \tag{3.3}$$

To simplify the optimization, the constraints in Eq. (3.3) are removed by transforming the coordinate system from Cartesian into that of the spherical, where a point is composed of the elevation angle θ and the Azimuth angle ϕ. Points on spherical coordinate system (θ, ϕ) are transformed back to $\mathbf{x} = [x, y, z]^\top \in \mathbb{R}^3$ as follows [59]:

$$\begin{aligned} x &= \cos\theta \sin\phi, \\ y &= \cos\theta \cos\phi, \\ z &= \sin\theta. \end{aligned} \tag{3.4}$$

By substituting $\mathbf{x}$ with its elements x, y, z, function to be minimized in Eq. (3.3) is rewritten by

$$\min \sum_{i=1}^{N} \sum_{j<i} \frac{1}{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}. \tag{3.5}$$

Finally, with substitution using Eq. (3.4), the objective function becomes

$$\min \sum_{i=1}^{N} \sum_{j<i} \frac{1}{2\left[\cos\theta_j \cos\theta_i \cos(\phi_j - \phi_i) + \sin\theta_j \sin\theta_i - 1\right]}. \tag{3.6}$$

Equation (3.6) can be solved by using unconstrained nonlinear optimization methods. Here, we used the trust-region method [60] from the Matlab optimization toolbox.

3.2.2 Feature extraction

Instead of directly using the vectorized raw pixel values of the 2D protein visualization image as a feature vector, we can employ feature extraction methods to improve the robustness to the position and size variation of the protein. In the following, we briefly provide the overview of two image feature extraction methods, namely, local binary pattern (LBP) based method [51, 52, 53] and higher-order local autocorrelation (HLAC) [61], which were used in the experiments.

3.2.2.1 LBP-based feature extraction

The underlying concept of LBP is to use joint distribution of the pixels of local textural patterns. The LBP [51] of an image is defined as the binary code of the local texture found by thresholding a neighborhood of pixels around a center pixel as follows:

$$\begin{aligned} LBP(x_c, y_c) &= \sum_{p=1}^{P} s(I_p - I_c) 2^{p-1}, \\ s(x) &= \begin{cases} 1 & \text{if } x \leq 0, \\ 0 & \text{otherwise,} \end{cases} \end{aligned} \tag{3.7}$$

where (x_c, y_c) is the position of the center pixel, and I_p and I_c are the grayscale values of the neighboring pixel and the center pixel, respectively. P is the number of pixels in the neighborhood. If $P = 8$, the number of the LBP patterns is 256 (2^P). An image I is then encoded by a histogram of the LBP codes that contains the micro-patterns of the image. The flow of LBP computation is shown in Figure 3.8.

To derive a general formulation of LBP, a radius r is introduced as follows:

$$\begin{aligned} x_p &= x_c + r\cos(2\pi p/P), \\ y_p &= x_c + r\sin(2\pi p/P), \end{aligned} \tag{3.8}$$

where x_p and y_p are the coordinates of I_p in Eq.(3.7) and r is the radius. Figure 3.9 illustrates the effect of different r.

Due to the effectiveness and simplicity of this method, many LBP extensions have been proposed and used in various applications, such as face recognition [52, 62, 63] and

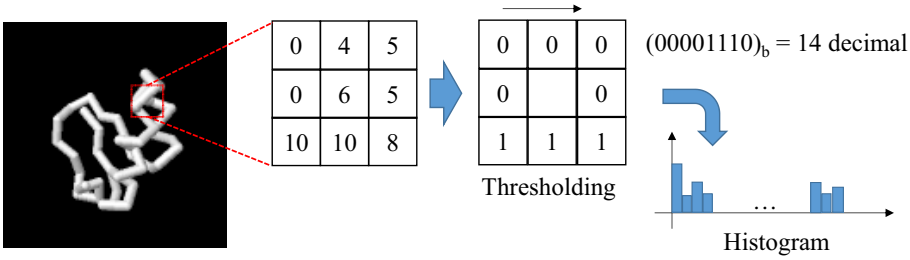

Figure 3.8: Flow of computing LBP histogram from an image.

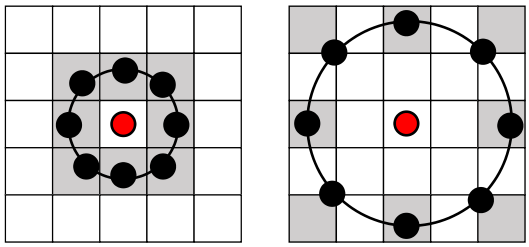

Figure 3.9: Neighborhood pixels affected by different radius r. Left figure: $r = 1$; right figure: $r = 2$.

facial expression analysis [64, 65], human detection [66, 67], and texture analysis [68] including medical image analysis [69] and bio-cell classification [53, 70]. In the following we reviewed some of LBP variants that we used for feature extraction of the protein visualization images.

Uniform LBP In uniform LBP, written as LBP^{u2}, there are two types of LBP code, namely, uniform pattern and non-uniform pattern [52]. LBP code containing at most two bitwise transitions is recognized as uniform pattern. For example, LBP codes 00000000 (no transition), 11111111 (no transition), 00110000 (2 transitions), and 00001111 (1 transition) are uniform. LBP codes 01001111 (3 transitions), 11001100 (3 transitions), 10101111 (4 transitions) are non-uniform. All LBP codes with non-uniform pattern are assigned to one LBP code, resulting in a lower number of LBP codes. For example, the number of the LBP patterns becomes 59 when $P = 8$. The motivations behind the proposal of uniform LBP are two folds [52]. The first is because through observation, most LBPs of texture images were found of containing uniform patterns up to 90% when $P = 8$. The second motivation is for robustness against noise.

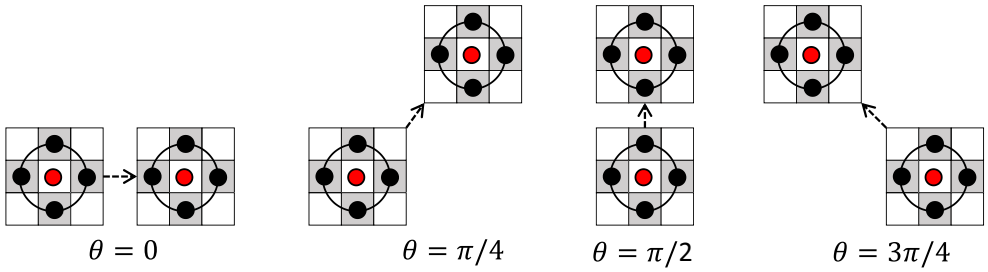

Figure 3.10: Configurations of CoALBP for four neighborhood pixels.

Rotational invariant LBP Invariance of rotation was introduced to LBP by applying rotation invariant mapping. In practice, the mapping is done through circular bitwise right rotation of the LBP code into its minimum value. For example, LBP codes 10000010, 10100000, 00001010 are mapped to the same minimum LBP code 00000101. Rotational invariant LBP is commonly written as LBP^{ri}. Combination of uniform LBP and rotational invariant LBP is termed uniform rotation invariant LBP, written as LBP^{riu2} [52].

Co-occurrence adjacent LBP In [70], an extension of LBP called co-occurrence adjacent LBP (CoALBP) was proposed. CoALBP encodes adjacent LBP (a pair of LBPs) into one binary code to consider their spatial relations. In relation to Eq.(3.7), CoALBP is written by

$$CoALBP(x_c, y_c; \mathbf{\Delta s}) = (LBP(x_c, y_c), LBP(x_c + \Delta s_1, y_c + \Delta s_2)), \tag{3.9}$$

where $\mathbf{\Delta s} = [s_1, s_2]^\top$ is a displacement vector with $s_1 = s\cos\theta$ and $s_2 = s\sin\theta$; s is the interval between two LBPs and $\theta = 0, \pi/4, \pi/2, 3\pi/4$. Figure 3.10 shows the configurations of CoALBP for four neighborhood pixels ($P = 4$). Due to this configuration, the number of possible LBP patterns of an image becomes $2^{2P} \times 4$.

In [53], rotational invariant of CoALBP, called rotation invariant co-occurrence LBP (RIC-LBP), was proposed. RIC-LBP searches for the rotation equivalent LBP pairs by considering cases where LBP pairs of $\theta = 0, \pi/4, \pi/2, 3\pi/4$ and LBP pairs which are rotated by 180 degrees have rotation equivalence. In RIC-LBP, the number of possible LBP patterns is reduced to $2^P(2^P + 1)/2$.

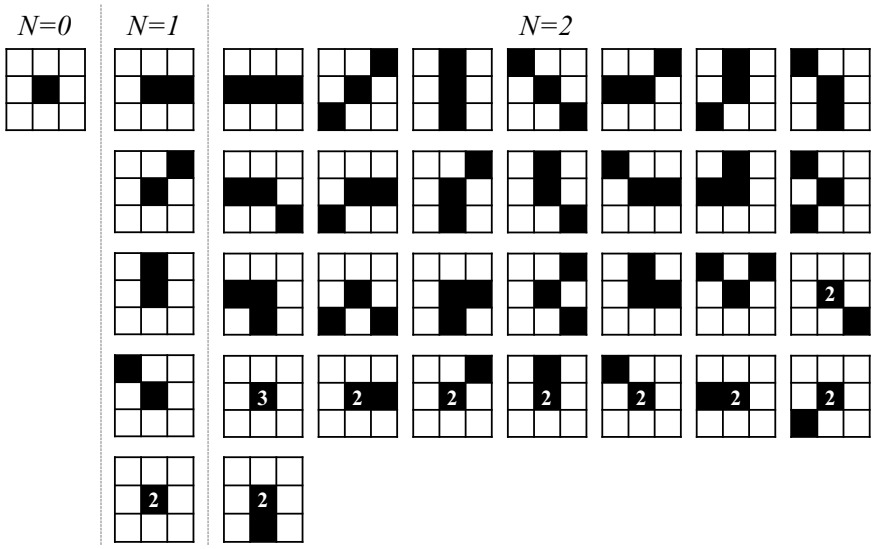

Figure 3.11: The HLAC's 35 mask patterns.

3.2.2.2 HLAC feature extraction

HLAC [61] encodes images as N-order autocorrelations of pixel values:

$$HLAC(I) = \sum_{r \in I} I(r)I(r + a_1) \dots I(r + a_N), \tag{3.10}$$

where r is the pixel position in image I and a_i is a pixel displacement with $(a_{ix}, a_{iy}) \in \{\pm\Delta, 0\}$. The order of N is limited to two ($N \in \{0, 1, 2\}$). Duplicate configurations of $r, r + a_1, \dots, r + a_N$ are removed so that the total number of local mask patterns is reduced to 35 (Figure 3.11), which results in a 35-dimensional HLAC feature vector. HLAC has the property of position invariance, which is useful for feature extraction from protein visualization images. To capture both the local and overall global structure, multiple of displacement range Δr, which are equivalent to using multiple image scales, are used (Figure 3.12). The details on the parameters used are described in the Section of Experiment.

3.2.3 Comparing two sets of protein images

In Chapter 3.2, we described how to compute the canonical angles between two subspaces corresponding to the protein structures, where the subspaces are generated by applying principal component analysis to the set of the feature vectors. Figure 3.13 il-

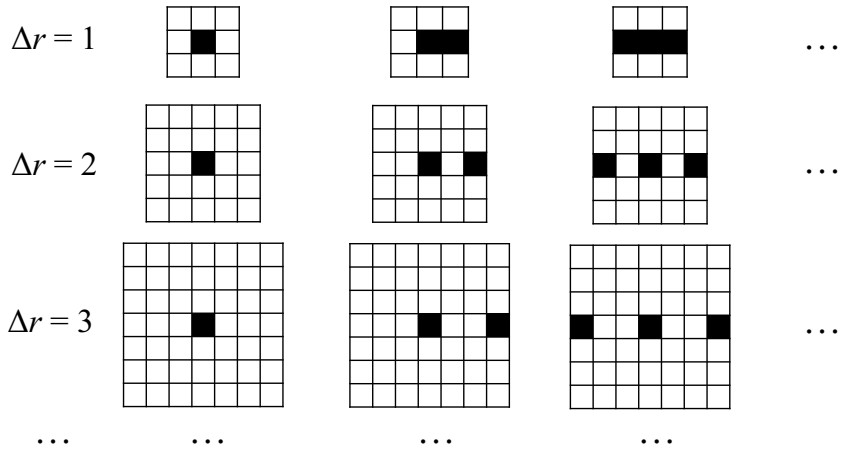

Figure 3.12: Example of multiple displacement ranges in HLAC.

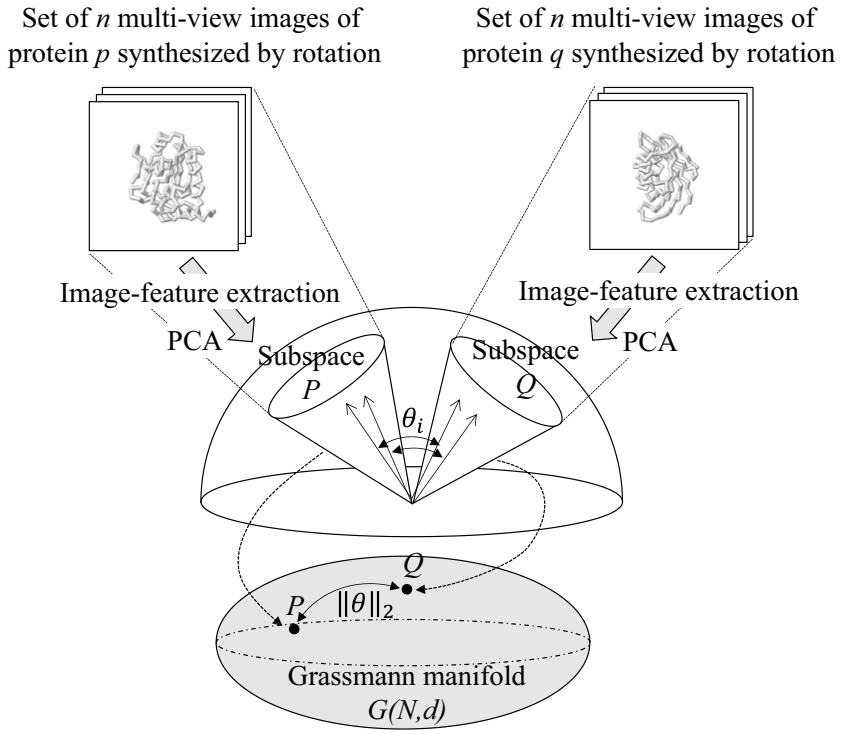

Figure 3.13: Protein structure similarity computed using canonical angles between two subspaces and its relation to distance on a Grassmann manifold.

lustrates the computation of the canonical angles between two corresponding subspaces generated from the visualizations of protein structures p and q. Since in this research

we conducted classification at class-level which requires the overall structural similarity, instead of using only the first canonical angle, we used the average of the canonical angles defined in (3.1). The final similarity measure between subspaces $\mathcal{Q}$ and $\mathcal{P}$ from (3.1) is then computed as

$$Sim(\mathcal{Q}, \mathcal{P}) = \frac{1}{min(N, M)} \sum_{i=1}^{min(N,M)} \cos^2 \theta_i. \tag{3.11}$$

A Grassmann manifold $\mathcal{G}(N, d)$ is defined as a set of N-dimensional subspaces of $\mathbb{R}^d$. In a Grassmann manifold, a subspace $\mathcal{P}$ is regarded as a point, as shown in Figure 3.13, represented by a $d \times N$ orthonormal matrix $\mathbf{U}$ that has the set of basis vectors $[\phi_1, \ldots, \phi_N]$ as its columns. The computation of canonical angles between subspaces can be regarded as the simplest computation of distance on the Grassmann manifold.

In the classification task, one subspace is used to represent one protein structure. However, each subspace is modeled without taking into account the variation between the subspaces. We can further extract powerful features for the corresponding protein structures to maximize the separation between different class proteins and minimize the variation within the same class proteins, by applying discriminant analysis. Here, we start discussion by describing the conventional discriminant analysis, followed by the Grassmann Discriminant Analysis which applies discriminant analysis to the set of points on a Grassmann manifold.

Let $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ and $\mathbf{Y} = \{y_i\}_{i=1}^n \in \{1, \ldots, C\}$ be a pair of a set of samples and a set of class labels, respectively. Linear discriminant analysis (LDA) is a supervised feature extraction technique that searches for a transformation $\mathbf{W}$ of $\mathbf{X}$ that maximizes the between-class variation and minimizes the within-class variation, through the optimization of the Rayleigh coefficient:

$$J(\mathbf{W}) = \frac{\mathbf{W}^\top \mathbf{S}_b \mathbf{W}}{\mathbf{W}^\top \mathbf{S}_w \mathbf{W}}, \tag{3.12}$$

$$\mathbf{S}_b = \sum_{c=1}^{C} n_c (\mu_c - \mu)(\mu_c - \mu)^\top, \tag{3.13}$$

$$\mathbf{S}_w = \sum_{c=1}^{C} \sum_{y_i \in c} (\mathbf{x}_i - \mu_c)(\mathbf{x}_i - \mu_c)^\top, \tag{3.14}$$

where $\mu_c = \frac{1}{n_c}\sum_{y_i=c}\mathbf{x}_i$ is the mean of the sample data of class c, and $\mu = \frac{1}{n}\sum_i \mathbf{x}_i$ is the mean of all samples. Assuming that $\mathbf{S}_w$ is invertible, the solution for $\mathbf{W}$ is the set of the $C-1$ largest eigenvectors of $\mathbf{S}_w^{-1}\mathbf{S}_b$.

To apply discriminant analysis on a Grassmann manifold, LDA is kernelized with the Grassmann kernel [50]. Let $\mathbf{\Phi}(\mathbf{x})$ be a function that maps $\mathbf{x}$ to a feature space $\mathcal{G}$. Assume that the solution $\mathbf{W} \in \mathcal{G}$ can be written as a linear combination of the mapped data [71]:

$$\mathbf{W} = \sum_{i=1}^{n} \alpha_i \mathbf{\Phi}(\mathbf{x}_i). \tag{3.15}$$

To avoid a direct computation of $\mathbf{\Phi}(\mathbf{x})$ in (3.15), dot products can be used. Suppose that $k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{\Phi}(\mathbf{x}_i) \cdot \mathbf{\Phi}(\mathbf{x}_j))$. Then

$$\begin{aligned} \mathbf{W}^\top \mu_c &= \frac{1}{n_c}\sum_{i=1}^{n}\sum_{y_j \in c} \alpha_i (\mathbf{\Phi}(\mathbf{x}_i) \cdot \mathbf{\Phi}(\mathbf{x}_j)) \\ &= \frac{1}{n_c}\sum_{i=1}^{n}\sum_{y_j \in c} \alpha_i k(\mathbf{x}_i, \mathbf{x}_j) \\ &= \hat{\alpha}^\top \hat{\mu}_c. \end{aligned} \tag{3.16}$$

By using (3.16), the numerator and denominator of (3.12) become

$$\begin{aligned} \mathbf{W}^\top \mathbf{S}_b \mathbf{W} &= \mathbf{W}^\top \left[\sum_{c=1}^{C} n_c (\mu_c - \mu)(\mu_c - \mu)^\top\right] \mathbf{W} \\ &= \hat{\alpha}^\top \left[\sum_{c=1}^{C} n_c (\hat{\mu}_c - \hat{\mu})(\hat{\mu}_c - \hat{\mu})^\top\right] \hat{\alpha} \\ &= \hat{\alpha}^\top \mathbf{M} \hat{\alpha} \end{aligned} \tag{3.17}$$

$$\begin{aligned} \mathbf{W}^\top \mathbf{S}_w \mathbf{W} &= \mathbf{W}^\top \left[\sum_{c=1}^{C}\sum_{y_i \in c} (\mathbf{\Phi}(\mathbf{x}_i) - \mu_c)(\mathbf{\Phi}(\mathbf{x}_i) - \mu_c)^\top\right] \mathbf{W} \\ &= \hat{\alpha}^\top \left[\sum_{c=1}^{C}\sum_{y_j \in c} (\mathbf{K}_j - \hat{\mu}_c)(\mathbf{K}_j - \hat{\mu}_c)^\top\right] \hat{\alpha} \\ &= \hat{\alpha}^\top \mathbf{N} \hat{\alpha}. \end{aligned} \tag{3.18}$$

The objective function (3.12) is then rewritten as

$$J(\hat{\alpha}) = \max_{\hat{\alpha}} \frac{\hat{\alpha}^\top \mathbf{KMK} \hat{\alpha}}{\hat{\alpha}^\top \mathbf{KNK} \hat{\alpha} + \sigma \mathbf{I}}, \tag{3.19}$$

where σ is the regularization parameter. For the Grassmann kernel, the following projection kernel can be used [50]:

$$k(\mathbf{P}, \mathbf{Q}) = \|\mathbf{P}^\top \mathbf{Q}\|_F^2, \tag{3.20}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are the projection matrices of subspaces $\mathcal{P}$ and $\mathcal{Q}$, which correspond to proteins p and q, respectively. Essentially, the projection kernel in (3.20) is equivalent to the similarity measures in (3.11).

3.2.4 Flow of the classification framework

Let $\{p_i\}_{i=1}^n$ and $\{y_i\}_{i=1}^n$ be a set of reference protein structures and a set of class labels, respectively. Let R and V be the number of rotations and visualization types for generating multiple views of 2D protein images, respectively.

Training phase:

Step 1 For each p_i, generate R protein images for each visualization ($\{I_r^{(v)}\}$, where $r = 1, \ldots, R$ and $v = 1, \ldots, V$) using Jmol [54].

Step 2 Apply feature extraction to each image in $\{I_r^{(v)}\}$ to obtain a set of feature vectors $\{\mathbf{x}_r^{(v)}\}$.

Step 3 Generate a subspace $\mathcal{P}_i^{(v)}$ for each visualization v.

Step 4 Compute the kernel matrix $\mathbf{K}_{train}$ for each pair of subspaces in the training set. When using multiple visualizations, the kernel matrix is computed using the average similarities from the multiple visualizations.

Step 5 Compute α in (3.19), using the methods given in [72].

Step 6 Compute the new training feature vectors $F_{train(i)} = \alpha^\top \mathbf{K}_{train(i)}$.

Testing phase:

Step 1 Generate R protein images for each visualization of input protein q ($\{I_r^{(v)}\}$, where $r = 1, \ldots, R$ and $v = 1, \ldots, V$) by using Jmol [54].

Step 2 Apply feature extraction to each image in $\{I_r^{(v)}\}$ to obtain a set of feature vectors $\{\mathbf{x}_r^{(v)}\}$.

Step 3 Generate the input subspace $\mathcal{Q}^{(v)}$ for each visualization v.

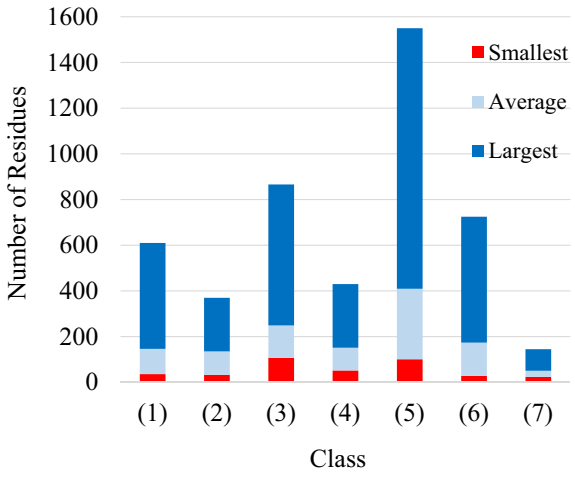

Figure 3.14: Sizes of proteins used in the experiment (number of residues): (1) α; (2) β; (3) α/β; (4) $\alpha+\beta$; (5) multidomain; (6) membrane; and (7) small proteins.

Step 4 Compute the kernel matrix $\mathbf{K}_{test}$ as the similarity between $\mathcal{Q}^{(v)}$ and $\{\mathcal{P}_i^{(v)}\}$ by using (3.11). When using multiple visualizations, the kernel matrix is computed using the average similarities from the multiple visualizations.

Step 5 Compute the new test feature vectors $F_{test} = \alpha^\top \mathbf{K}_{test}$.

Step 6 Classify the input protein using k-NN classification of $F_{train(i)}$ and F_{test}.

3.3 Experiments

To evaluate the effectiveness of the proposed similarity measure, we classified proteins into 7 classes according to the protein classification scheme of the SCOP database [10]: (1) α-proteins (containing mainly α-helices), (2) β-proteins (containing mainly β-sheets); (3) α/β-proteins (containing both α and β structures where the β-sheets are parallel); (4) $\alpha+\beta$-proteins (containing both α and β structures where the β-sheets are anti-parallel); (5) multi-domain proteins that have multi-functions; (6) membrane and cell surface proteins; and (7) small proteins. We compare our proposed method with the widely used alignment methods CE [23], FATCAT [24], and TM-align [25].

3.3.1 Experimental setting

We used 700 proteins (100 proteins for each class), which were randomly selected from the Astral SCOP dataset [11] so that each protein has $\leq$20% sequence similarity. Fig-

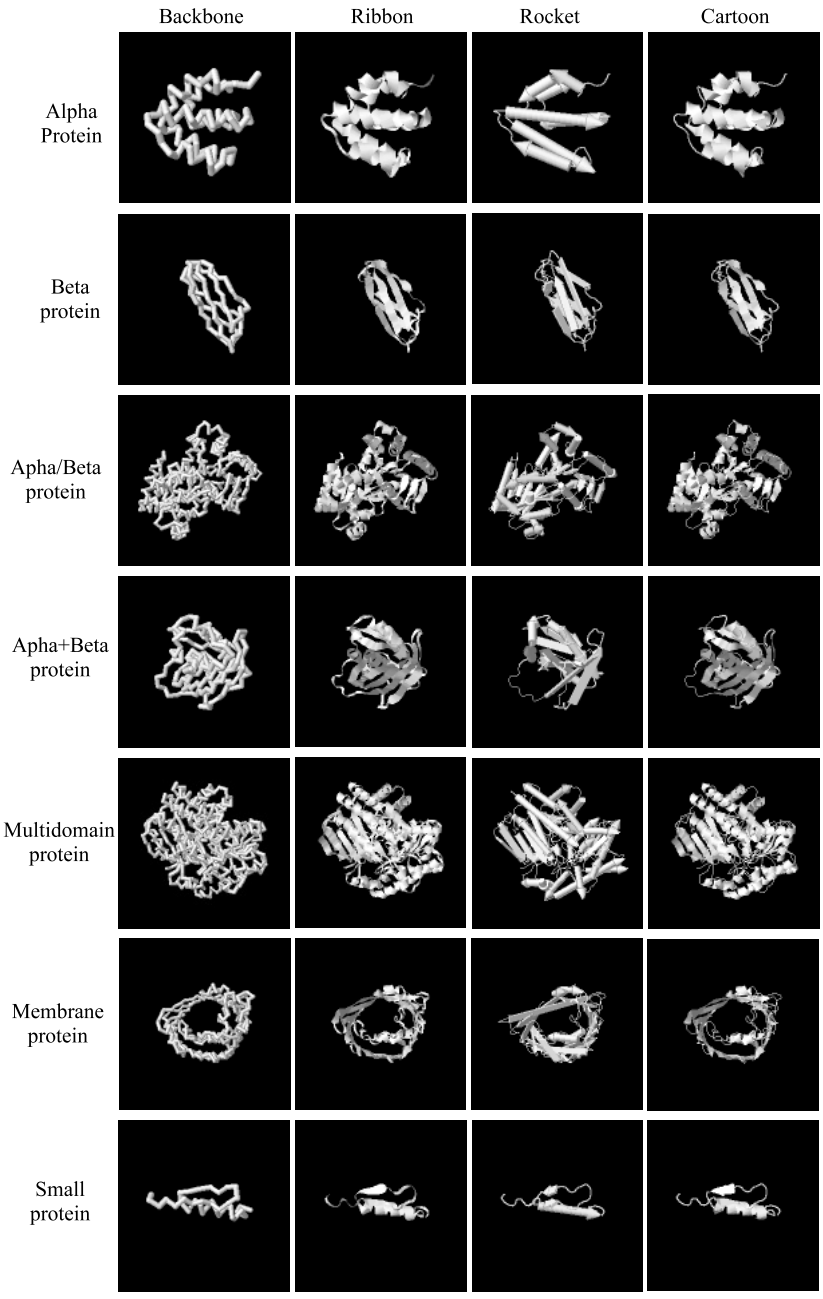

Figure 3.15: Examples of proteins used in the experiments.

ures 3.14 and 3.15 show the plot of the protein sizes and the examples for each class, respectively. In the experiments, 10-fold validation scheme was used. Ten proteins

Table 3.1: List of feature extraction parameters and dimensionality for each feature.

	HLAC	LBP^{u2}	LBP^{ri}	LBP^{riu2}	RIC-LBP
Parameters	$\Delta = 2, 4, 8$	$r = 1, 2, 4$ $P = 8$	$r = 1, 2, 4$ $P = 8$	$r = 1, 2, 4$ $P = 8$	$(r, s) = (1, 2)$, $(2, 4), (4, 8)$
Dimensionality	105	177	108	30	408

from each class were used for the test and the rest for training and the experiments were repeated 10 times. Most of the proteins that belong to the same family are used as either training or testing, but not both. This strategy was used to increase the variability between training and testing proteins that belong to the same class.

Jmol [54] was used to synthesize 500 uniformly rotated 128×128 grayscale visualization images of the backbone, ribbon, rocket, and cartoon types. HLAC, LBP^{u2}, LBP^{ri}, LBP^{riu2}, and RIC-LBP were used as features for the visualization images. The parameters and dimensionality of each feature are shown in Table 3.1. Since several parameters were used, the final feature vector is the result of concatenating the feature vectors extracted by using each parameter. Cross validation of training data was used for tuning the parameter k for k-NN and subspace dimension.

Classifications using conventional methods were similarly conducted using 10-fold validation scheme. First, the similarity or dissimilarity between an input protein and the training protein was computed. Then, k-NN was used to classify the protein. The parameter for k-NN was automatically tuned by cross validation of the training set. For the TM-align method, since the TM-score is not symmetric, the average of the TM-scores was used as the similarity measure.

3.3.2 Single visualization

Table 3.2 shows the results of the classification for the conventional methods. The experimental results for the view-based methods using MSM and GDA are reported in Tables 3.3 and 3.4, respectively.

When using RMSD, CE only obtained the average accuracy of 33.86%. The performance of CE improved to 49.14% when using Z-score statistics. Table 3.5 shows the confusion matrix for CE when RMSD was used as the distance metric. When using RMSD, CE was unable to classify multi-domain proteins, and frequently classified input proteins as either membrane or small proteins. These experimental results confirm the limitations of RMSD as a distance measure for protein structure classification task.

From Tables 3.3 and 3.4, we can see that the overall performance when using GDA

Table 3.2: Classification results for conventional methods (accuracy in %).

Methods/ Class	(1)	(2)	(3)	(4)	(5)	(6)	(7)	Avg.
CE [23, 73]								
- RMSD	9	36	42	4	0	**78**	68	33.86
- Z-score	**80**	**82**	69	32	16	32	33	49.14
FATCAT (Raw score) [24, 73]	62	63	74	36	57	51	29	53.14
TM-align (TM-Score)[25]	76	80	**89**	45	12	77	69	64.00

was significantly better than that of using only MSM. When using backbone visualization, the accuracy of the classification using GDA reached 60%, outperforming CE and FATCAT. On the other hand, when using MSM, the accuracy was only up to 51.71%. The proposed method using GDA with rocket visualization and LBP-based feature achieved accuracy rates of more than 65%, which outperformed TM-align.

In general, alignment-based methods attempt to align and superpose the backbone structures of the proteins. Methods such as TM-align [25] apply secondary structure alignment as an initial alignment. In the view-based method, the usage of rocket visualizations can be regarded as analogous to the alignment method in that it uses information about the secondary and backbone structures.

Tables 3.6 and 3.7 show the confusion matrices for TM-align and rocket visualization with LBP^{u2}, respectively. From the confusion matrix shown in Table 3.7, we can see clearly that by using the visualization images, the multi-domain protein class (5), which contains mostly large sized proteins, and the small protein class (7), which contains mostly proteins with fewer residues than in other classes, can be classified more accurately than by the use of alignment-based methods.

3.3.3 Multiple visualizations and effect of rotation number

As mentioned previously, each type of visualization exposes different structural characteristics. Here, we demonstrate how the use of multiple visualizations improves the classification results. We used a simple approach to combination, taking the average of the similarity values of all visualizations. Table 3.8 shows a summary of the experi-

Table 3.3: Classification results for view-based MSM (accuracy in %).

Methods/ Class	(1)	(2)	(3)	(4)	(5)	(6)	(7)	Avg.
Backbone:								
- HLAC	47	65	44	30	54	47	48	47.86
- LBP^{u2}	30	59	55	24	49	53	77	49.57
- LBP^{ri}	41	48	61	20	49	41	78	48.29
- LBP^{riu2}	28	42	42	29	52	41	61	42.14
- RIC-LBP	42	60	57	28	43	50	82	51.71
Ribbon:								
- HLAC	55	62	39	22	57	40	58	47.57
- LBP^{u2}	49	71	60	37	46	45	85	56.14
- LBP^{ri}	55	52	45	22	51	31	81	48.14
- LBP^{riu2}	38	54	56	28	55	36	81	49.71
- RIC-LBP	57	71	68	28	45	40	**92**	57.29
Rocket:								
- HLAC	50	63	49	31	46	39	77	50.71
- LBP^{u2}	57	62	63	44	52	53	60	55.86
- LBP^{ri}	69	66	56	36	42	46	73	55.43
- LBP^{riu2}	66	74	54	26	46	44	58	52.57
- RIC-LBP	65	71	62	43	56	49	80	60.86
Cartoon:								
- HLAC	62	60	34	23	53	38	58	46.86
- LBP^{u2}	52	66	52	35	49	47	84	55.00
- LBP^{ri}	61	58	49	19	**64**	31	77	51.29
- LBP^{riu2}	45	58	46	18	47	33	77	46.29
- RIC-LBP	57	66	65	28	55	42	**92**	57.86

mental results with using various configurations of visualizations and RIC-LBP features with GDA. By using the unweighted combination of ribbon, rocket, and cartoon visualizations, the proposed method achieved a 69.43% accuracy rate, which is much better than that of the TM-align.

Next, we evaluated the effect of the number of multi-view images, that is, the number of rotations, on the classification performance of the proposed method using all the visualizations and combinations with different feature extraction methods used in the previous experiments. Figure 3.16 shows box plots of the experimental results with rotation numbers of $50, 100, 500$, and 1000. It can be seen that the overall experimental

Table 3.4: Classification results for view-based GDA (accuracy in %).

Methods/ Class	(1)	(2)	(3)	(4)	(5)	(6)	(7)	Avg.
Backbone:								
- HLAC	58	66	53	33	63	41	82	56.57
- LBP^{u2}	50	71	53	39	61	45	79	56.86
- LBP^{ri}	54	74	54	46	59	52	81	60.00
- LBP^{riu2}	52	66	44	38	59	54	80	56.14
- RIC-LBP	60	67	55	42	54	54	**90**	60.29
Ribbon:								
- HLAC	58	68	52	44	**64**	38	85	58.43
- LBP^{u2}	62	65	48	51	**64**	58	82	61.43
- LBP^{ri}	68	68	58	38	53	49	86	60.00
- LBP^{riu2}	61	66	61	35	52	48	83	58.00
- RIC-LBP	66	73	54	50	53	61	86	63.29
Rocket:								
- HLAC	75	67	53	35	59	46	84	59.86
- LBP^{u2}	73	74	66	**53**	54	57	84	**65.86**
- LBP^{ri}	68	**82**	61	51	55	55	88	65.71
- LBP^{riu2}	64	78	59	52	54	47	82	62.29
- RIC-LBP	72	72	63	**53**	63	54	83	65.71
Cartoon:								
- HLAC	62	69	51	39	58	44	80	57.57
- LBP^{u2}	62	72	56	48	52	56	81	61.00
- LBP^{ri}	59	72	61	40	52	53	86	60.43
- LBP^{riu2}	57	67	66	35	54	49	84	58.86
- RIC-LBP	67	72	58	48	55	62	88	64.29

results for different feature extraction methods and visualizations converged when the number of the rotations reached 500.

3.3.4 Significance test over alignment method

To ensure the performance difference in between our method and alignment method is marginally significant, we conducted statistical test using McNemar's test [74], which is a nonparametric test typically used for comparing two classification methods in terms of classification errors. The null hypothesis assumes that both methods have the same performance. The procedure of the McNemar's test to compare methods $M1$ and $M2$

Table 3.5: Confusion matrix for CE with RMSD.

		Predicted class						
		(1)	(2)	(3)	(4)	(5)	(6)	(7)
True class	(1)	9	0	0	0	1	65	25
	(2)	0	36	0	0	2	3	59
	(3)	1	0	42	1	2	21	33
	(4)	1	1	1	4	1	30	62
	(5)	2	0	10	0	0	28	60
	(6)	3	0	1	0	1	78	17
	(7)	1	7	2	3	6	13	68

Table 3.6: Confusion matrix for TM-align.

		Predicted class						
		(1)	(2)	(3)	(4)	(5)	(6)	(7)
True class	(1)	76	0	0	3	0	20	1
	(2)	5	80	0	7	1	7	0
	(3)	3	1	89	2	0	5	0
	(4)	6	14	12	45	1	20	2
	(5)	21	4	20	13	12	30	0
	(6)	17	6	0	0	0	77	0
	(7)	8	6	0	13	0	4	69

Table 3.7: Confusion matrix for rocket with LBP^{u2}+GDA.

		Predicted class						
		(1)	(2)	(3)	(4)	(5)	(6)	(7)
True class	(1)	73	0	2	2	5	13	5
	(2)	0	74	3	14	3	0	6
	(3)	2	0	66	10	20	1	1
	(4)	3	18	15	53	8	0	3
	(5)	5	0	31	7	54	3	0
	(6)	23	5	0	6	6	57	3
	(7)	5	6	0	3	0	2	84

is as follows.

Step 1 : Generate a (2×2)-contingency matrix, with the following elements $\{c_{ij}\}$. The element c_{10} is the number of tests in which the test data are classified correctly by $M1$ but misclassified by $M2$; c_{11} is the number of tests in which the test data are correctly classified by both methods; c_{01} and c_{00} are the complements of c_{10}

Table 3.8: Experimental results, using a combination of multiple visualizations and RIC-LBP with GDA (accuracy in %).

Backbone	Ribbon	Rocket	Cartoon	(1)	(2)	(3)	(4)	(5)	(6)	(7)	Avg.
✓	✓			64	75	63	53	50	59	**90**	64.86
✓	✓	✓		68	74	68	53	57	**64**	88	67.43
✓	✓	✓	✓	66	74	**69**	51	56	61	88	66.43
✓		✓		**72**	75	66	53	61	61	82	67.14
✓		✓	✓	68	75	66	53	**64**	62	89	68.14
✓			✓	67	76	57	52	59	59	**90**	65.71
	✓	✓		69	**78**	64	57	54	57	88	66.71
	✓	✓	✓	**72**	77	66	**59**	**64**	61	87	**69.43**
		✓	✓	70	74	65	56	62	63	86	68.00

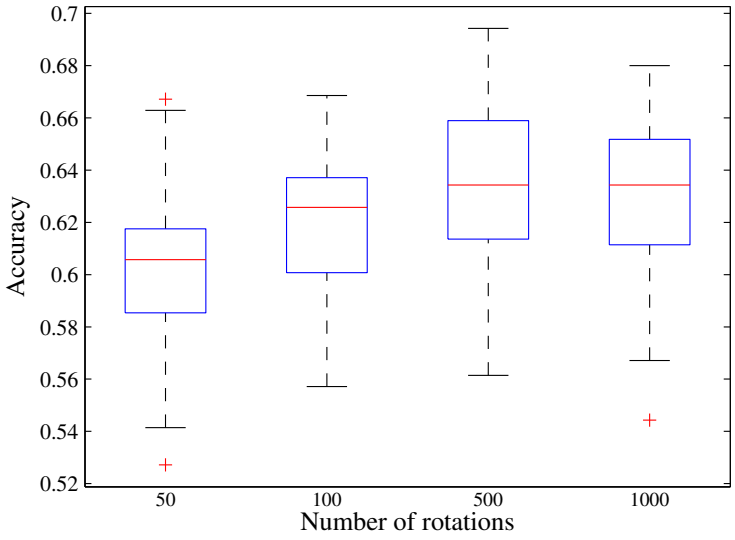

Figure 3.16: Accuracy of the proposed method, using various parameters for visualization and different feature extraction methods, for different numbers of view points (rotations).

and c_{11}, respectively.

Step 2 : Compute the χ^2 statistic: $\chi^2_{Mc} = \frac{(|c_{01}-c_{10}|-1)^2}{c_{01}+c_{10}}$.

Step 3 : Obtain the p-value for this statistic from the χ^2 distribution with 1 degree

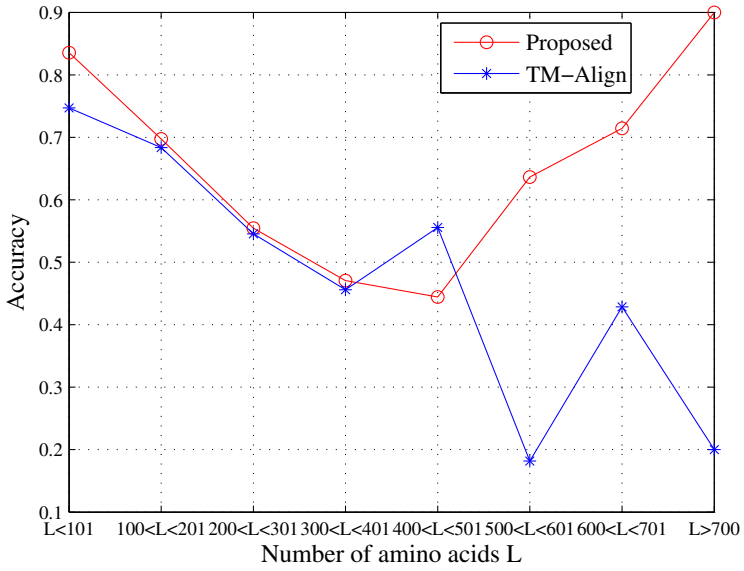

Figure 3.17: Plot of accuracy over the number of the amino acids. The horizontal axis indicates the length of the amino acid sequence. The vertical axis indicates the accuracy of the classification results.

of freedom.

We compared the performance of the proposed method, when using 500 rotation and RIC-LBP features, with that of the TM-align based method. The McNemar's test produced $p = 0.01338$, which means that we can reject the null hypothesis at the 5% level and conclude that the classification performance of the proposed method is better than that of the TM-align based method.

3.4 Further discussion

In this chapter, we discuss further the strength and weakness of the proposed method. Figure 3.17 shows the overall accuracy for different size of proteins, where the proposed method was using multiple visualization (ribbon, rocket, and cartoon) with RIC-LBP and GDA. Similarly, Figure 3.18 shows a plot of all the proteins and the proteins that were correctly classified by the same proposed method. It can be seen that the proposed method can correctly classify proteins with a wide range of sizes. As a comparison, Figure 3.19 shows the same plot for the TM-align method.

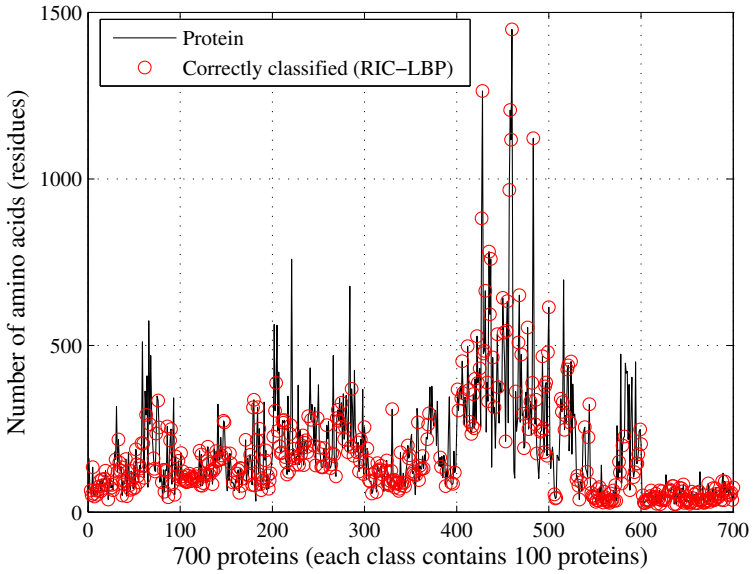

Figure 3.18: Plot of the proteins correctly classified by the proposed method (ribbon, rocket, and cartoon visualization with RIC-LBP). The horizontal axis indicates the protein structure (1 to 100: α-proteins; 101 to 200: β-proteins; 201 to 300: α/β-proteins; 301-400: $\alpha+\beta$-proteins, 401-500: multidomain proteins; 501-600: membrane proteins; 601-700: small proteins). The vertical axis gives the size of the protein (number of amino acids).

In our classification experiment, the protein structures are categorized at class-level hierarchy which is still at a coarse-level. In this case, the proteins that belong to the same class category do not necessarily have very similar structures. By using the set of the multiple view of visualization images, the proposed method captures the overall structures of the proteins while discards some of the geometrical detail. Large protein tends to have a complicated structure; in this case, view-based method can capture the overall complex structure, while discarding the geometrical detail which does not contain discriminative information for classification at the class-level hierarchy. Thanks to this, view-based method performed much better than the alignment based method for classifying the large proteins. Another thing to be noted is that since protein is functioning by binding with other molecules, the important parts are the structures on the *outer* of the protein, which are visible through the use of the multi-view images. However, for the case of classification of proteins with similar structures, alignment based methods may outperform the view-based method since they can take into account

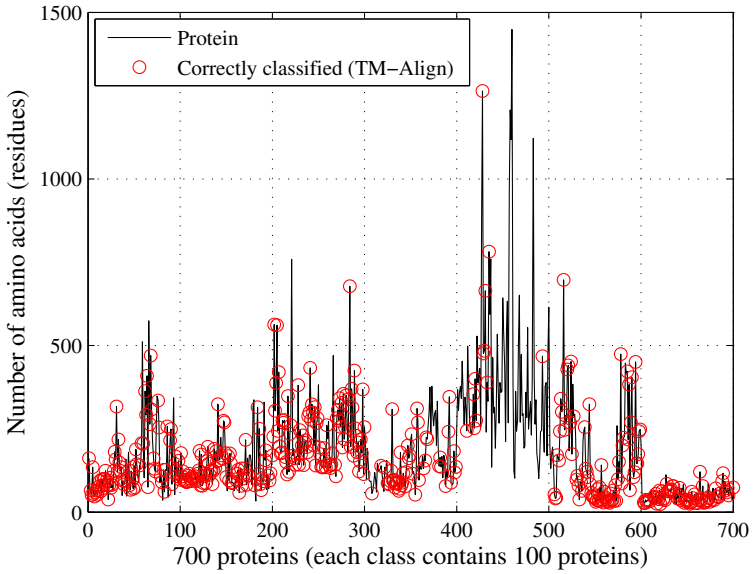

Figure 3.19: Plot of the proteins correctly classified using the TM-align method. The horizontal axis indicates the protein structure (1 to 100: α-proteins; 101 to 200: β-proteins; 201 to 300: α/β-proteins; 301-400: $\alpha+\beta$-proteins, 401-500: multidomain proteins; 501-600: membrane proteins; 601-700: small proteins). The vertical axis gives the size of the protein (number of amino acids).

the 3D geometrical detail and local topology of the protein structures.

3.5 Online system: View-based Protein Comparison (VPC)

Unlike the conventional methods, the proposed method requires the generation of visualization images and feature extractions so that the implementation can be slightly complicated. To ease the difficulties, we designed a modular framework for implementing the proposed method and developed an online system, called View-based Protein Comparison (VPC), which is an implementation of our basic idea that uses the canonical angles as similarity measures between two protein structures[1]. In the following, we describe how we implement the algorithm. Then, we show the computational speed of the current implementation.

[1]URL: http://www.cvlab.cs.tsukuba.ac.jp/~chendra/proteinfront/

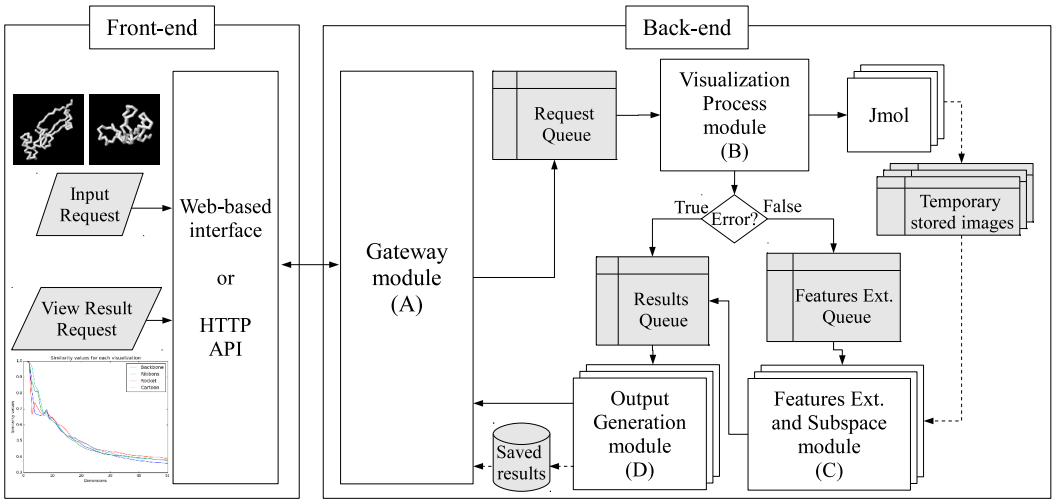

Figure 3.20: General process flow of the VPC system. We adopted a modular architecture to improve scalability and easy maintenance, by using POSIX interprocess communication. First, front-end receives requests either through web-based interface or HTTP API. The input proteins can be in the form of PDB file format or the seven-characters SCOP ID. Then, the requests are passed to the back-end through Gateway module (A). User will be given a unique code to each request, so that the user can query the results any time. Each request received by module (A) is stored temporarily in the message queue, called request queue. Visualization Process module (B) keeps checking the request queue. If any queue comes, visualization process then starts by spawning Jmol. When generation of multi-view images is completed without error, a queue is passed to feature extraction queue. On the other hand, if any error occurs, an error message is passed to results queue. The feature extraction queue is watched by Feature Extraction and Subspace module (C), which applies feature extractions, generates subspaces, and calculates the canonical angles. Finally, if all the process is either successfully completed or with error, the results are passed to the results queue. The results queue is handled by Output Generation module (D), which generates the final output files and store them to the magnetic disk.

3.5.1 Implementation of VPC

The general process flow of our system is shown in Figure 3.20. The system consists of two Linux OS based machines as follows:

1. Front-end: Shared web server using Intel Celeron G1101 2.27 Ghz with 512MB RAM.

2. Back-end: Intel dual Xeon E-2680V2 2.8Ghz with 128GB RAM, where 32 GB of

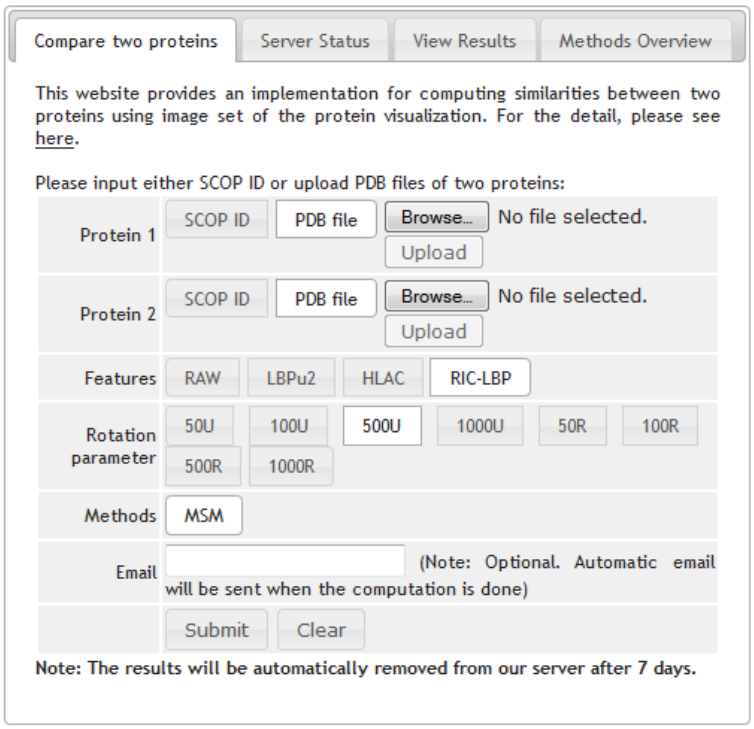

Figure 3.21: User interface of the VPC system.

Table 3.9: Average time for computing one pair of protein structure similarities using our demonstration online system.

Process	A (in seconds)	B (in seconds)
Visualization using Jmol	197	19
Feature extraction of RIC-LBP	442	61
Subspace generation and similarity measurement	17	8

the RAM is specifically used as virtual disk.

The front-end machine, developed mostly using PHP, accepts requests from either the web-based interface or through HTTP API. The back-end machine was developed mostly using Python, Bottlepy [75], SciPy [76], and Jmol [54] version 13. We used the virtual disk to temporarily store the visualization image files.

Figure 3.21 shows the web interface of VPC system. Through the web interface, VPC system accepts requests to compare two proteins in the form of PDB files or inputting the SCOP ID. The available feature types are the raw pixel values, HLAC, LBP^{u2}, and the RIC-LBP features. The option for rotation parameters are 50, 100,

500, or 1000 uniform or random rotations. For every request, our system produces a unique code for the user, so that the results can be queried any time through the web interface that we provide (the results are stored for 7 days). The output of our online tool is a compressed Matlab file format containing the set of the extracted features, basis vectors and eigenvectors of the corresponding N-dimensional subspaces, and the average of the n-th cosine of the canonical angles, where $n = 1, \ldots, N$, and N is fixed to 50. Through the web interface, users can also view the plot of the nth similarity values and the 3D visualization of the protein structures.

3.5.2 Computational speed of VPC

The computational speed of our system is as follows. To complete 100 concurrent requests for computing the protein structure pair similarities, with parameters of 500 uniform rotations and RIC-LBP features, our system took time about 741 seconds. The average computational speed for one request of the 100 concurrent request is shown on the column A in Table 3.9. Column B in Table 3.9 shows the average computational speed for one request of 10 consecutive requests where none of them were executed at the same time. We note that the computational speed can further be improved by optimizing the implementation using C/C++ and GPGPU (General-purpose computing on Graphics Processing Units) programming and removing the dependency with third party molecular visualization software that is not specifically designed for our purpose.

3.6 Summary

We proposed a new approach to protein structures comparison based on image sets synthesized by the protein visualization. In this approach, a protein structure is encoded as a subspace generated from a set of 2D visualizations of the 3D structures. Then, the similarity between protein structures is defined by the canonical angles between the corresponding subspaces. The main advantage of this approach is that alignment is not required, so that proteins of the same class that have very different structures can be classified more accurately. Multiple similarities can also be obtained, although here we simply used the average of the similarity degrees. Through different types of visualization, we can also embed additional information, such as the secondary structure of the protein. We adopted Grassmann discriminant analysis into the classification framework to extract discriminative features from the corresponding subspaces for each

protein structure. The validity of the proposed method is demonstrated through experiments classifying proteins with 20% sequence identity into 7 classes according to the SCOP classification scheme. The experimental results demonstrate that the view-based method outperformed conventional methods, especially for the classes where the the proteins have very different structures. To ease the difficulties in implementing the proposed method, we also designed a software framework and developed an online protein comparison system, called View-based Protein Comparison (VPC) system.

This research can also be considered as a proof of concept that view-based method is useful and has big potential in protein structure analysis. The experimental results can be regarded as a baseline performance of the view-based method. To further improve the performance, some directions for future works include the parameter tuning related to the generation of the visualization images. For example, embedding amino acid types through different color and adjusting illumination setting, initial structural orientation, and *virtual camera* parameters may outcome better descriptor for comparing the structures. Since proteins carry out biological function by binding with other molecules, the most important parts are the structures on the surface of the protein. This suggests that while complicated structures can contain many occluded parts, the occlusion can be dealt with simultaneous usage of multiple *virtual cameras* and adaptive number of view-points for the visualization images. To deal with classification at fold-level and deeper hierarchies where proteins with similar structures belong to different categories, preprocessing such as local structural segmentation and rough local alignment prior to the generation of the visualization images can be considered.

Finally, the use of subspace as the protein structure representation can be extended to embed various protein properties, not only visualization images, and be used for different purposes other than classification task. In the future, we intend to study the adoption of our current work, which is based on the subspace representation including nonlinear subspace, to other tasks such as protein structure prediction and protein docking problem.

Chapter 4

Combination of Distance Metrics for Protein Fold Recognition

As mentioned in Chapters 2 and 3, many methods have been proposed for comparing the similarity or dissimilarity between protein fold structures. It is difficult to select one best method from a number of the protein structural comparison methods. For example, the methods based on the molecular visualization are superior for classifying protein that have large intra-variation but have problems when classifying proteins with very similar structures. In contrast, alignment-based methods are superior for differentiating very similar protein structures but inferior for comparing very different structures. This brought us to the idea of combining multiple methods to optimize the classification peformance which is the main focus of this chapter. We generalize the concept of the large margin nearest neighbor (LMNN) to learn an optimal weight combination for multiple metrics. The rest of this chapter is organized as follows. The background of the proposed method is discussed in Chapter 4.1. Related works on fusion techniques are reviewed in Chapter 4.2. A brief overview of LMNN is provided in Chapter 4.3. The proposed method is described in Chapter 4.4. The experimental results and discussion are provided in Chapter 4.5. Finally the summary is given in Chapter 4.6.

4.1 Background

In general classification task, many fusion and metric learning algorithms have been proposed for combining multiple methods to improve the classification results [77, 78, 79, 80, 81, 82, 83]. However, most of the methods require the availability of a

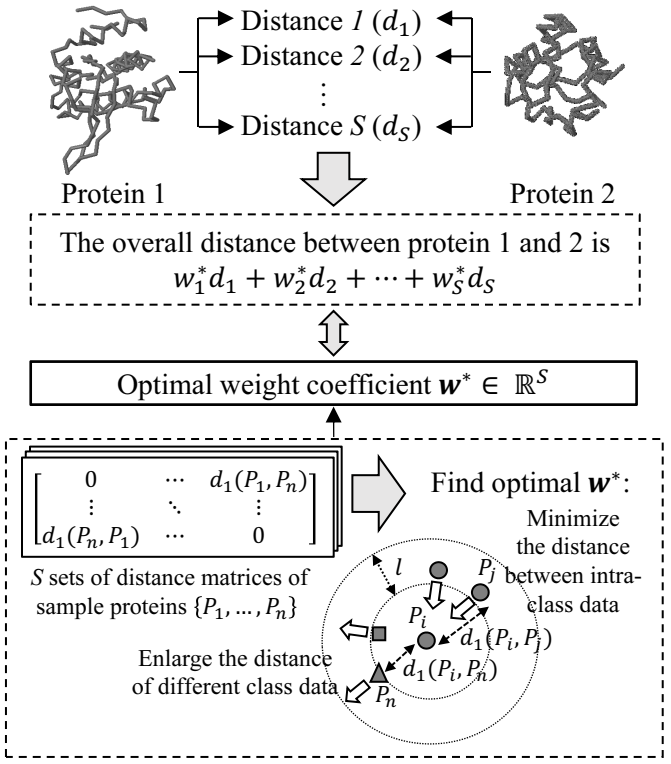

Figure 4.1: Basic idea of the proposed metrics combination method. Let $d_s(P_i, P_j)$ be the distance between two sample proteins P_i and P_j computed using method s. The objective is to optimize the weight combination of multiple distance measures $\{1, \ldots, S\}$, by minimizing the distances between each protein and its intra-class proteins and enlarging the distances of different class proteins under margin l, using only the sets of the distance matrices.

feature vector representation. Unfortunately, for the case of protein structural comparison, feature vector representation is not always available. For example, alignment methods [22, 23, 24, 25, 73] only produce either distances or similarity values with the aligned sub-structures. Subspace-based representation proposed in Chapter 3 and Graph-based representation [84] are another two examples of nontrivial features representation for protein structures, from which only similarity metrics can be obtained.

Here, we propose a combination of multiple distance metrics for protein fold classification, illustrated in Figure 4.1. Given S distance measures $\{d_i\}_{i=1}^S$ (assumed to be obtained by using multiple techniques), our task is to learn an optimal weight coefficient $\boldsymbol{w}^* \in \mathbb{R}^S$ by minimizing the distance between samples that belong to the same class and enlarging the distance of samples that belong to other classes only using the sets of

the distance matrices. This concept is similar to that of the large margin nearest neighbor (LMNN) [79] and closely related to the support vector machine (SVM), wherein the separation between classes is optimized according to convex optimization with a hinge loss function. However, unlike SVM, which is theoretically designed for two-class classification tasks, our method and the conventional LMNN are naturally applicable to multi-class problems. While the original LMNN learns a Mahalanobis-based distance metric from a set of feature vectors of training data, our proposed method learns an optimal weight coefficients for combination of the distance metrics from training data. The final distance, termed as *overall distance*, is then defined as the linear combination of the distances that can be used for any distance-based classification such as k-nearest neighbor classification.

The main advantage of our proposed method is the capability in finding an optimal combination for many distance metrics, possibly including poor metrics (metrics that output poor performance in the classification). Accordingly, when there are a number of distance measures available, we can eliminate the difficulties in selecting the appropriate measures for the combination. In practice, this property is important because the distance measure that can perform the best on a certain data is commonly not known beforehand.

We demonstrate the effectiveness of the proposed method through classification experiments on 27 fold classes of proteins using Ding Dubchak dataset [13, 14] and six classes of protein enzymes using ENZYMES dataset [84]. Our proposed method is closely related to multiple kernel learning (MKL) [85, 86, 87], where MKL combines multiple kernel matrices instead of distances. Therefore, we compared the performance of our proposed method with generalized MKL (GMKL) [86] in addition to naïve averaging and voting.

In brief, the contributions of this work are as follows:

- Generalization of the concept of the LMNN to learn optimal weight coefficients for a combination of distance metrics.
- Introduction of three loss functions to the problem formulation for combining distance metrics using hinge loss, smooth hinge loss, and logistic loss.
- Comprehensive experiments on protein fold and enzymes classification, including performance comparison with naïve methods (averaging and voting) and GMKL [86] using the public protein dataset of Ding Dubchak [13, 14] and ENZYMES [84].

4.2 Related works

Metric learning is defined as a method that constructs a new distance function which hopefully perform better than that of the original distance function [88]. We regard our method is closely related to metric learning because our method learns a new metric from given multiple metrics by minimizing the L2 distances of the same class data and enlarging those of the different classes' data. There are many metric learning methods have been proposed. One of the most well-known metric learning methods is large margin nearest neighbor (LMNN) [79]. LMNN learns a Mahalanobis distance metric for k-NN classification, such that the data within the k-nearest neighbor belong to the same class and the data of different classes are separated by a large margin. Another example is relevant component analysis (RCA) [77]. RCA learns a Mahalanobis distance metric by first generating a set of *chunklet*[1]. The covariance matrix in RCA is then computed using weighted scatter matrix based on the chunklets. In [78], Neighborhood Component Analysis (NCA) was proposed for improving k-NN classification. NCA learns Mahalanobis distance metrics by explicitly maximizing the probability of a data point to be correctly classified through introduction of a stochastic cost function solved using gradient decent. In [80], instead of Mahalanobis-based distance metric learning, a random forest-based metric learning was proposed by considering both the relative location and the absolute positions of the data samples. There are many other metric learning methods such as [81, 82, 83]. However, all of them require feature vector representation of the data samples.

Since the proposed method combines multiple methods, the proposed method is also related to fusion techniques. The simplest approach to multiple methods combination is either by averaging or voting. As we combined multiple distance obtained from multiple methods, our work is very similar to multiple kernel learning (MKL) [85, 86, 87]. The difference is that MKL-based methods combine similarity matrices, while our method combines distance matrices.

From the view point of statistical analysis and information theory, finding an optimal combination from multiple methods is related to model selection, such as stepwise regression method, with Akaike information criterion (AIC) or Bayes information criterion (BIC) as the indicator of the model performance. However, stepwise regression can be suboptimal due to local-optima and the optimization of indirect performance indicator, while the proposed method directly optimizes the distance specifically for k-NN classification. Another problem is on the scalability since the computational cost

[1] Chunklet is defined as a set of data of the same class but the class label is unknown [77].

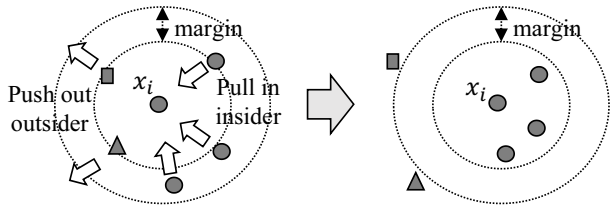

Figure 4.2: Illustration of how LMNN works. The small circle in the center represents one sample data point x_i. LMNN learns a transformation matrix from a set of feature vectors of samples to pull in data points of the same class and push out data points of different classes with a margin of one unit distance.

significantly increases by the number of the models.

4.3 Large margin nearest neighbor (LMNN)

Conventional metric learning algorithms search for a transformation of feature vectors of the sample data to obtain an optimal metric, used for task that depends on distances, such as classification using nearest-neighbor method [88]. In the following, we provide an overview of LMNN [79], which is the basis of our proposed method.

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ be a training set, where $x_i \in \mathbb{R}^d$ is a data point (a feature vector) and $y_i \in \{1, 2, \ldots, C\}$ is the class label of x_i. Then, LMNN searches for a linear transformation $\mathbf{W} : \mathbb{R}^d \to \mathbb{R}^f$ $(f \leq d)$, ensuring that the data points in the same class are brought closer to each other and that the margins between different classes are made larger, as shown in Figure 4.2. The cost function to be minimized consists of two terms as follows:

$$\begin{aligned} F(\mathbf{W}) = &\sum_{i=1}^{n} \sum_{j:y_j=y_i} \|\mathbf{W}(x_i - x_j)\|^2 + \\ &\mu \sum_{i=1}^{n} \sum_{j:y_j=y_i} \sum_{h:y_h \neq y_i} \left[1 + \|\mathbf{W}(x_i - x_j)\|^2 - \|\mathbf{W}(x_i - x_h)\|^2\right]_+, \end{aligned} \tag{4.1}$$

where $\mu > 0$ is a balancing parameter between the two terms. The first term imposes a cost when the distances between the data points in the same class are large and the second term imposes a cost when the distances between the data points in different classes are smaller than the distances to data points within the class. Here, $[\cdot]_+$ is the

hinge loss defined as

$$[x]_+ = \begin{cases} x, \text{ if } x > 0, \\ 0, \text{ otherwise.} \end{cases} \tag{4.2}$$

To optimize $F(\mathbf{W})$, the transformation matrix $\mathbf{W}$ is parameterized as the Mahalanobis-based distance metric with a covariance matrix $\mathbf{C} = \mathbf{W}^\top \mathbf{W}$, so that Eq.(4.1) becomes

$$\begin{aligned} F(\mathbf{C}) = &\sum_{i=1}^{n} \sum_{j:y_j=y_i} \mathcal{M}(x_i, x_j) + \\ &\mu \sum_{i=1}^{n} \sum_{j:y_j=y_i} \sum_{h:y_h \neq y_i} \left[1 + \mathcal{M}(x_i, x_j) - \mathcal{M}(x_i, x_h)\right]_+ , \end{aligned} \tag{4.3}$$

where $\mathcal{M}(x_i, x_j) = (x_i - x_j)^\top \mathbf{C}(x_i - x_j)$. The cost function $F(\mathbf{C})$ can be optimized using semidefinite programming or subgradient descent [79].

4.4 Proposed method

While the conventional distance metric learning aims to learn a new distance metric through transformation of *feature vectors*, the goal of our proposed method is to learn an optimal *overall distance* through combination of multiple distance metrics. The motivation is to deal with the protein fold classification task where a number of protein structural dissimilarity measures are defined without feature vector representation in addition to dissimilarity measures defined in vector spaces with feature vectors representation. In the following, we reformulate Eq.(4.1) to work with distances instead of feature vectors.

Let the distance measure between two data points x_i and x_j computed using a concrete distance function with corresponding weight $\boldsymbol{w}$ be $d(x_i, x_j; \boldsymbol{w})$. The cost function in Eq.(4.1) is then rewritten as a cost function $J(\boldsymbol{w})$:

$$J(\boldsymbol{w}) = \sum_{i=1}^{n} \sum_{j:y_j=y_i} \left[d^2(x_i, x_j; \boldsymbol{w}) + \mu \sum_{h:y_h \neq y_i} [L(i, j, h; \boldsymbol{w})]_+ \right], \tag{4.4}$$

$$L(i, j, h; \boldsymbol{w}) = l + d^2(x_i, x_j; \boldsymbol{w}) - d^2(x_i, x_h; \boldsymbol{w}). \tag{4.5}$$

The same as in Eq.(4.1), $\mu > 0$ is a balancing parameter between the two terms. However, here the margin is set as a tuning parameter $l \geq 0$, while in Eq.(4.1) the margin is fixed to 1. In the original formulation of LMNN [79], the distance d was

parameterized as a Mahalanobis-based distance. On the other hand, we parameterize d as a convex combination of multiple distance measures.

4.4.1 Distance combination

As mentioned in the introduction, methods for comparing protein fold structures use either distances (dissimilarities) or similarity scores to evaluate the similarity between two proteins. Before combination of the distances or similarity scores, these values need to be normalized into a distance measure with range $0 \leq d_{ij} \leq 1$, by redefining $d_{ij} = D_{ij}/\max(\mathbf{D})$, where d_{ij} is the distance between x_i and x_j, and $\mathbf{D}$ is the $n \times n$ distance matrix for n training samples. Likewise, we can convert a similarity measure to a distance measure by letting $d_{ij} = 1 - (u_{ij}/\max(\mathbf{U}))$, where u_{ij} denotes the similarity value between x_i and x_j, and $\mathbf{U}$ is the $n \times n$ similarity matrix for n training samples.

The linear combination of S distance measures is written as follows:

$$d(x_i, x_j; \boldsymbol{w}) = \sum_{s=1}^{S} w_s d_s(x_i, x_j), \tag{4.6}$$

where $d_s(x_i, x_j)$ are the normalized distance measures between x_i and x_j as obtained from the multiple protein structure comparison methods. To ensure that the convexity and S-simplex constraints are satisfied, the weight for each distance measure $\boldsymbol{w} = (w_1, \ldots, w_S)^\top \in \mathbb{R}^S$ must satisfy

$$\sum_{s=1}^{S} w_s = 1, \;\; w_s \geq 0. \tag{4.7}$$

Assuming that each distance $d_s(x_i, x_j)$ satisfies the axiom of distance metric, i.e., non-negativity, coincidence axiom, symmetry, and triangle inequality, Eq.(4.6) also satisfies the axiom of distance metric.

4.4.2 Optimization algorithm

In the following, we formulate the optimization algorithm to obtain the optimal parameter $\boldsymbol{w}^*$. First, Eqs.(4.4) and (4.5) are rewritten as

$$J(\boldsymbol{w}) = \boldsymbol{w}^\top \mathbf{M} \boldsymbol{w} + \mu \sum_{i=1}^{n} \sum_{j:y_j=y_i} \sum_{h:y_h \neq y_i} [L(i, j, h; \boldsymbol{w})]_+ \tag{4.8}$$

$$L(i,j,h;\boldsymbol{w}) = l + \boldsymbol{w}^\top \boldsymbol{d}_{ij}\boldsymbol{d}_{ij}^\top \boldsymbol{w} - \boldsymbol{w}^\top \boldsymbol{d}_{ih}\boldsymbol{d}_{ih}^\top \boldsymbol{w}, \tag{4.9}$$

where

$$\mathbf{M} = \sum_{i=1}^{n} \sum_{j:y_j=y_i} \boldsymbol{d}_{ij}\boldsymbol{d}_{ij}^\top, \tag{4.10}$$

$$\boldsymbol{d}_{ij} = (d_1(x_i,x_j),\ldots,d_S(x_i,x_j))^\top \in \mathbb{R}^S, \tag{4.11}$$

$$d(x_i,x_j;\boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{d}_{ij}. \tag{4.12}$$

The first term in Eq.(4.8) is a smooth quadratic function that can be directly optimized. In contrast, the second term contains hinge loss $[x]_+$, which is not straightforward to optimize since it is not a smooth function. In the following, we first provide an iterative algorithm to optimize Eq.(4.8) and then present three approaches for solving the second term. The first approach optimizes hinge loss by using the subgradient algorithm. The second approach adds relaxation to the hinge loss through smoothing the hinge loss. Finally, the third approach replaces hinge loss with logistic loss.

4.4.2.1 Optimization algorithm

Algorithm 1 provides an iterative algorithm based on the gradient method for optimizing Eq.(4.8). In Algorithm 1, $\boldsymbol{g}_{ijh}$ is replaced with either $\boldsymbol{g}_{ijh}^{HL}$, $\boldsymbol{g}_{ijh}^{SHL}$, or $\boldsymbol{g}_{ijh}^{LL}$, which corresponds to the gradient of the second terms when using hinge loss (Chapter 4.4.2.2), smooth hinge loss (Chapter 4.4.2.3), or logistic loss (Chapter 4.4.2.4), respectively.

4.4.2.2 Hinge loss optimization

Hinge loss $[\cdot]_+$ in Eq.(4.8) is not differentiable at the origin, but it is possible to optimize the hinge loss by using the subgradient method. The subgradient of $[L(i,j,h;\boldsymbol{w})]_+$ is as follows:

$$\mathbb{R}^s \ni \boldsymbol{g}_{ijh}^{HL} = \begin{cases} 2(\boldsymbol{d}_{ij}\boldsymbol{d}_{ij}^\top - \boldsymbol{d}_{ih}\boldsymbol{d}_{ih}^\top)\boldsymbol{w}, \text{ if } z > 0, \\ 0, \text{ otherwise,} \end{cases} \tag{4.13}$$

where $z = L(i,j,h;\boldsymbol{w})$.

4.4.2.3 Smooth hinge loss optimization

Smooth hinge loss [89] adds an intermediate criterion to the hinge loss so that it is differentiable everywhere. Smooth hinge loss is defined as follows:

$$h_{smooth}(x) = \begin{cases} x - \frac{1}{2}, \text{ if } x \geq 1, \\ \frac{1}{2}x^2, \text{ if } 0 < x < 1, \\ 0, \text{ otherwise.} \end{cases} \tag{4.14}$$

By replacing $[L(i,j,h;\boldsymbol{w})]_+$ with $h_{smooth}(L(i,j,h;\boldsymbol{w}))$, the gradient becomes

$$\mathbb{R}^s \ni \boldsymbol{g}_{ijh}^{SHL} = \begin{cases} 2(\boldsymbol{d}_{ij}\boldsymbol{d}_{ij}^\top - \boldsymbol{d}_{ih}\boldsymbol{d}_{ih}^\top)\boldsymbol{w}, \text{ if } z \geq 1, \\ (2(\boldsymbol{d}_{ij}\boldsymbol{d}_{ij}^\top - \boldsymbol{d}_{ih}\boldsymbol{d}_{ih}^\top)\boldsymbol{w})z, \text{ if } 0 < z < 1, \\ 0, \text{ otherwise,} \end{cases} \tag{4.15}$$

where $z = L(i,j,h;\boldsymbol{w})$.

4.4.2.4 Logistic loss optimization

Another alternative to using hinge loss is to use logistic loss $\log(1+e^x)$, which is convex and differentiable everywhere. The cost function $J(\boldsymbol{w})$ in Eq.(4.8), approximated by logistic loss, is then written as follows:

$$\boldsymbol{w}^\top \mathbf{M}\boldsymbol{w} + \mu \sum_{i=1}^{n} \sum_{j:y_j=y_i} \sum_{h:y_h \neq y_i} \log(1+e^z), \tag{4.16}$$

where $z = L(i,j,h;\boldsymbol{w})$. The gradient of Eq.(4.16) is given by

$$\mathbb{R}^s \ni \boldsymbol{g}_{ijh}^{LL} = \frac{2e^{l+\boldsymbol{w}^\top(\boldsymbol{d}_{ij}\boldsymbol{d}_{ij}^\top - \boldsymbol{d}_{ih}\boldsymbol{d}_{ih}^\top)\boldsymbol{w}}(\boldsymbol{d}_{ij}\boldsymbol{d}_{ij}^\top - \boldsymbol{d}_{ih}\boldsymbol{d}_{ih}^\top)\boldsymbol{w}}{1 + e^{l+\boldsymbol{w}^\top(\boldsymbol{d}_{ij}\boldsymbol{d}_{ij}^\top - \boldsymbol{d}_{ih}\boldsymbol{d}_{ih}^\top)\boldsymbol{w}}}. \tag{4.17}$$

4.4.3 Flow of the classification framework

Figure 4.3 shows the flow of the classification framework.

Training phase:

Step 1: Given n training proteins and S methods that compute either similarities or

Algorithm 1 Gradient based Algorithm for Minimizing $J(\boldsymbol{w})$

Initialize: $\boldsymbol{w}_0 = (1/S, ..., 1/S)$, $\mu > 0$, $l > 0$, ϵ are set to small values, such as 10^{-5} or 10^{-6}, and $\mathbf{M}$ is as in Eq.(4.10).

repeat

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \epsilon \left\{ \mathbf{M}\boldsymbol{w} + \mu \sum_{i=1}^{n} \sum_{j:y_j=y_i} \sum_{h:y_h \neq y_i} \boldsymbol{g}_{ijh} \right\}$$

Simplex projection:

for $s = 1, \ldots, S$ **do**

if $w_s < 0$ **then** $w_s \leftarrow 0$

end for

for $s = 1, \ldots, S$ **do**

$w_s \leftarrow \frac{w_s}{\sum_{s=1}^{S} w_s}$

end for

until converges

Output: optimal coefficient $\boldsymbol{w}^* \leftarrow \boldsymbol{w}$

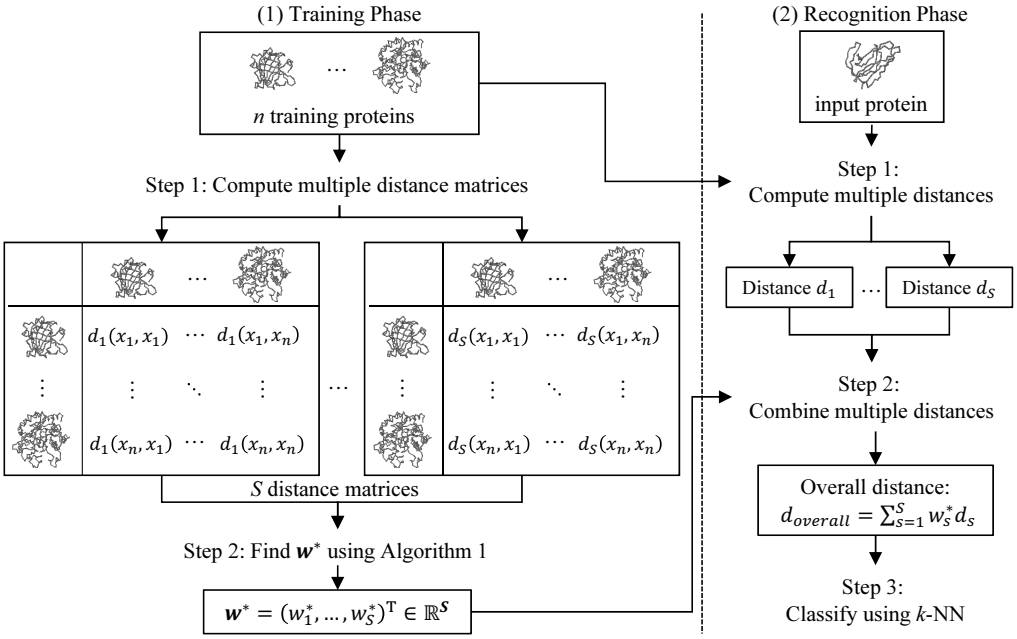

Figure 4.3: Flow of the classification framework.

distances, compute S $n \times n$ distance matrices in which each element is normalized ($0 \leq d_s(x_i, x_j) \leq 1$).

Step 2: Find the optimal $\boldsymbol{w}^* \in \mathbb{R}^S$ by using Algorithm 1.

Recognition phase:

Step 1: Given an input protein, compute S distances $(d_1, \ldots, d_S)$ between the input protein and each training protein and normalize the value similarly to that in the training phase.

Step 2: Compute the overall distance $d_{overall}$ by combining the S distances, using $\boldsymbol{w}^*$.

Step 3: Apply k-nearest neighbor to the overall distance $d_{overall}$ to predict the fold class of the input protein.

4.5 Experiments

To evaluate the validity of the proposed method, we conducted protein fold classification experiments using Ding Dubchak dataset [13, 14] and ENZYMES dataset [84], and compared the results with those from conventional methods based on averaging, voting, and GMKL [86]. The classification performance was evaluated in terms of precision (ratio of true positives to true positives and false positives) and recall (ratio of true positives to true positives and false negatives).

4.5.1 Ding Dubchak dataset

The Ding Dubchak dataset [13, 14] contains 27 fold classes of 693 proteins, as shown in Table 4.1. We conducted the classification experiments by combining distance metrics from the extracted features of the amino acid sequences and the 3D structural comparison methods. The original dataset does not contain the 3D structure of the proteins. Therefore, we downloaded the 3D structure for each protein from ASTRAL SCOP [12] for use by the 3D structural comparison methods. In the experiments, 18 types of similarity (or dissimilarity) metrics were used. Table 4.2 lists all distance metrics. We used 12 distances computed from 12 types of feature vectors extracted from the sequence of the amino acids [14] (D-1 to D-12), 2 dissimilarity measures from two 3D structural comparison methods (D-13 and D-17), and 4 similarity measures from four 3D structural comparison methods (D-14, D-15, D-16, and D-18). All the distance measures D1 to D-18 satisfy the axiom of a metric. Note that measures D-13 to D-18 lack of explicit feature vector representation. Since our main focus is to validate the effectiveness of the proposed method for metrics combination, we did not use any novel methods for computing the distances of the features from the sequences of the amino acids.

Table 4.1: 27-fold protein compositions used in the experiments.

No.	Fold class	# Proteins
1	Alpha; Globin-like	19
2	Alpha; Cytochrome c	16
3	DNA-binding 3-helical bundle	32
4	Alpha; Four-helical up-and-down bundle	15
5	Alpha; 4-helical cytokines	18
6	Alpha; EF-hand	15
7	Beta; Immunoglobulin-like beta-sandwich	74
8	Beta; Cupredoxins	21
9	Beta; Viral coat and capsid proteins	29
10	Beta; ConA-like lectins/glucanases	13
11	Beta; SH3-like barrel	16
12	Beta; OB-fold	32
13	Beta; beta-Trefoil	12
14	Beta; Trypsin-like serine proteases	13
15	Beta; Lipocalins	16
16	A/B; beta/alpha (TIM)-barrel	77
17	A/B; FAD (also NAD)-binding motif	23
18	A/B; Flavodoxin-like	24
19	A/B; NAD(P)-binding Rossmann-fold	40
20	A/B; P-loop nucleotide triphosphate hydrolases	22
21	A/B; Thioredoxin-like	17
22	A/B; Ribonuclease H-like motif	22
23	A/B; alpha/beta-Hydrolases	18
24	A/B; Periplasmic binding protein-like	15
25	A+B; beta-Grasp	15
26	A+B; Ferredoxin-like	40
27	Small; Small inhibitors, toxins, lectins	39
Total		693

4.5.1.1 Experimental setting

We conducted the experiments 10 times by repeatedly splitting the dataset into 50% training and 50% test data using stratified random sampling and used the same 10 set of training and test data for the conventional methods and the proposed methods. Since we compared the proposed method with GMKL [86], all metrics needed to be converted to kernel matrices, which are basically similarity matrices. All of the metrics are also needed to be converted to distance matrices for applying the proposed method. For GMKL, we converted all distances into kernel matrices by using a radial basis function

Table 4.2: List of similarity and dissimilarity metrics used in the experiments. Metrics D-1 to D-12 are based on the feature extraction of the protein sequences [13, 14]; D-13 to D-18 are similarity (or dissimilarity) measurements based on the 3D protein structures.

Name	Type	Features or methods
D-1	Euclidean distance	Amino acid composition
D-2	Euclidean distance	Predicted 2nd structure
D-3	Euclidean distance	Hydrophobicity
D-4	Euclidean distance	van der Waals Volume
D-5	Euclidean distance	Polarity
D-6	Euclidean distance	Polarizability
D-7	Euclidean distance	Pse Amino Acid-1
D-8	Euclidean distance	Pse Amino Acid-4
D-9	Euclidean distance	Pse Amino Acid-14
D-10	Euclidean distance	Pse Amino Acid-30
D-11	Euclidean distance	SW BLOSUM62
D-12	Euclidean distance	SW PAM50
D-13	RMSD dissimilarity	CE [23, 73]
D-14	Z-Score similarity	CE [23, 73]
D-15	Canonical angles similarity	CMSM (view-based)
D-16	FATCAT similarity	FATCAT [24, 73]
D-17	RMSD dissimilarity	TM-align [25]
D-18	TM-score similarity	TM-align [25]

$\exp(-d_{ij}^2/\sigma)$. The σ was tuned by simulated annealing, by using 5-fold cross validation of the training data. For the proposed method, to ensure the equivalence with those in GMKL, the obtained kernel matrices were converted back to distance matrices by using the method described in Chapter 4.4.1.

The parameters of the proposed method (margin l, balancing parameter μ, and k for k-NN) and GMKL were also tuned using 5-fold cross validation of the training data. For the proposed method, a number of balancing parameters and margins were then prepared, with values ranging from 0.01 to 1 in steps of 0.05. For GMKL, several step size limits for the backtracking line-search algorithm were set as follows: upper values: $0.1, 1.0, 2.1$; lower values: $0.1, 0.3, 0.9$. For the SVM used by GMKL, we also tuned the misclassification penalty, which was set to 0.5, 1, 1.5, and 2. We then applied the proposed method to all combinations of parameters with hinge loss, smooth hinge loss, and logistic loss optimization, and GMKL with L2 regularization. When using averaging, voting, and the proposed method, the k-nearest neighbor of the combined distances was used. When using GMKL, one-against-all SVM was used.

Table 4.3: Average precision (Prec.) and recall (Rec.) [%] for each baseline feature and method.

Metrics	D-1	D-2	D-3	D-4	D-5	D-6	D-7	D-8	D-9
Prec.	42.4	36.6	27.8	30.9	28.2	24.9	35.0	33.8	30.1
Rec.	38.2	37.4	24.9	25.5	25.1	21.8	32.1	29.0	23.9
Metrics	D-10	D-11	D-12	D-13	D-14	D-15	D-16	D-17	D-18
Prec.	27.8	51.4	48.1	86.6	92.7	75.3	92.4	83.3	98.8
Rec.	20.4	44.8	43.5	83.4	92.2	74.2	91.2	59.9	98.7

4.5.1.2 Experimental results

Firstly, as the baseline performance, classification experiments were conducted for each feature and method listed in Table 4.2 with k-nearest neighbor. Table 4.3 summarizes the experimental results for all the features based on amino acid sequences and methods based on the 3D structures. In the study of protein analysis, protein fold prediction using the features from amino acid sequences has been known to be a very difficult task. Consequently, the classification results when using D-1 to D-12 were poor, just as expected. In contrast, although protein fold classification using the 3D structure is also a challenging task, since the fold categories are determined by the 3D geometrical structure the performances when using D-13 to D-18 were much better than when using D-1 to D-12. From Table 4.3, we can also confirm that the scoring used in the protein structure comparison, either similarity or dissimilarity, could significantly affect the classification results. For example, when using CE [23, 73], the recall when using RMSD (D-13) was 83.4%, but when using the Z-Score (D-14) the recall improved to 92.2%. When using TM-align, the recall when using RMSD (D-17) was only 59.9%. In contrast, when using TM-score (D-18), the performance significantly improved to 98.7%.

Secondly, we used several combinations of the distance matrices to demonstrate both the effectiveness and limitations of the proposed method, and compared the results with naïve methods such as averaging, voting, and GMKL [86] as the representative method from multiple kernel learning based algorithm. We did not compare our method with the conventional feature-based distance metric learning methods [77, 79, 80, 81, 82, 83**?**] because of the unavailability of the explicit feature vectors. Tables 4.4 and 4.5 show the combinations of distance matrices and the classification results for each method, respectively. In Table 4.5, HL, SHL, and LL are the proposed methods using hinge loss, smooth hinge loss, and logistic loss, respectively. In the following, we discuss the performance of each method shown in Table 4.5.

Table 4.4: Combinations of distance matrices.

Comb.	D-1 to D-12	3D structure based					
		D-13	D-14	D-15	D-16	D-17	D-18
C-1	✓						
C-2	✓						✓
C-3	✓	✓	✓	✓	✓	✓	✓
C-4		✓	✓	✓	✓	✓	✓
C-5	✓	✓					
C-6	✓	✓	✓				
C-7	✓	✓	✓	✓			
C-8	✓	✓	✓	✓	✓		
C-9	✓	✓	✓	✓	✓	✓	

When using averaging, which is basically the same as using a uniform weight (each entry of $\boldsymbol{w}$ is $1/S$, where S is the number of the distances) to combine the distances, the performance was significantly affected by the number of good distance metrics. For example, when all the distance metrics were from the 3D structure-based method (combination C-4), the averaging method achieved high precision (97.0%) and recall (96.4%). However, when the metrics from both amino acid sequence and 3D structure based methods were combined, averaging performed poorly compared with the other methods.

The voting-based method uses the most votes among the distance metrics to determine the fold category of an input protein. The overall performance of the voting-based method was better than that of averaging, but voting also requires a majority of the distance metrics to have good accuracy. The performance of the voting method was better than averaging when multiple distance metrics from 3D structure-based methods were included (combinations C-3).

Multiple kernel learning is known to be the most widely used method for combination of metrics. When GMKL [86] was used, the classification results improved significantly compared to results with the averaging and voting. For combination C-1, GMKL outperformed the other methods. However, the results for the other combinations were worse than with the proposed method. For comparison purpose, Table 4.6 shows the experimental results using original LMNN, SVM, and random forest (RF) for combination C-1 where feature representations were available. The combination was done by either voting or concatenation of the feature vectors. In voting-based method, the LMNN, SVM, or RF was first applied to each type of the features in the combi-

Table 4.5: Average classification results in term of precision and recall [%] for combinations of distance matrices. HL, SHL, and LL stand for hinge loss, smooth hinge loss, and logistic loss, respectively.

	Precision					
	Averaging	Voting	GMKL [86]	Ours		
				HL	SHL	LL
C-1	52.3	58.0	**58.0**	51.0	51.5	50.6
C-2	63.8	67.7	86.6	**99.0**	98.7	98.1
C-3	89.4	91.4	97.6	**98.8**	98.5	97.8
C-4	97.0	97.0	92.5	**98.8**	98.4	97.3
C-5	65.8	66.1	86.0	90.1	90.2	**90.4**
C-6	75.5	75.4	93.1	95.4	**95.9**	95.7
C-7	76.9	78.5	92.5	96.0	**96.1**	95.9
C-8	79.4	86.1	93.4	**96.3**	**96.3**	96.0
C-9	85.8	87.9	96.5	**96.8**	96.5	96.4
	Recall					
	Averaging	Voting	GMKL [86]	Ours		
				HL	SHL	LL
C-1	41.3	50.3	**54.8**	45.6	45.9	42.2
C-2	54.0	59.5	84.4	**98.9**	98.6	98.0
C-3	87.0	88.3	97.4	**98.7**	98.3	97.5
C-4	96.4	96.5	92.3	**98.7**	98.2	96.8
C-5	55.2	58.3	84.4	89.2	89.2	**89.5**
C-6	67.1	68.3	92.3	95.0	**95.6**	95.4
C-7	69.5	71.5	91.7	95.7	**95.8**	95.6
C-8	72.5	80.6	92.8	**96.0**	**96.0**	95.7
C-9	81.6	82.9	96.2	**96.4**	95.9	95.9

Table 4.6: Average precision and recall [%] for combination C-1 using LMNN, SVM, and random forest (RF).

	Precision		Recall	
	Voting	Concatenation	Voting	Concatenation
LMNN	61.78	56.25	53.04	52.50
SVM	63.74	59.66	32.18	38.50
RF	63.86	63.48	46.18	52.41

nation C-1 and then the final classification results were decided through voting among the 12 distances. In concatenation method, all feature vectors were first concatenated. Then, the LMNN, SVM, or RF was applied to the concatenated features. For SVM, we

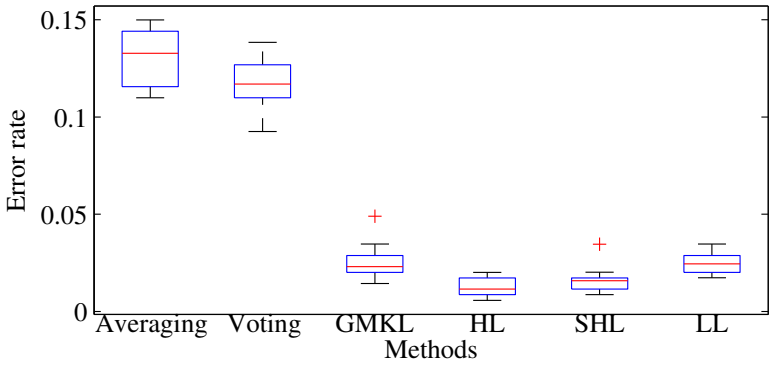

Figure 4.4: Error rates for combination C-3.

tested using linear, quadratic, and third degree polynomial kernel, where we reported the best results which were produced with quadratic kernel. For the RF, the number of trees was empirically set to 1000. When using voting-based method, all the methods obtained higher precisions than that of GMKL, where the highest precision of 63.86% was obtained by using random forest. With concatenation of the features before applying each method (LMNN, SVM, and RF), the precisions decreased. However, in term of recall, the performance of GMKL was still better than the performance of LMNN, SVM, and RF with either voting-based method or concatenation of the feature vectors.

The proposed method was used together with the three types of loss functions described in Chapter 4.4.2. The overall performances for the three loss functions were comparatively similar to one other. Although LMNN-based methods did not perform well for combination C-1, the precision and recall for the proposed method with smooth hinge loss (SHL) were slightly better than the best results from the single measure (D-11). In the case when the combination consists of distances from both amino acid sequence features and 3D structure-based methods, the proposed method outperformed averaging, voting, and GMKL. This suggests that when distance metrics are very different in terms of performance, our method is still able to find the optimal weights for combination. In combination C-2, where there were 12 poor distances with one very good distance measure, the proposed method with hinge loss could still achieve precision and recall comparable to the single measure of D-18.

Figure 4.4 shows the boxplot of the error rate for combination C-3. Table 4.7 shows the p-value of the t-test for the 10 repeated experiments using the proposed method with combination C-3, GMKL, and the single measure D-18. From the Table 4.7, we can conclude that with more than 95% confidence ($p = 0.0005$) the proposed method

Table 4.7: The p-value of the t-test for single measure D-18 and combination C-3.

	D-18	GMKL	HL	SHL	LL
D-18	-	0.0046	0.7351	0.2693	0.0007
GMKL	0.0046	-	0.0005	0.0265	0.6470
HL	0.7351	0.0005	-	0.0661	0.0002
SHL	0.2693	0.0265	0.0661	-	0.0081
LL	0.0007	0.6470	0.0002	0.0081	-

using hinge loss performed better than that of the GMKL. The t-test results show that the combination of all 18 distances using the proposed method with hinge loss and smooth hinge loss performed the same as the single measure D-18. This points to the capability of the proposed method in finding an optimal combination when combining heterogeneous metrics to eliminate the necessity for *preselecting* the distance metrics.

4.5.1.3 Computational time and further discussion

Table 4.8 compares the average computational time and iteration required for the proposed method and GMKL to complete the optimization for combination C-3. The experiments were conducted using Intel Xeon E5-2630 2.3 Ghz with 32 GB RAM. The proposed methods were implemented using Matlab, while the computation of the cost function and gradient for smooth hinge loss and logistic loss were implemented using C. When using the proposed method with hinge loss optimization, one experiment can be completed in a relatively short time, this is because the computation of the subgradient is very simple. With smooth hinge loss, more iterations were required before convergence, which points to the longer computational time. Logistic loss required the least iterations to converge. However, since it requires to compute exponential function in each iteration, the average computational time was longer than that of the hinge loss. For GMKL, we used the publicly available Matlab code from [86] that uses the quadratic programming function from Matlab optimization toolbox. Average computational time of GMKL was 628.63 seconds. GMKL was basically fast, except for the case when convergence solution could not be obtained. In that case, we used active-set method in quadratic programming, which can deal with non-convex problem with the trade-off in computational time.

Next, we conducted 10 repeated experiments for combination C-1, C-2, C-3, and C-4 by using hinge loss with random initial weights instead of equal weights. For combination C-1, the average precision and recall were 50.6% and 45.3%, respectively. For

Table 4.8: Average computational time and iterations for the proposed methods and GMKL for combination C-3.

	HL	SHL	LL	GMKL
Time (in secs)	11.05	148.88	133.59	628.63
Iteration	80	131	60	-

Table 4.9: Average of optimal weight coefficients [%] for combination C-3, obtained by hinge loss (HL), smooth hinge loss (SHL), and logistic loss (LL) over the 10 repeated experiments.

Distances	HL	SHL	LL
D-1	3.06 ± 0.11	0.89 ± 0.03	4.20 ± 0.00
D-2	1.61 ± 0.04	0.77 ± 0.03	2.37 ± 0.00
D-3	0.41 ± 0.01	0.27 ± 0.01	2.56 ± 0.00
D-4	0.17 ± 0.00	0.24 ± 0.01	2.42 ± 0.00
D-5	0.00	0.24 ± 0.01	2.55 ± 0.00
D-6	0.00	0.26 ± 0.01	2.01 ± 0.00
D-7	0.10 ± 0.00	0.23 ± 0.00	2.86 ± 0.00
D-8	0.05 ± 0.00	0.21 ± 0.00	2.30 ± 0.00
D-9	0.00	0.18 ± 0.00	1.07 ± 0.00
D-10	0.00	0.00	2.20 ± 0.01
D-11	0.45 ± 0.01	0.37 ± 0.01	0.52 ± 0.00
D-12	0.00	0.28 ± 0.01	0.19 ± 0.00
D-13	3.03 ± 0.06	4.88 ± 0.25	17.63 ± 0.02
D-14	3.78 ± 0.22	4.63 ± 0.38	16.22 ± 0.01
D-15	18.92 ± 2.70	24.14 ± 4.75	8.16 ± 0.02
D-16	0.32 ± 0.01	0.48 ± 0.02	3.85 ± 0.00
D-17	4.39 ± 0.24	4.10 ± 0.28	13.06 ± 0.01
D-18	63.72 ± 5.45	57.83 ± 7.97	15.85 ± 0.01

combination C-2, the average precision and recall were 99.0% and 98.9%. For combination C-3, the average precision and recall were 98.7% and 98.6%, respectively. For combination C-4, the average precision and recall were 98.8% and 98.7%, respectively. By comparing these results with those in Table 4.5, we can see that the proposed method is not largely affected by the initial weights.

Finally, we discuss the characteristics of the hinge loss, smooth hinge loss, and logistic loss. Figure 4.5 shows how the costs imposed by the hinge loss were smoothened by smooth hinge loss and logistic loss. Table 4.9 shows the optimal weight coefficients when combining all distance metrics (combination C-3) using each loss function. Hinge

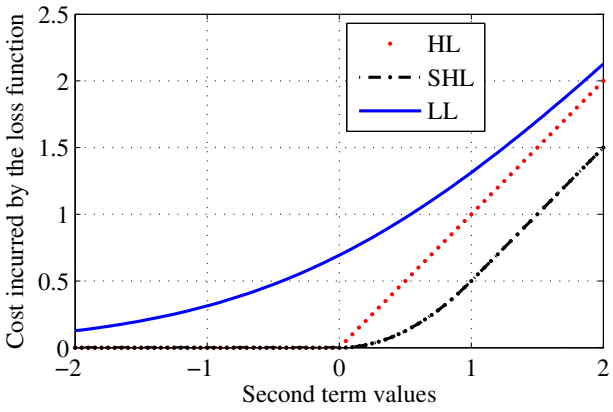

Figure 4.5: Costs imposed by hinge loss (HL), smooth hinge loss (SHL), and logistic loss (LL).

loss tends to sparsely pick up the metrics, where the majority of the weights were given to D-18 (63.72%). Although D-15 performed worse than other 3D structure based method (see Table 4.3), the average weight for D-15 was 18.92%, which were higher than the other distances. This suggests that the proposed method can *selectively adjust the weight* for the most optimal combination. When using hinge loss, there were five distances with zero weights (D-5, D-6, D-9, D-10, and D-12). Smooth hinge loss also put most of the weights to D-18 (57.83%). However, smooth hinge loss considered more distance metrics than hinge loss whereas only D-10 had zero weight. When the cost function used logistic loss, all of the distance metrics were used. This suggests that using smooth loss functions, such as smooth hinge loss or logistic loss, the weights were more uniformly distributed over the available distance metrics.

4.5.2 ENZYMES dataset

In this experiment, we conducted classification experiments on ENZYMES dataset that contains six classes of protein enzymes (100 proteins in each class), where a graph-based representation is used to represent each protein [84]. Since it is not trivial to combine graph-based representation, conventional feature combinations and metric learning cannot be used. We combined three types of graph-based kernel matrices: the Weisfeiler-Lehman based subtree kernel [90], propagation based graph kernel [91], and GraphHopper kernel [92]. We conducted three repeated experiments by randomly selecting 30% of samples in each class as training and the rest as testing. When using the proposed method, the kernel matrices were converted into distance matrices as in

Table 4.10: Summary of the classification results in term of precision and recall [%] for ENZYMES dataset.

Method	Precision	Recall
Single metric (k-NN):		
- GraphHopper [90]	55.3	54.4
- Propagation [91]	19.6	18.3
- Weisfeiler-Lehman [92]	31.8	27.5
Combination:		
- Averaging (k-NN)	53.2	46.9
- Voting (k-NN)	52.7	31.7
- GMKL (SVM)	55.6	50.1
Ours (fixed step size, 1-NN):		
- Hinge loss	55.3	54.4
- Smooth hinge loss	55.3	54.4
- Logistic loss	53.7	52.7
Ours (with Armijo rule, 1-NN):		
- Hinge loss	54.2	53.4
- Smooth hinge loss	**55.8**	**54.9**
- Logistic loss	53.1	52.4

the previous experiments with the Ding Dubchak dataset and fixed value of $k = 1$ for the k-NN was used. When using GMKL, the kernel matrices were directly used. The parameters for the proposed methods and the GMKL were tuned in the same manner as in the previous experiments. In addition to the use of the fixed value for step size in the proposed methods ($\epsilon = 10^{-5}$), we also used an adaptive step size by using Armijo rule [93]. We reported the experimental results based on the average of the three repeated experiments.

4.5.2.1 Experimental results

Table 4.10 summarizes the experimental results. By using k-NN, the classification results using each kernel were relatively low. With averaging and voting, the performance were worse than those with the GraphHopper kernel which has 55.3% precision and 54.4% recall. When using GMKL, the precision was slightly better than that of GraphHopper, while the recall was not as bad as the averaging and voting. When using the proposed method with fixed step size, the hinge loss and smooth hinge loss put weight only to GraphHopper kernel, resulting the same performance as GraphHopper kernel. With Armijo rule, the proposed method with smooth hinge loss could slightly improve

the performance of the GraphHopper kernel, where the precision and recall were improved to 55.8% and 54.9%, respectively. These results suggest that the proposed method can still effectively deal with distance metrics that have poor performance.

4.6 Summary

We proposed a method that finds optimal combinations of distance metrics for protein fold classification task by generalizing the concept of LMNN for combining multiple distance metrics. LMNN was originally proposed for learning Mahalanobis-based distance metric from a set of feature vectors of sample data. On the other hand, our objective is to find an optimal combination of distance metrics from multiple protein structural measurement methods, where natural feature vector representation is not always available. The final distance, termed as *overall distance*, is then defined as a linear combination of the multiple distance measures. The convex optimization problem is solved by using an iterative algorithm based on the subgradient and gradient methods; for this we adopted three types of loss functions: hinge loss, smooth hinge loss, and logistic loss. The effectiveness of the proposed method was demonstrated through classification experiments using the Ding Dubchak dataset and ENZYMES dataset, where the proposed method outperformed the naive conventional methods and GMKL. In particular, the proposed method could effectively find an optimal combination for distance metrics with very different performance. Thus, we can avoid the difficulties in *pre-selecting* distance metrics among a number of available distance metrics. However, an analyst should make an effort to make a list of possible distance measures before applying the proposed method. If there is no single good measure is included in the list of the metrics, the proposed method may not work effectively. In contrast, it is not a big problem if many poor metrics exist in the list because the proposed method can construct an optimal combination of poor metrics (poorly performed comparison methods) and good metrics.

To further improve the performance of the proposed method, imposing a locality neighborhood constraint by considering only the instances in a close region [79] is one direction of our future works. To impose explicit locality, another direction is by imposing different weights for different region ranges, as in the local metric learning [94], to our problem formulation. Since the proposed method is technically generic, we will also consider applying the proposed method to different classification tasks other than the protein fold classification problem.

Chapter 5

Concluding Remarks

In this chapter, we provide the summary of the work presented in this thesis with regard to the goal and the research direction in the future.

5.1 Summary

In this research, as a general tool for the protein structure analysis, we address the limitation of the conventional methods on 3D protein structural comparison. While protein structural analysis is a broad subject, we limit the scope by focusing on classification task as the test case.

In Chapter 3, we explored the possibility of using image set based method for measuring the similarity between the protein structures, by utilizing the 3D visualization molecular software. Our motivation comes from the fact that the 3D visualization molecular software is commonly used to manually inspect the 3D structure of proteins. However, to our best knowledge, our work is the first that demonstrate the usefulness of using the set of protein visualization images in automatic protein structure classification. Our proposed method, termed as *view-based* method, addresses the difficulties of the alignment method in computing the similarity between the very different structures. By using the proposed method, it is also possible to use of multiple visualizations of the protein structure to enrich the protein descriptor. Moreover, since the set of the visualization images is modeled by a subspace, it is straightforward to adapt any subspace learning algorithm in the classification framework. The validity of the proposed method was demonstrated through the experimental results in classification of seven protein classes, especially where the structures are very different.

Various similarity and dissimilarity measures for comparing protein structures have

been proposed. However, it is difficult to determine which method is the best among them for general classification task. In Chapter 4, we generalized the concept of the large margin nearest neighbor (LMNN) for obtaining an optimal distance metrics combination, termed as *overall distance*. The main advantage of the proposed method is the capability on finding an optimal combination even with the inclusion of comparison methods that do not perform well in the combination. The validity of the proposed method was demonstrated through the experimental results in classification of 27 protein folds and six ENZYME classes.

5.2 Future work

Although the experimental results shows that the proposed methods in Chapters 3 and 4 outperformed conventional methods in the classification task, there are many work to be done in the future.

In Chapter 3, we used uniformly fixed distributed viewpoints with the central viewing axis of the protein structure as the initialization for generation of the multiple-view images. This approach works well to capture the overall structure of the protein and to differentiate protein at coarse level. In the future, we will consider applying alignment as the preprocessing prior to the generation of the multiple-view images and consider adaptive number of viewpoints. In generating the visualization images, we used a third party software Jmol [54] which is not specifically designed for our purpose. Developing our own visualization software can further speed up the computation of the proposed method. Exploring various computer graphics parameters such as illumination and color is also an important direction for improving the proposed method. We will also consider to apply the proposed method in the pipeline of protein-protein docking framework.

In Chapter 4, we used straightforward weighting scheme. In the future, we will consider to add adaptive weights, i.e., by imposing different weights for different neighborhood ranges [81], or add neighborhood constraint such as introduced in [79]. We will also consider to apply the proposed method for classification task other than the protein structure classification.

Appendix A: List of Proteins in the Experiments

This appendix provides the list of the SCOP ID of the 700 proteins in sequential order used in the experimental section of Chapter 3.3. In the classification experiments, we divided the dataset into 10 subsets of testing data to conduct 10 repeated experiments: in the first experiment, we used the first 10 proteins from each class as test data and the rest for training; in the second experiment, we used the 11th to 20th proteins from each class as test data, and the rest for training; and so on.

To compute the average computation time for 100 concurrent requests, we used 100 proteins from the Alpha (α) and Beta (β) proteins. In computing the average computation time for each of 10 consecutive requests (only one process at a time), we used the first 10 proteins from the Alpha (α) and Beta (β) proteins.

The protein stored in SCOP database has three types of identifiers as follows:

1. *sccs* (SCOP concise classification string). Sccs is a dot notation with format of [class.fold.superfamily.family] to describe the hierarchical category of the protein. The identifier for a class are in alphabet, while the rest are in numeric. For example, protein with sccs of a.2.6.3 represents class "a" which is α, fold number 2 in class α which is a fold of *Long alpha-hairpin*, superfamily number 6 which is *tRNA-binding arm*, family number 3 which is *Valyl-tRNA synthetase (ValRS) C-terminal domain*.

2. *sunid*. Sunid is a unique identifier in SCOP in the form of a number for any entry in the SCOP.

3. *sid*. Sid is a stable domain identifier with format of seven characters, as follows: character "d", followed by four character of PDB ID (original protein identifier stored in PDB), followed by the protein chain in the form of alphanumeric ("_" if no chain; "." if multiple chains), and finally is a character that specified the

segmented part (domain) of the protein structure ("_" if contains a whole structure).

In the following, the list of the proteins for each class is listed using sid identifiers.

Alpha (α) proteins

d3l0fa_ d2iy5a1 d2f6ma_ d1z0jb1 d1e29a_ d3k2aa_ d1gv2a2 d2cqra1 d2iw5b1 d1ofcx1 d1hlva1 d2gfna1 d1rr7a_ d2coba1 d1r1ta_ d1d8ka_ d1ka8a_ d1t0fa1 d1xb4a1 d1u5tb1 d2obpa1 d1z96a1 d1vdla_ d1h7ca_ d4a5xa_ d2wkxa1 d1jfia_ d1h6ga2 d1dova_ d2fzfa1 d1syya_ d1d9ca_ d1ivsa2 d1x91a_ d2fefa1 d3dbya2 d1nh2d1 d2icta_ d1wlza1 d1qasa1 d1p5sa_ d1irqa_ d4dnda_ d1s2xa_ d2b4jc1 d1tuka_ d3h8ja2 d1a0pa1 d2o6ka1 d1jvra_ d1l9la_ d1fkma2 d2i53a1 d2a5yb2 d1dgna_ d1q8ca_ d1hc1a1 d1dbha1 d1q4ga1 d1lj8a3 d2i76a1 d1kwfa_ d2wy8a_ d2g0da_ d1iiea_ d1vnsa_ d1h6ka1 d4b8ja_ d2onda1 d1j1ja_ d1wy6a_ d2ouxa1 d2fx0a2 d1fcya_ d1fpsa_ d3vj8a_ d3hlxa_ d1ef1c_ d1j0pa_ d1m1qa_ d1kcfa1 d1wija_ d1i2ta_ d1u61a_ d1ztda1 d2q0ta1 d1ov9a_ d1iyjb2 d1vfga1 d2ozbb1 d1w53a_ d1us7b_ d1n93x_ d1ojha_ d1sj7a1 d1sj8a2 d1q8da_ d2choa1 d1ykha1 d2p6va1

Beta (β) proteins

d2giya1 d1l6za1 d1nezg_ d1l6za2 d1biha3 d1olza1 d4hbqa1 d2yxma_ d3bn3b2 d1bqua2 d1i1ra1 d3csba1 d1cd9b2 d2hyma2 d1cfba1 d1cwva2 d1my7a_ d1h3ga1 d2fhfa1 d2fhfa2 d1ex0a1 d2d7na1 d3ugua1 d2omza1 d1z0na1 d2q0zx2 d2nqda_ d1e5ba_ d3zuca_ d1j8ra_ d2okma_ d2j43a2 d3ef4a_ d1rlwa_ d1f53a_ d1wgva_ d2iqya_ d1od3a_ d1pmhx_ d1gwma_ d3ku3a_ d1flca1 d1dmza_ d1cq3a_ d1d2sa_ d1kqra_ d2uwaa_ d1jova_ d1zy9a1 d3tcqa2 d2xiwa_ d1r4ka_ d1vbva1 d1gu7a1 d1o89a1 d2i6va1 d2q3ga_ d3vqfa_ d2ylba_ d1prtf_ d3tssa1 d1o7ia_ d3kdfb_ d1ueba3 d1b3qa2 d1xhba1 d4ac9a2 d1g7sa1 d1kzla2 d2vbua_ d3mmga_ d2fug31 d1foea2 d1t77a2 d1lf7a_ d4iaxa_ d3kffa_ d2f09a1 d3cu9a_ d1gxra_ d2f2ha3 d1rk8c_ d4g3qa2 d1qwra_ d1vj2a_ d1sefa_ d3s4ya2 d1v8qa_ d1tula_ d2pp6a1 d2c3fa1 d1udxa1 d1pgl11 d2b78a1 d2anea1 d2b97a_ d2q6ka2 d1wida_ d2geca1 d2ed6a1

Alpha and beta (α/β) proteins

d3vzxa_ d2fhfa5 d3hn3a3 d1x38a1 d1gqia1 d1x7fa2 d2fiqa1 d2p10a1 d1xa0a2 d1qora2 d1wmaa1 d2h7ma_ d3cina1 d2f1ka2 d1kyqa1 d1pjqa1 d2ivda1 d2gqwa2 d1w4xa2 d2i5ia1 d1h16a_ d3vbaa_ d3osxa_ d2d0oa1 d1vq8y1 d1io0a_ d1n8yc2 d3h0sa2 d1j3aa_ d1vq8c1 d3rqia_ d2z98a_ d4fgla_ d2y71a_ d4au1a_ d1sura_ d2r8oa1 d2iyva_ d1a7ja_ d2akab1 d1e9ra_ d1cr1a_ d1g6oa_ d1yksa2 d2fwra1 d1r6bx3 d1jqlb_ d1u0ja_ d1rifa_ d1t1ua1

d1i9sa_ d1d5ra2 d1yt8a1 d1lu4a_ d2g50a3 d1vhua_ d1w94a1 d2hqsa2 d1ckqa_ d1na6a2 d1wtea_ d1dzfa1 d3beda1 d2fxua1 d2v3za1 d3hl8a_ d1wlja_ d2vgna2 d1o13a_ d3fuca_ d1m4la_ d2gfqa1 d1nd6a_ d1g60a_ d2f8la1 d2gtia1 d1lc5a_ d2cb9a_ d2ocga_ d2absa1 d2v9la_ d2phpa1 d4atya_ d3ml1a1 d2bfwa1 d2iw1a_ d2olra1 d1ko7a2 d1dp4a_ d2pw9a1 d1u02a_ d2ghta_ d2o2xa1 d2obba1 d3k7pa_ d1vl1a_ d2nx2a1 d3lnla_ d2fcja1 d2g5gx1

Alpha and beta ($\alpha + \beta$) proteins

d3hzsa_ d4hlsa_ d2je6a1 d1ueka1 d1k8rb_ d1j0ga_ d1l5pa_ d3fila_ d1tkea1 d2fug13 d2bo9b2 d2oqea2 d1v58a2 d1idpa_ d1oh0a_ d3blza1 d3g8za_ d2a4da1 d1t11a3 d2f23a2 d1ll7a2 d3rmua_ d2hnga1 d4ffua_ d3rjsa_ d2dmya1 d4htga2 d1q8ba_ d1s7ia_ d2gvka1 d3znua_ d2gffa_ d1nh8a2 d2vona2 d2bopa_ d1tdja2 d1zvpa2 d2f06a2 d1qd1a1 d1zj8a1 d1in0a1 d1q8ka2 d1tiga_ d2zfza_ d1l0wa2 d1dt9a2 d1t3ta5 d1vk3a2 d1ydwa2 d1ewfa1 d1ztpa1 d3dk9a3 d2choa3 d2izva2 d2hbaa1 d1e3ha5 d1seta2 d2p0la1 d2fh1a2 d4kefa_ d4espa_ d2fh5a1 d1l3la2 d1p0za_ d2p7ja2 d2b06a1 d3c9fa1 d2hkja3 d2ggca_ d1b6aa2 d1kbla3 d2r85a2 d2ybxa_ d2cmwa_ d2jc5a_ d1p9ea_ d1wraa1 d2p97a1 d3dsda_ d1qmea3 d1jb0d_ d1tt8a_ d4gf3a_ d1k8kf_ d2od0a1 d1lg7a_ d2d7va1 d1go3e2 d1nr3a_ d1ss6a_ d1vlya2 d1v33a_ d3hwua_ d1wmia1 d2a6qa1 d2dira1 d2ghvc1 d3m7va2 d1zd0a1 d3c8wa1

Multi-domain proteins (alpha and beta)

d2v95a_ d2i06a_ d2hdsa_ d1ci9a_ d1vqqa3 d2bg1a1 d1qmea4 d2p74a_ d1yqsa_ d1u60a_ d4k0wa_ d1gwea_ d1u5ua_ d1w07a3 d2wbia1 d3swoa1 d3mkha1 d1ka1a_ d1lbva_ d1inpa_ d2bjia_ d3ncia2 d2py5a2 d4dqqa2 d4g1qa1 d3fsia_ d1mswd_ d1muka_ d1u09a_ d3bsoa_ d1uvja_ d1jx4a2 d1jiha2 d1t94a2 d2yi9a_ d3pwta_ d1bjta_ d4elya_ d3ilwa_ d1d3ya_ d1dd9a_ d1nuia1 d2eiya_ d1wuil_ d2fug41 d1wuis_ d1sg6a_ d1o2da_ d3uhja_ d1pg4a_ d2d1sa_ d3o83a_ d1i2aa_ d1oa0a_ d1su8a_ d1h5wa_ d1uf2a_ d1twfb_ d4g7hc_ d1twfa_ d1io1a_ d1kyqa2 d1pjqa3 d1gqea_ d1lfpa_ d1ldja3 d1k8ta_ d1m1ca_ d3aa0a_ d1szqa_ d1nh1a_ d4jbua_ d1q88a_ d3vdpa_ d1xfia_ d2g8la1 d1uwka_ d1z0sa_ d2bona1 d1vkya_ d1xqba_ d1u7la_ d1urja_ d1zjca1 d1zbpa1 d2hq2a1 d2hqva1 d2azea1 d2azeb1 d2avue1 d2p62a1 d2oeza1 d3er9b_ d3rlff1 d2i5ha1 d2i71a1 d2o3ia1 d2i0za2 d4dyna_ d2guma1

Membrane and cell surface proteins and peptides

d1cola_ d3bl2a_ d1q59a_ d1ddba_ d2vofa_ d1nkza_ d1nkzb_ d1lghb_ d1qjpa_ d3gp6a_ d1ys5a1 d3arco_ d2zfga_ d2pora_ d2fgqx_ d1by5a_ d2vdfa_ d2x55a_ d1tlya_ d1ek9a_ d3pika_ d1uuna_ d1lsha2 d1lsha3 d1lshb_ d3c0na2 d3zjxa_ d3n40f1 d1m0ka_ d4j9zb_ d2oara1 d1c99a_ d3s8gb1 d1fftb2 d3ag3b1 d4i0ua2 d1kqfc_ d1ppjc2 d2bs2c_ d1kf6c_

d2wdqc_ d2wdqd_ d1y5ic1 d2qi9a_ d3ag3g_ d3ag3j_ d3ag3k_ d3ag3l_ d3ag3m_ d1m56d_ d3s8gc_ d2wjnh1 d1ppje2 d2e74d2 d1ppjg_ d1ppjj_ d1jb0f_ d1jb0i_ d1jb0x_ d1l2pa_ d1kqfb2 d4i7zc3 d1q90l_ d1q90m_ d4i7zf_ d4i7zh_ d1rh5b_ d1rkla_ d3arcl_ d3arch_ d3arcm_ d3arce_ d3arcf_ d3ag3c_ d1ppjf_ d1jb0l_ d1ppjc1 d3n5ka1 d2vv5a3 d1iwga7 d1iwga8 d1oeda_ d1pw4a_ d2cfqa_ d1rh5a_ d1rwta_ d1u7ga_ d2clyc_ d1slqa_ d2nwwa1 d1r5sa_ d2clya1 d2clyb1 d3arcc_ d2uuia1 d4al0a_ d2zy9a3 d3rlff2 d3d31c1 d2qfia2

Small proteins

d1viba_ d1ss3a_ d1lu0a_ d1fu3a_ d1axha_ d1lupa_ d1d1ha_ d1dl0a_ d1qk7a_ d1agga_ d2sn3a_ d1acwa_ d1fjna_ d1lpba2 d1q4ga2 d1kloa1 d1lr7a1 d1b9wa1 d1deca_ d2pw8i_ d1oiga_ d3tvji_ d1tocr2 d1ut3a_ d1zuea1 d2nlsa_ d2tgia_ d1hcna_ d1quba3 d2o39c1 d1elva2 d1ok3a1 d2ok5a4 d2z3qb1 d2h9ec_ d1sg1x4 d1exta1 d2heyr2 d2vgha_ d1e88a3 d1tpga2 d1u5ma_ d1iyca_ d2hipa_ d1isua_ d3a38a_ d1bboa1 d1bhia_ d1tf3a3 d1x6ha1 d1x6ha2 d1rmda1 d1znfa_ d1a1ia1 d1x3ca1 d1wjpa1 d2csha2 d2ctda1 d1x5wa1 d2ghfa3 d2en2a1 d1wjva1 d2vrda1 d1wira_ d2vy4a1 d1zmec1 d3g9ma_ d1ibia1 d1x68a1 d1x62a1 d2dloa2 d1xpaa2 d2d8ra1 d2vuti_ d1dsva_ d1a6bb_ d1akya2 d1twfi2 d1yuza2 d1vzia2 d1gh9a_ d2k4xa1 d2j9ub1 d1n0za_ d2apob_ d2jnea1 d2zjr41 d1weoa_ d2g45a_ d1rjuv_ d1eyfa_ d3ueja_ d1z60a1 d1weea_ d2vnfa_ d1f81a_ d1lpva_ d1t50a_ d1oc0b_ d2dkta1

Appendix B: Documentation of View-based Protein Comparison (VPC) System

We provided VPC System including the source code for off-line usage. This appendix provides the details of the architecture of VPC, system manual for installation and maintenance, HTTP API interface manual, and a closing note for the VPC system. There are technical terms related to web-based technology and operating system in this document. The detail description for each technical term is beyond the scope of this appendix.

B.1 Architecture

As described in Chapter 3.5, VPC system consists of multiple modules which can be divided into two parts: front-end and back-end. The front-end handles the incoming and outgoing data from and to end-users. The back-end handles all the required computation. This section describes the detailed framework of both the front-end and the back-end.

Front-end

The front-end module of VPC uses Apache web server and PHP server-side scripting. The main function of the front-end is as the interface for end-users to submit the protein structures and the algorithm parameters for the comparison. End-user can submit the comparison request either through web interface or through HTTP API.

Table B.1 lists the library dependencies other than the Apache and PHP. Table B.2 provides the structure of directories in the front-end, respectively.

Table B.1: Library dependencies for the VPC's front-end.

Libraries	Description
PHPMailer (ver. 5.2.13)	An open-source PHP library for sending email in PHP.
Jquery and Jquery-ui (ver. 1.11.2)	An open-source javascript library that simplifies HTML event handling and AJAX interactions.
JSmol	An open-source library for viewing 3D protein structure using HTML5.

Table B.2: Structure of directories in the VPC's front-end.

Directories	Description
`ajax/`	Contains scripts for AJAX and event handling of the user interface.
`api/`	Contains a script which handles the request for comparing two proteins through HTTP API.
`css/`	Contains the style sheets for the user interface.
`internal/`	Contains scripts for internal communication with back-end, to control file deletion and forwarding results.
`js/`	Contains javascript libraries for the user interface, including Jquery.
`jsmol/`	Contains JSmol libraries.
`lib/`	Contains PHP scripts for supporting the user interface, including PHPMailer.
`results/`	Stores the results of comparison. The results will be removed after seven days from the creation date.
`upload/`	Stores the uploaded protein files.

In the following, files for main process of the VPC system are described:

- `/index.php`: The starting point script.
- `/ajax/atab0.php`: Input form (first tabular page).
- `/ajax/atab0submit.php`: Handling the request from the first tabular page.
- `/ajax/atab0upload.php`: Handling file upload.
- `/ajax/atab1.php`: Server status viewer (second tabular page).
- `/ajax/atab1submit.php`: Handling the request from the second tabular page.
- `/ajax/atab2.php`: Result viewer (third tabular page).
- `/ajax/atab2get.php`: Forwarding result files from the result directory.
- `/ajax/atab2submit.php`: Handling the request from the third tabular page.

- `/ajax/atab3.php`: Displays the description of the VPC system and the proposed similarity method.
- `/ajax/atab4.php`: Displays the development history of the VPC system.
- `/ajax/UploadHandler.php`: PHP Class to handle file upload in AJAX.
- `/api/comparepair.php`: PHP script which processes protein comparison request through HTTP API.
- `/internal/index.php`: PHP script which is called from back-end to store the comparison results' files in the front-end storage and notify user through email if requested when the comparison process is completed.
- `/internal/indexdel.php`: PHP script which is called from back-end to remove the comparison results.
- `js/init.js`: a javascript file that contains initialization in the user interface. The rest of the files and directories contain Jquery libraries.
- `lib/checkmail.php`: a PHP script to validate the email address format.
- `lib/libcommon.php`: a PHP script that contains server parameters used in the VPC system, including the base URL, path of the script, the back-end's IP address, and the results path. The parameters have to be consistence with those in the back-end modules.

Back-end

Back-end of the VPC system consists of four main modules which were mainly developed using Python with utilization of POSIX interprocess communication, as follows:

- Gateway module (module A): constantly waiting for any HTTP request from the front-end. The request received from the front-end is reformatted into JSON (JavaScript Object Notation) before passed to different modules through POSIX message queue.
- Visualization process module (module B): constantly waiting for any POSIX message from gateway module in *Request queue*. If a message is available, this module will insert the message to MySQL database. If required, this module will also download PDB file from SCOP Astral database server. Once the PDB files are available, Jmol is then spawn and the database entry corresponding to the request is updated to indicate that the generation of visualization images has been started. When the visualization is completed, a message is passed to module C through *Features extraction queue*.
- Features extractions and subspace module (module C): constantly waiting for

any POSIX message from module B in *Features extraction queue*. If a message is available, this module will update the database entry corresponding to the request to indicate the feature extraction is started. In this module, feature extraction, subspace generation, and similarity measurement are performed. When the computation is completed, a message is passed to moduleD through *Results queue*.

- Output generation module (module D): constantly waiting for any POSIX message in *Results queue*, either error message or completion message. If a message is available, this module will update the database entry corresponding to the request to indicate that the computation has completed. If the message indicates completion instead of error, this module will generate Matlab files including PNG files for use in the front-end. This module then sends the file (in tar.bz2 format) to front-end by accessing the URL path of `internal/index.php` of the front-end.

The entity-relationship diagram of the database used in the MySQL is shown in Figure B.1. Below are the description of each table:

- TRequestRaw2: stores the raw request data. The primary key `id` is automatically assigned by auto-increment integer value. Field `Json` contains the Json formatted data. Field `pidhandler` is reserved for storing the process ID that is currently handling the request. Field `processCode` contains pre-defined integer constant to indicate the process status. Table B.3 displays the list of the code.
- TRequest2: stores the parsed request data. The primary keys `id` and `reqid` are the foreign key from Table TRequestRaw2 and the random request code generated for end-user who requested the computation to be used later when querying the results, respectively. Field `rotations` contains the information of the number of view-points (rotations) followed by either letter 'U' or 'R' to denote uniform or random rotations, respectively. For example, '50U' indicates 50 uniform view-points, while '50R' indicates 50 random view-points. Field `features` contains either 'RAW', 'LBPu2', 'HLAC', or 'RIC-LBP' feature extraction. These values are defined in the front-end PHP script `lib/libcommon.php`. Fields `protein1` and `protein2` contain either SCOP IDs or the file name of the PDBs. Field `methods` is currently fixed to 'MSM'.
- TResult2: stores the result of the computation.
- TProcessDetail2: stores the time stamp for every process listed in Table B.3. When there is any insertion or update for Table TRequestRaw2, new entry will be inserted automatically to this table.

Table B.3: List of process code in VPC.

Code	Description
0	Message is received by gateway.
1	Generation of visualization images is started.
5	Feature extraction is started.
7	Construction of subspaces and similarity computation are started.
8	Similarity computation has just completed.
10	Error occurred.
100	Computation has been completed with success.
101	Computation has been terminated with failure.

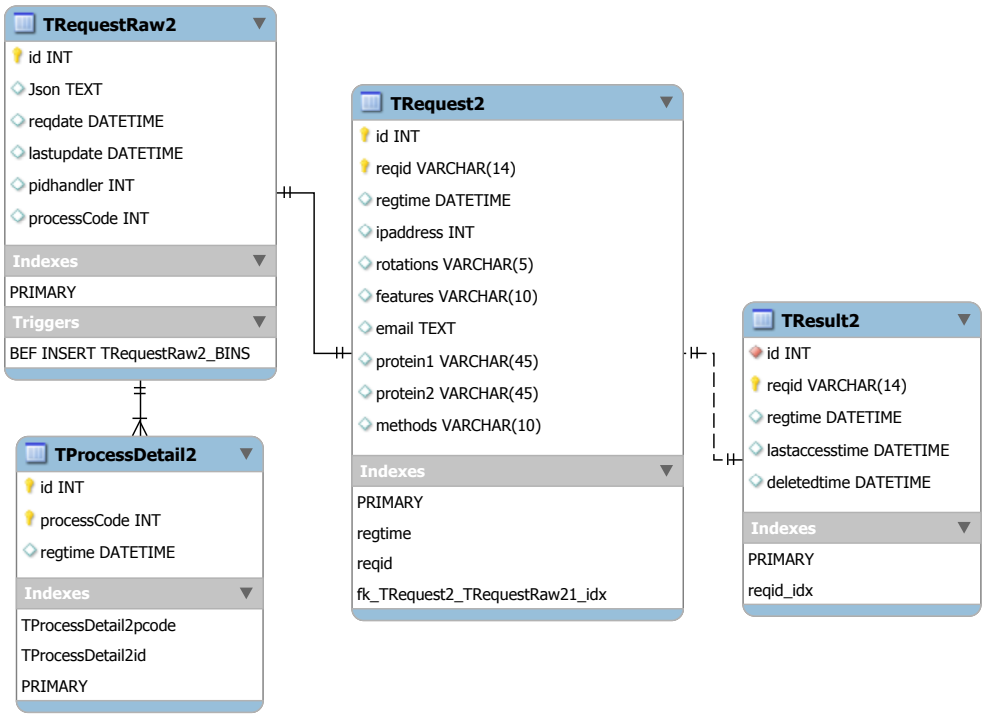

Figure B.1: The entity-relationship diagram of the database.

In the implementation, some of the key constraints are not explicitly enforced in the MySQL for the sake of simplicity.

Tables B.4 and B.5 show the library dependencies and the structure of directories in the back-end, respectively. The files that directly involve in the main process of the VPC system are described as follows:

- `/modules/apaconfig.py`: Contains the configuration of the VPC system,

Table B.4: Library dependencies for the VPC's back-end.

Libraries	Description
Bottlepy [75] (ver. 0.12.3)	A lightweight web-server based on Python for communication with front-end.
MySQL database server (ver. 14.14)	A relational database management system for storing comparison requests.
Jmol [54] (ver. 13.1.16_2013.05.20a)	A 3D molecular visualization software for generating multi-view images.
SciPy and Numpy [76]	Numerical library packages for Python.
Posix_ipc library for Python	A library used for controlling POSIX message queue.
MySQLdb library for Python	A library used for interfacing with MySQL.

Table B.5: Structure of directories in the VPC's back-end.

Directories	Description
`log/`	Stores log file of each module.
`modules/`	Contains classes and modules.
`results/`	Stores the comparison results.
`rotationlist/`	Stores pre-calculated uniformly distributed view-points (virtual camera position) for $20, 50, 100, 250, 500, 750, 1000, 1500,$ and 2000 view-points.
`savedproteins/`	Stores uploaded and downloaded PDB files.
`tmp/`	Stores files with the process id (PID) of running modules.

including paths, database, and other server settings.

- `/modules/apadb.py`: Python class for handling database as a wrapper of MySQLdb.
- `/modules/daemonB.py`: main script for module B.
- `/modules/daemonCv4.py`: main script for module C.
- `/modules/daemonDv2.php`: main script for module D.
- `/modules/deleteResultsNdays.py`: main script for checking if any deletion of result files is required.
- `/modules/featureext.py`: Python class for image feature extraction process.
- `/modules/gateway.py`: main script for gateway module.
- `/modules/jmolscriptgenerator.py`: Python class for generating Jmol script which is used to generate the protein visualization images.

- `/modules/subspacemethod.py`: Python class for construction of subspace and computation of average canonical angles between two subspaces.
- `/modules/run.sh`: Bash script to start the back-end's process.
- `/modules/stop.sh`: Bash script to stop the back-end's process.

B.2 System manual

Front-end setup

Assuming that an Apache web server with PHP extension have been installed, the VPC's front-end setup is straightforward. Although in the current implementation the front-end and the back-end are separated, it is possible to use a single machine for both front-end and back-end. In setting up the front-end, the system configurations in `/lib/libcommon.php` need to be adjusted:

- `BASEURL`: the base URL of the web interface of front-end.
- `UPLOADPATH`: the path for storing uploaded PDB files in front-end.
- `PROTEINSERVER`: the base URL of the back-end's gateway module.
- `RESULTSPATH`: the path for storing the comparison results in front-end.
- `ADMINEMAIL`: displayed email address when sending email.

Back-end setup

Here, we assume that all required dependencies have been installed and all the files and directories have been put in place. Before running the VPC's back-end, the following setup are required:

- Enabling MySQL service and generating the required tables as shown in Figure B.1 using the provided SQL script.
- Creating virtual disk to store the visualization images temporarily. For example: `mount -t ramfs -o mode=1777,size=32G ramfs /mnt/ramdisk/`, assuming that the virtual disk size is 32 GB and located at `/mnt/ramdisk`. Note that this process may need to be executed every time the operating system is restarted. Automation can be done through cron scheduling.
- Initializing the POSIX message queue. Firstly, create an empty directory for storing the message queue files. Then, mount the message queue with the following command:
  `mount -t mqueue none /dev/mqueue`,

assuming that the message queue is stored in directory `/dev/mqueue`. Similarly, this process requires to be executed every time the operating system is restarted.

- The default configuration in Linux OS for the maximum number of message queue is small. Therefore, increasing the limit for the message queue number is highly recommended, by modifying `/etc/sysctl.conf`. For the detail, please refer to the Linux OS documentation.
- The automatic deletion of results' files are performed through scheduler. Therefore, schedule daily execution of `/modules/deleteResultsNdays.py` using daily cron scheduling.
- The system configurations are located in `/modules/apaconfig.py`. Some of the important configurations which require adjustment are as follows:
  - `WWWBASE`: the base URL of the web interface of front-end.
  - `WWWINTERNAL`: the path to access `/internal/index.php` relative to the `WWWBASE`.
  - `WWWINTERNALDEL`: the path to access `/internal/indexdel.php` relative to the `WWWBASE`.
  - `AUTHINTERNAL`: base64 encoded text containing the user and password to access `WWWINTERNAL` and `WWWINTERNALDEL` (it is advisable to restrict the access to `WWWINTERNAL` and `WWWINTERNALDEL` if the system is open to public).
  - `FILEUPLOADPATHUSER`: the directory in the back-end to store the uploaded PDB files.
  - `FILEUPLOADPATHASTRAL`: the directory in the back-end to store the downloaded PDB files.
  - `FILERESULTS`: the directory in the back-end to store the results.
  - `DBHOST`, `DBUSER`, `DBPASS`, `DBNAME`: the configurations for the database (host, user name, password, and database name, respectively).

Running and maintenance

Since the front-end is only a web interface, there is no specific requirement for starting the process of the VPC's front-end. However, Jmol visualization software used in the back-end requires a Graphical User Interface (GUI) environment, such as X11, to synthesize the protein visualization. Consequently, the execution of the starting script for the back-end has to be performed under GUI environment. The back-end is started by executing `/modules/run.sh`. The back-end can be terminated by execut-

ing `/modules/stop.sh`.

As mentioned previously, the removal of results' files are performed through scheduler. However, the current implementation does not perform any deletion in the database. It is advisable to add a regular maintenance of the database as well, especially when the VPC system is extensively used.

B.3 HTTP API interface

Besides through web interface, VPC system can be accessed through HTTP request. The HTTP request requires HTTP POST method directed to the following relative path `api/comparepair.php`. For example, if the base URL of the web interface of the front-end is defined by `BASEURL`, the URL for HTTP API interface is then `BASEURL`/api/comparepair.php.

The request fields are as follows:

- `prot1type`: Set to '1' if the input for protein 1 is in the form of SCOP ID. Set to '2' if the input protein for protein 1 is a PDB file.
- `prot1txt`: Set to the seven-characters of the SCOP ID if `prot1type='1'`. Otherwise, set to empty or let undefined.
- `prot1file`: Set to the original file name of PDB file for protein 1 if `prot1type='2'`. Otherwise, set to empty or let undefined.
- `prot1fileContent`: Set to the content of the PDB file of protein 1 if `prot1type='2'`. Otherwise, set to empty or let undefined.
- `prot2type`: Set to '1' if the input for protein 2 is in the form of SCOP ID. Set to '2' if the input protein for protein 2 is a PDB file.
- `prot2txt`: Set to the seven-characters of the SCOP ID if `prot2type='1'`. Otherwise, set to empty or let undefined.
- `prot2file`: Set to the original file name of PDB file for protein 2 if `prot2type='2'`. Otherwise, set to empty or let undefined.
- `prot2fileContent`: Set to the content of the PDB file of protein 2 if `prot2type='2'`. Otherwise, set to empty or let undefined.
- `features`: Set to feature extraction method, either 'RAW', 'LBPu2', 'HLAC', or 'RIC-LBP'.
- `rotations`: Set to the number of view-points and the type, either uniform or random. Valid values are '50U', '100U', '500U', and '1000U' for uniform type and '50R', '100R', '500R', and '1000R' for random type.
- `methods`: Set to 'MSM'.

- `email`: Set to the email address that will receive notification if the computation is complete (optional). If not needed, set to empty or let undefined.

If there is no error, the response from HTTP the request is a random code which can be used for querying the result. For example, response "`OK:56305cef5006b`" indicates that the request is successfully accepted with request code `56305cef5006b` for querying the result. On the other hand, when error is occurred, an error message is returned as the response. For example, "`ERR:Rotation number is not valid.`" indicates an error has occurred because the parameter for the rotation number is not valid. The list of the error messages can be found in the back-end's file `/modules/apaconfig.py`. To access the comparison results which are submitted through HTTP API, a user can use either the web interface by inputting the request code or through HTTP GET method by passing field `reqcode` with the request code and the result's type `code=12` for obtaining the compressed result file. As an example, if the request code is `56305cef5006b`, the HTTP GET method is directed to `BASEURL/ajax/atab2get.php?reqcode=56305cef5006b&code=12`.

B.4 Closing note

The provision of the VPC system is to reduce the difficulties in implementing the proposed method. Owing to the modularity of the system, it can be easily customized for different types of feature extractions and comparison methods, including different analysis for the given two input proteins. To reduce the computational time, future work includes the implementation of the system using C/C++ programming language and GPU programming.

References

[1] H. B. Vickery, "The origin of the word protein," *The Yale Journal of Biology and Medicine*, vol. 22, no. 5, pp. 387–393, 1950.

[2] A. Liljas, L. Liljas, J. Piskur, G. Lindblom, P. Nissen, and M. Kjeldgaard, *Textbook of Structural Biology*. World Scientific, 2009.

[3] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, *Analyzing Protein Structure and Function*. New York: Garland Science, 4th ed., 2002.

[4] L. Chen, J. K. Morrow, H. T. Tran, S. S. Phatak, L. Du-Cuny, and S. Zhang, "From laptop to benchtop to bedside: Structure-based drug design on protein targets," *Current Drug Metabolism*, vol. 18, no. 9, pp. 1217–1239, 2012.

[5] P. Koehl, "Protein structure similarities," *Current Opinion in Structural Biology*, vol. 11, no. 3, pp. 348–353, 2001.

[6] S. Yang, R. Valas, and P. Bourne, "Evolution studied using protein structure," in *Structural Bioinformatics*, pp. 559–571, Hoboken, NJ: Wiley-Blackwell, 2 ed., 2009.

[7] G. Lancia and S. Istrail, "Protein structure comparison: Algorithms and applications," in *Protein Structure Analysis and Design*, vol. 2666 of *Lecture Notes in Bioinformatics*, pp. 1–33, Berlin Heidelberg: Springer, 2006.

[8] M. Marti-Renom, E. Capriotti, I. Shindyalov, and P. Bourne, "Structure comparison and alignment," in *Structural Bioinformatics*, pp. 397–418, Hoboken, NJ: Wiley-Blackwell, 2 ed., 2009.

[9] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, "The protein data bank," *Nucleic Acids Research*, vol. 28, no. 1, pp. 235–242, 2000. URL: http://www.rcsb.org.

[10] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia, "SCOP: a structural classification of proteins database for the investigation of sequences and structures," *Journal of Molecular Biology*, vol. 247, no. 4, pp. 536–40, 1995.

[11] N. K. Fox, S. E. Brenner, and J.-M. Chandonia, "SCOPe: Structural classification of proteins–extended, integrating SCOP and ASTRAL data and classification of new structures," *Nucleic Acids Research*, vol. 42, no. Database issue, pp. D304–9, 2014.

[12] J.-M. M. Chandonia, G. Hon, N. S. Walker, L. Lo Conte, P. Koehl, M. Levitt, and S. E. Brenner, "The ASTRAL Compendium in 2004," *Nucleic Acids Research*, vol. 32, pp. D189–D192, 2004.

[13] C. H. Q. Ding and I. Dubchak, "Multi-class protein fold recognition using support vector machines and neural networks," *Bioinformatics*, vol. 17, pp. 349–358, 2001.

[14] T. Damoulas and M. A. Girolami, "Probabilistic multi-class multi-kernel learning: on protein fold recognition and remote homology detection," *Bioinformatics*, vol. 10, pp. 1264–1270, 2008.

[15] A. Bairoch, "The ENZYME database in 2000," *Nucleic Acids Research*, vol. 28, no. 1, pp. 304–305, 2000.

[16] C. A. Ouzounis, R. M. R. Coulson, A. J. Enright, V. Kunin, and J. B. Pereira-Leal, "Classification schemes for protein structure and function," *Nature Reviews Genetics*, vol. 4, no. 7, pp. 508–519, 2003.

[17] C. A. Orengo, A. D. Michie, D. T. Jones, M. B. Swindells, and J. M. Thornton, "CATH–a hierarchic classification of protein domain structures," *Structure*, vol. 5, no. 8, pp. 1093–1108, 1997.

[18] L. Holm and C. Sander, "The FSSP database: Fold classification based on structure-structure alignment of proteins," *Nucleic Acids Research*, vol. 24, no. 1, pp. 206–209, 1996.

[19] W. Kabsch and C. Sander, "Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features," *Biopolymers*, vol. 22, no. 12, pp. 2577–2637, 1983.

[20] S. E. Brenner, C. Chothia, T. J. P. Hubbard, and A. G. Murzin, "Understanding protein structure: Using SCOP for fold interpretation," *Methods in Enzymology*, vol. 266, no. 1995, pp. 635–643, 1996.

[21] G. Csaba, F. Birzele, and R. Zimmer, "Systematic comparison of SCOP and CATH: a new gold standard for protein structure analysis," *BMC Structural Biology*, vol. 9, p. 23, 2009.

[22] L. Holm and C. Sander, "Protein structure comparison by alignment of distance matrices," *Journal of Molecular Biology*, no. 233, pp. 123–138, 1993.

[23] I. N. Shindyalov and P. E. Bourne, "Protein structure alignment by incremental combinatorial extension (CE) of the optimal path," *Protein Engineering*, vol. 11, no. 9, pp. 739–47, 1998.

[24] Y. Ye and A. Godzik, "Flexible structure alignment by chaining aligned fragment pairs allowing twists," *Bioinformatics*, vol. 19, no. Suppl 2, pp. ii246–ii255, 2003.

[25] Y. Zhang and J. Skolnick, "TM-align: a protein structure alignment algorithm based on the TM-score," *Nucleic Acids Research*, vol. 33, no. 7, pp. 2302–9, 2005.

[26] L. A. Mirny and E. I. Shakhnovich, "Protein structure prediction by threading. Why it works and why it does not.," *Journal of Molecular Biology*, vol. 283, no. 2, pp. 507–526, 1998.

[27] 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.

[28] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs," *Nucleic Acids Research*, vol. 25, no. 17, pp. 3389–3402, 1997.

[29] C. Chothia and A. M. Lesk, "The relation between the divergence of sequence and structure in proteins," *The EMBO Journal*, vol. 5(4), pp. 823–826, 1986.

[30] Y. Zhang and J. Skolnick, "Scoring function for automated assessment of protein structure template quality," *Proteins*, vol. 57, no. 4, pp. 702–710, 2004.

[31] P. Røgen, "Evaluating protein structure descriptors and tuning Gauss integral based descriptors," *Journal of Physics Condensed Matter*, vol. 17, pp. 1523–1538, 2005.

[32] T. Harder, M. Borg, W. Boomsma, P. Røgen, and T. Hamelryck, "Fast large-scale clustering of protein structures using Gauss Integrals," *Bioinformatics*, pp. 510–515, 2012.

[33] X. Zhou, J. Chou, and S. T. C. Wong, "Protein structure similarity from principle component correlation analysis," *BMC Bioinformatics*, vol. 7, p. 40, 2006.

[34] A. S. Konagurthu, P. J. Stuckey, and A. M. Lesk, "Structural search and retrieval using a tableau representation of protein folding patterns," *Bioinformatics*, vol. 24, no. 5, pp. 645–51, 2008.

[35] A. Stivala, A. Wirth, and P. J. Stuckey, "Tableau-based protein substructure search using quadratic programming," *BMC Bioinformatics*, vol. 10, no. 1, p. 153, 2009.

[36] L. Sael, B. Li, D. La, Y. Fang, K. Ramani, R. Rustamov, and D. Kihara, "Fast protein tertiary structure retrieval based on global surface shape similarity," *Proteins*, vol. 72, pp. 1259–1273, 2008.

[37] G. Mirceva, I. Cingovska, Z. Dimov, and D. Davcev, "Efficient approaches for retrieving protein tertiary structures," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 9, no. 4, pp. 1166–79, 2012.

[38] S. I. O'Donoghue, D. S. Goodsell, A. S. Frangakis, F. Jossinet, R. A. Laskowski, M. Nilges, H. R. Saibil, A. Schafferhans, R. C. Wade, E. Westhof, and A. J. Olson, "Visualization of macromolecular structures," *Nature Methods*, vol. 7, no. 3 Suppl, pp. S42–S55, 2010.

[39] S. Bottomley and E. Helmerhorst, "Molecular visualization," in *Structural Bioinformatics*, pp. 237–268, Hoboken, NJ: Wiley-Blackwell, 2 ed., 2009.

[40] K. Maeda, "From the subspace methods to the mutual subspace method," in *Computer Vision*, vol. 285 of *Studies in Computational Intelligence*, pp. 135–156, Berlin Heidelberg: Springer, 2010.

[41] T. Kim, J. Kittler, and R. Cipolla, "Discriminative learning and recognition of image set classes using canonical correlations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1005–1018, 2007.

[42] H. Sakano, O. Yamaguchi, T. Kawahara, and S. Hotta, "On the behavior of kernel mutual subspace method," in *Computer Vision - ACCV 2010 Workshops*, vol. 6469

of *Lecture Notes in Computer Science*, pp. 364–373, Berlin Heidelberg: Springer, 2011.

[43] K. Fukui and O. Yamaguchi, "Face recognition using multi-viewpoint patterns for robot vision," in *Proc. of the 11th International Symposium of Robotics Research*, pp. 192–201, 2003.

[44] K. Fukui, B. Stenger, and O. Yamaguchi, "A framework for 3D object recognition using the kernel constrained mutual subspace method," in *Computer Vision - ACCV 2006*, vol. 3852 of *Lecture Notes in Computer Science*, pp. 315–324, Berlin Heidelberg: Springer, 2006.

[45] K. Fukui and O. Yamaguchi, "The kernel orthogonal mutual subspace method and its application to 3D object recognition," in *Computer Vision - ACCV 2007*, vol. 4844 of *Lecture Notes in Computer Science*, pp. 467–476, Berlin Heidelberg: Springer, 2007.

[46] K. Fukui and A. Maki, "Difference subspace and its generalization for subspace-based methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 11, pp. 2164–2177, 2015.

[47] M. Peris and K. Fukui, "Both-hand gesture recognition based on komsm with volume subspaces for robot teleoperation," in *Proc. of IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, pp. 191–196, 2012.

[48] Y. Ohkawa and K. Fukui, "Hand-shape recognition using the distributions of multi-viewpoint image sets," *IEICE Transactions on Information and Systems*, vol. E95-D, no. 6, pp. 1619–1627, 2012.

[49] H. Niigaki and K. Fukui, "Classification of similar 3D objects with different types of features from multi-view images," in *Proc. of the 3rd Pacific Rim Symposium on Advances in Image and Video Technology*, pp. 1046–1057, 2009.

[50] J. Hamm and D. D. Lee, "Grassmann discriminant analysis: A unifying view on subspace-based learning," in *Proc. of the 25th International Conference on Machine Learning*, pp. 376–383, 2008.

[51] T. Ojala, M. Pietikainen, and D. Harwood, "A comparative study of texture measures with classification based on feature distribution," *Pattern Recognition*, vol. 29, no. l, pp. 51–59, 1996.

[52] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[53] R. Nosaka, C. H. Suryanto, and K. Fukui, "Rotation invariant co-occurrence among adjacent LBPs," in *Computer Vision - ACCV 2012 Workshops*, vol. 7728 of *Lecture Notes in Computer Science*, pp. 15–25, Berlin Heidelberg: Springer, 2013.

[54] R. M. Hanson, "Jmol - a paradigm shift in crystallographic visualization," *Journal of Applied Crystallography*, vol. 45, no. 5, pp. 1250–1260, 2010.

[55] A. Björck and G. H. Golub, "Numerical methods for computing angles between linear subspaces," *Mathematics of Computation*, vol. 27, pp. 579–594, 1973.

[56] J. Poland, "Three different algorithms for generating uniformly distributed random points on the n-sphere," *Unpublished research note*, 2000. URL: http://www-alg.ist.hokudai.ac.jp/~jan/randsphere.pdf accessed 2015/12/16.

[57] H. Cohn and A. Kumar, "Universally optimal distribution of points on spheres," *Journal of the American Mathematical Society*, vol. 20, no. 1, 2006, 0607446.

[58] H. Peng and Y. Yu, "Optimization on the surface of the (hyper)-sphere," *Unpublished project report*, 2012. URL: https://www.cs.purdue.edu/homes/pengh/reports/590OP.pdf accessed 2015/12/16.

[59] C. G. Koay, "Distributing points uniformly on the unit sphere under a mirror reflection symmetry constraint," *Journal of Computational Science*, 2014, 1403.3851v2.

[60] J. J. Moré and D. C. Sorensen, "Computing a trust region step," *SIAM Journal on Scientific and Statistical Computing*, vol. 4, pp. 553–572, 1983.

[61] N. Otsu and T. Kurita, "A new scheme for practical flexible and intelligent vision systems," in *IAPR Workshop CV*, pp. 431–435, 1988.

[62] S. Liao and A. Chung, "Face recognition by using elongated local binary patterns with average maximum distance gradient magnitude," in *Computer Vision - ACCV 2007*, vol. 4844, pp. 672–679, Berlin Heidelberg: Springer, 2007.

[63] X. Tan and W. Triggs, "Fusing Gabor and LBP feature sets for kernel-based face recognition," in *Analysis and Modeling of Faces and Gestures*, vol. 4778 of *Lecture Notes in Computer Science*, pp. 235–249, Berlin Heidelberg: Springer, 2007.

[64] C. Shan, S. Gong, and P. W. McOwan, "Appearance manifold of facial expression," in *ICCV 2005 Workshop on HCI*, vol. 3766, pp. 221–230, Berlin Heidelberg: Springer, 2005.

[65] C. Shan and T. Gritti, "Learning discriminative LBP-histogram bins for facial expression recognition," in *Proc. of the British Machine Vision Conference*, pp. 27.1–27.10, 2008.

[66] Y. Mu, S. Yan, Y. Liu, T. Huang, and B. Zhou, "Discriminative local binary patterns for human detection in personal album," in *Proc. of the Computer Vision and Pattern Recognition*, pp. 1–8, 2008.

[67] X. Wang, T. X. Han, and S. Yan, "An HOG-LBP human detector with partial occlusion handling," in *Proc. of 12th IEEE International Conference on Computer Vision*, pp. 32–39, 2009.

[68] T. Mäenpää and M. Pietikäinen, "Classification with color and texture: jointly or separately?," *Pattern Recognition*, vol. 37, no. 8, pp. 1629–1640, 2004.

[69] L. Nanni, A. Lumini, and S. Brahnam, "Local binary patterns variants as texture descriptors for medical image analysis," *Artificial intelligence in Medicine*, vol. 49, no. 2, pp. 117–25, 2010.

[70] R. Nosaka, Y. Ohkawa, and K. Fukui, "Feature extraction based on co-occurrence of adjacent local binary patterns," in *Advances in Image and Video Technology*, vol. 7088 of *Lecture Notes in Computer Science*, pp. 82–91, Berlin Heidelberg: Springer, 2012.

[71] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, A. Smola, and K.-R. Muller, "Constructing descriptive and discriminative nonlinear features: Rayleigh coefficients in kernel feature spaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 623–628, 2003.

[72] D. Cai, X. He, and J. Han, "Speed up kernel discriminant analysis," *The VLDB Journal*, vol. 20, no. 1, pp. 21–33, 2011.

[73] A. Prlic, S. Bliven, P. W. Rose, W. F. Bluhm, C. Bizon, A. Godzik, and P. E. Bourne, "Precalculated protein structure alignments at the RCSB PDB website," *Bioinformatics*, vol. 26, pp. 2983–2985, 2010.

[74] N. Japkowicz and M. Shah, "Statistical significance testing," in *Evaluating learning algorithms: a classification perspective*, pp. 206–290, New York: Cambridge University Press, 2011.

[75] "Bottle: Python web framework." URL: http://bottlepy.org/ accessed 2015/12/16.

[76] T. E. Oliphant, "Python for scientific computing," *Computing in Science and Engineering*, vol. 9, no. 3, pp. 10–20, 2007.

[77] N. Shental, T. Hertz, D. Weinshall, and M. Pavel, "Adjustment learning and relevant component analysis," in *Computer Vision ECCV 2002* (A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, eds.), vol. 2353 of *Lecture Notes in Computer Science*, pp. 776–790, Springer Berlin Heidelberg, 2002.

[78] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighbourhood components analysis," in *Advances in Neural Information Processing Systems*, pp. 513–520, 2005.

[79] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, pp. 207–244, 2009.

[80] C. Xiong, D. Johnson, R. Xu, and J. J. Corso, "Random forests for metric learning with implicit pairwise position dependence," in *Proc. of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 958–966, 2012.

[81] F. Wang, J. Sun, and S. Ebadollahi, "Composite distance metric integration by leveraging multiple experts' inputs and its application in patient similarity assessment," *Statistical Analysis and Data Mining*, vol. 5, no. 1, pp. 54–69, 2012.

[82] Z. Cui, W. Li, D. Xu, S. Shan, and X. Chen, "Fusing robust face region descriptors via multiple metric learning for face recognition in the wild," in *Proc. of the Computer Vision and Pattern Recognition*, pp. 3554–3561, 2013.

[83] X. Zhai, Y. Peng, and J. Xiao, "Heterogeneous Metric Learning with Joint Graph Regularization for Cross-Media Retrieval," in *Proc. of the 27th AAAI Conference on Artificial Intelligence Heterogeneous*, pp. 1198–1204, 2013.

[84] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. V. N. Vishwanathan, A. J. Smola, and H. P. Kriegel, "Protein function prediction via graph kernels," *Bioinformatics*, vol. 21, pp. 47–56, 2005.

[85] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *Journal of Machine Learning Research*, vol. 5, pp. 27–72, 2004.

[86] M. Varma and B. R. Babu, "More generality in efficient multiple kernel learning," in *Proc. of the 26th Annual International Conference on Machine Learning*, ICML '09, (New York, NY, USA), pp. 1065–1072, ACM, 2009.

[87] H. Hino, N. Reyhani, and N. Murata, "Multiple kernel learning with gaussianity measures," *Neural Computation*, vol. 24, no. 7, pp. 1853–1881, 2012.

[88] B. Kulis, "Metric learning: A survey," *Foundations and Trends in Machine Learning*, vol. 5, no. 4, pp. 287–364, 2013.

[89] J. D. M. Rennie and N. Srebro, "Loss functions for preference levels: Regression with discrete ordered labels," in *Proc. of the IJCAI Multidisciplinary Workshop on Advances in Preference Handling*, pp. 180–186, 2005.

[90] N. Shervashidze and K. Borgwardt, "Fast subtree kernels on graphs," in *Proc. of the Neural Information Processing Systems*, pp. 1660–1668, Neural Information Processing Systems Foundation, 2009.

[91] M. Neumann, N. Patricia, R. Garnett, and K. Kersting, "Efficient graph kernels by randomization," in *Proc. of the 2012 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part I*, pp. 378–393, 2012.

[92] A. Feragen, N. Kasenburg, J. Petersen, M. de Bruijne, and K. M. Borgwardt, "Scalable kernels for graphs with continuous attributes," in *Proc. of the Neural Information Processing Systems*, pp. 216–224, 2013.

[93] C. T. Kelley, "Line search methods and the Armijo rule," in *Iterative Methods for Optimization*, vol. 18, pp. 40–52, SIAM, 1999.

[94] J. Wang, A. Woznica, and A. Kalousis, "Parametric local metric learning for nearest neighbor classification," in *Advances in Neural Information Processing Systems*, pp. 1610–1618, 2012.

List of Related Publications

1. Chendra Hadi Suryanto, Shukun Jiang, Kazuhiro Fukui. Protein structures similarity based on multi-view images generated from 3D molecular visualization. *Proc. of 21st International Conf. on Pattern Recognition (ICPR)*, pp.3447-3451, 2012.

2. Chendra Hadi Suryanto, Hiroto Saigo, Kazuhiro Fukui. Protein Clustering on a Grassmann Manifold. *Proc. of 7th IAPR International Conf. on Pattern Recognition in Bioinformatics (PRIB)*, LNCS vol. 7632, pp.71-81, 2012.

3. Chendra Hadi Suryanto, Hideitsu Hino, Kazuhiro Fukui. Combination of Multiple Distance Measures for Protein Fold Classification. *Proc. of 2nd IAPR Asian Conference on Pattern Recognition (ACPR)*, 2013.

4. Chendra Hadi Suryanto, Kazuhiro Fukui, Hideitsu Hino. Protein Fold Classification using Large Margin Combination of Distance Metrics. IEICE Transactions on Information and Systems (scheduled to be published on Vol.E99-D, No.3. 2016).