INFORMATION-ASSISTED VOLUME RENDERING AND VISUAL EVALUATION THROUGH MACHINE INTELLIGENCE

by

Naimul Mefraz Khan

Master of Science, University of Windsor, 2010

Bachelor of Science,

Bangladesh University of Engineering and Technology, 2008

A Dissertation

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

in the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2014

Author's Declaration

Information-Assisted Volume Rendering and Visual Evaluation through Machine Intelligence

Doctor of Philosophy 2014

Naimul Mefraz Khan

Electrical and Computer Engineering

Ryerson University

Abstract

Exploration and visualization of complex data has become an integral part of life. But there is a semantic gap between the users and the visualization scientists. The priority of the users is usability while that of the scientists is techniques. Information-Assisted Visualization (IAV) can help bridge this gap, where additional information extracted from the raw data is presented to the user in an easily interpretable way. This thesis proposes some novel machine intelligence based systems for intuitive IAV.

The majority of the thesis focuses on Direct Volume Rendering, where Transfer Functions (TF) are used to color the volume data to expose structures. Existing TF design methods require manipulating complex widgets, which may be difficult for the user. We propose two novel approaches towards TF design.

In the data-centric approach, we generate an organized representation of the data through clustering and provide the user with some intuitive control over the output in the cluster domain. We use Spherical Self-Organizing Maps (SSOM) as the core of this approach. Instead of manipulating complex widgets, the user interacts with the simple SSOM color-coded lattice to design the TF.

In the image-centric approach, the user interaction with the data is direct and minimal. The user interactions create the training data, and supervised classification is used to generate the TF. First, we propose novel supervised classifiers that combine the local information available through Support Vector Machine-based classifiers and the global information available through Nonparametric Discriminant Analysis-based classifiers. Using these classifiers, we propose a TF design method where the user interacts with the volume slices directly to generate the output.

Finally, we explore the use of IAV for home-based physical rehabilitation. We propose an information-assisted visual evaluation framework which can compare a user's performance of a physical exercise with that of an expert using our novel Incremental Dynamic Time Warping method and communicate the results visually through our color-mapped skeleton silhouette.

All the proposed techniques are accompanied by detailed experimental results comparing them against the state-of-the-art. The results shows the potential of using machine learning techniques to achieve visualization tasks in a simpler yet more effective way.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my PhD supervisor, Dr. Ling Guan, for his patience, support and guidance throughout my time at Ryerson University. His enthusiasm, inspiration, and advice made my PhD journey both delightful and productive. He is truly a role model: not only as a keen and productive researcher, but also as a great human being.

I would also like to thank Dr. Matthew Kyan for his valuable guidance and technical input which essentially shaped the outline of my dissertation in the beginning. I also thank him for serving as the chair of the defense committee.

I am grateful to Dr. Steve Lin and my co-supervisor Dr. Baining Guo for their mentorship during my brief stay at Mircosoft Research Asia. Their insightful comments were very helpful in producing significant research results within a short period.

It was a blessing for me to be part of the Ryerson Multimedia Research Laboratory (RML). I gained both knowledge and friendship during my tenure at RML, working with my fellow researchers and students. My years in RML have been and will always be the most memorable time of my life.

For this dissertation, my gratitude also goes to the defense committee members: Dr. George K. Knopf, Dr. Tim McInerney, Dr. Xiao-Ping Zhang, and Dr. Sri Krishnan for their time and valuable suggestions.

Last but not the least, I would like to thank my family for their unconditional love, encouragement, and continual support for the past four years.

Table of Contents

List of Tables

List of Figures

List of Acronyms

BMU	Best Matching Unit
COSVM	Covariance-guided One-Class Support Vector Machine
CT	Computed Tomography
DTW	Dynamic Time Warping
DVR	Direct Volume Rendering
FSCL	Frequency-Sensitive Competitive Learning
GMM	Gaussian Mixture Model
GUI	Graphical User Interface
IAV	Information-Assisted Visualization
IDTW	Incremental Dynamic Time Warping
k-NN	k-Nearest Neighbors
KAV	Knowledge-Assisted Visualization
KFD	Kernel Fisher Discriminant Analysis
KN-SVM	Kernel Nonparametric Support Vector Machine
KSVM	Kernel Support Vector Machine
LDA	Linear Discriminant Analysis
MOSVM	Mahalanobis One-class Support Vector Machine
MRI	Magnetic Resonance Imaging
NDA	Nonparametric Discriminant Analysis
OSG	OpenSceneGraph
OSVM	One-class Support Vector Machine
PCA	Principal Component Analysis
RBF	Radial Basis Function
ROC	Region of Convergence

SOM	Self-Organizing Map
SSOM	Spherical Self-Organizing Map
SVDD	Support Vector Data Description
SVM	Support Vector Machine
TBI	Traumatic Brain Injury
TF	Transfer Function
VTK	Visualization Toolkit

Chapter 1

Introduction

1.1 The Problem

The human perception system is more capable of inferring required information from visual patterns rather than mere raw data [1]. As a result, visualization have found its application in several fields, including medical imaging [2, 3], seismology [4], computer network analysis [5], traffic analysis [6], organizational management [7] etc.

Despite the significant expansion of visualization research over the past few years, we have still not been able to provide a universal satisfactory visualization framework, which can simultaneously provide fast interactive rendering and easily accessible information. The main obstacle in establishing easy accessibility is the complexity of a visualization system. Presentation of complex information in an interpretable way is generally considered a daunting task. Any visualization system requires active involvement of domain experts and visualization researchers. The hardware and software demands of even a simple visualization system is also understandably high [8]. As an example, we can consider the field of 3D medical imaging, which demands high accuracy and reliability in both the presentation and interpretation of the information being visualized [9]. The difficulty with such a system is evident from the observation that even the medical professionals (domain experts) deal with confusion when they have to pick one among several visualization techniques, resulting in selection of a tool or technique that eventually provides below-par performance [10].

The complexity in designing a good visualization system arises from the so-called "semantic gap" [11] (Figure 1.1). Visualization researchers have always been focused on the underlying algorithms, rendering techniques and hardware setup to make a system more efficient and interactive. On the other hand, users (domain experts) are mainly concerned with the ease-of-use and the information availability. Most of the times the visualization system designer does not have a clear idea about what the user expects from the system, and vice versa [12]. Unfortunately, these two groups have not been able to successfully merge their requirements and goals yet, and the users still find the visualization systems to be complex and unintuitive [13].

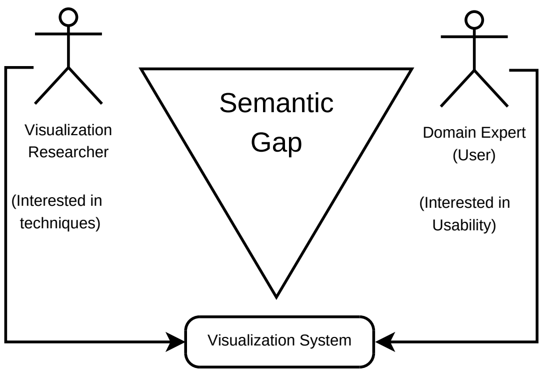


Figure 1.1: The semantic gap.

One of the key factors affecting the performance of a visualization system nowadays is the human-computer interaction side of it [13]. Figure 1.2 shows a typical visualization process. In the user space, the user interacts with the data through some form of controls (style, layout, viewing position etc.) to interact with the data. From these controls, the visualization is generated. The target of the user is to gather knowledge from the visualization [14].

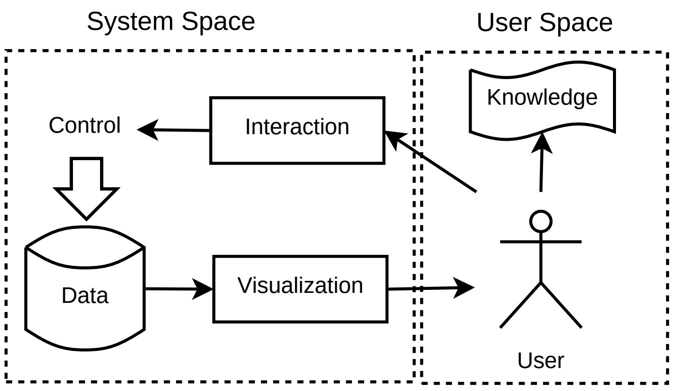


Figure 1.2: A typical visualization process.

To facilitate the user in making the decision of how the controls should be used, the term *Information-Assisted Visualization (IAV)* was coined in [14]. In IAV, additional information regarding the data is presented to the user through some processing (Figure 1.3). Information is defined as "data that represents the results of a computational process, such as statistical analysis, for assigning meanings to the data, or the transcripts of some meanings assigned by human beings" [14]. Examples of information for visuaization are histograms, clustering results etc. The user can make use of this information to guide the decision and control process. In essence, the information also provides the user with some knowledge about the data.

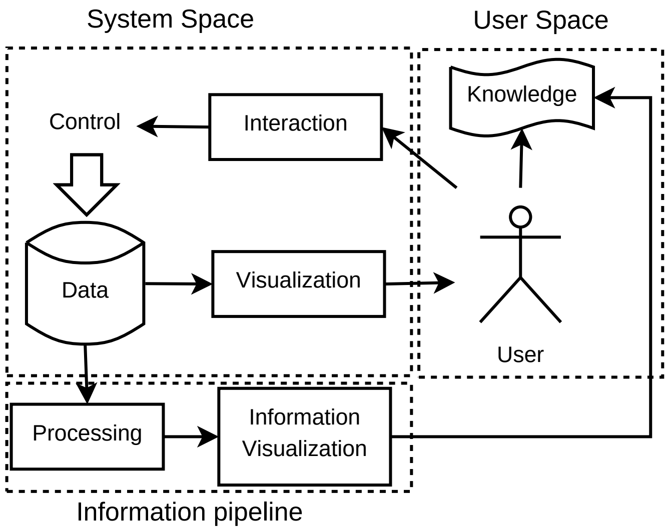


Figure 1.3: Information-Assisted Visualization.

1.2 Thesis Contributions

This thesis explores ways to reduce the aforementioned semantic gap through IAV. Our target is to combine the information pipeline of IAV with the interaction (Figure 1.3) through intuitive interfaces, so the user can directly benefit from the processing of the data. To process the data for extracting information, we use novel machine learning techniques where the user can participate in the decision making process through easy-to-use controls. We want to provide the user with the right amount of control over the visualization process. Our philosophy is that the user should decide *what* he wants to see, not *how* it should be shown. Hence, we automate the rigorous visualization parameters selection process, but leave the data selection up to the user.

The majority of the thesis focuses on the field of Direct Volume Rendering (DVR) [15], which is a very popular technique for volume visualization. DVR generates the 3D view directly from the data without requiring any intermediate representation. The generation of 3D views depends on the definition of the *Transfer Function* (TF). A traditional TF maps the intensity values of data images to different colors and opacities to generate a novel view.

Most of the existing DVR systems provide graphical form of histograms [16] to the user to manipulate with. However, since the end-users may not be familiar with computer graphics and image processing, the complexity of a histogram-based interface can hinder their work flow.

In this thesis, we explore the use of several machine learning techniques to classify the volume data before the generation of the TF, and present the classification information to the user in a more interpretable way. We still want to keep the flexibility of the visualization system intact. By flexibility we mean that the user should have control over how the data should be separated. This flexibility is necessary because as discussed before, the user is ultimately the domain expert, and only he/she knows what to look for.

This thesis presents two alternatives for intuitive visualization of volume data:

- **Data-centric Approach**: In the data-centric approach, the key idea is to perform some kind of clustering on the volume data first. The cluster information is then presented to the user in a visual form. The user interacts with the cluster visualization to discover interesting structures in the volume data. The user interaction is translated into rendering parameters. Using these rendering parameters, the volume data is rendered and shown to the user immediately. This approach is called data-centric because an intermediate form extracted from data properties is presented to the user rather than the data itself. The user does not interact with the volume data directly.

  Since the volume data is clustered before any user interaction, we opted for unsupervised classification with the data-centric approach. We have used Self Organizing Maps (SOM) [17] to perform the clustering. SOM retains the topological relations among the clusters while maintaining a lower dimensional form. Another attractive property of SOM is that it maintains a regular lattice structure which is derived from the relations among the clusters. Since this lattice structure already has a connected form, it is easy to visualize.

Hence, SOM was our choice of unsupervised classifier.

After the classification, the user can interact with the SOM lattice to explore cluster relations. The volume rendering is automatically generated based on the user selection on the lattice. Details on this method are presented in Chapter 3.

- **Image-centric Approach**: In the image-centric approach, instead of working in the cluster domain, the user works with the data directly. Volume data consists of slices. The user is presented with slices with the most variation (based on Entropy). Interesting regions in the volume are selected by the user. The user selection acts as the training data for supervised classification. For supervised classification, there are two popular directions. One is the Linear Discriminant Analysis (LDA)-based methods such as the Nonparametric Discriminant Analysis (NDA) [18]. These methods discriminate between the classes based on the between and within class scatter matrices. In other words, they rely on the global or near-global information available in the training data. On the other hand, Support Vector Machine (SVM)-based methods construct the decision hyperplane with the help of support vectors which lie near the boundary of the classes. In other words, they rely on the local information available in the training data. In this thesis, we combine these two available sources of information and propose the Kernel Nonparametric Support Vector Machine (KN-SVM) classifier. We also apply the same principle of combination for one-class classification or novelty detection, which is an important field in pattern recognition [19]. For one-class classification, we propose the Covariance-guided One-Class Support Vector Machine (COSVM) classifier.

  Our KN-SVM classifier is then incorporated into the image-centric approach towards visualization. Based on the training data selected by the user, the whole volume is classified and eventually the volume rendering is shown to the user.

We have omitted the discussion on one important step in volume rendering till now. This step is the assignment of color and opacity values to the classified voxels. In most existing volume rendering frameworks, the color and opacity assignment is mostly manual or adhoc [20]. However, the color assignment step should be given more importance because ultimately the only way the user can differentiate between the voxel classes is through visualization. Therefore, we have opted for using harmonic colors [20] in our volume visualization frameworks. Harmonic colors are a set of colors which look aesthetically pleasing when visualized together.

Besides using harmonic colors, we use the user's perception and viewing habits to automate the opacity defining process. Due to the automation of the color and opacity assignment procedure, our proposed frameworks takes a lot of burden off the user. This automated procedure is used in both the data-centric and image-centric approach.

Finally, we utilize the benefits of IAV in a field where it has never been used before; medical rehabilitation. We propose a novel in-home rehabilitation process where the user does not need to be under the constant supervision of a medical professional. The user can simply follow the pre-recorded demonstration video of an expert. To evaluate the user's performance in real-time, we propose a novel Incremental Dynamic Time Warping (IDTW) algorithm. The IDTW algorithm extends the popular Dynamic Time Warping (DTW) algorithm by incorporating the ability to compare an incremental sequence (the user's) with a complete sequence (the expert's). Moreover, the IDTW utilizes the recurring nature of DTW calculation and optimizes the computational time so that the distance measurement can be performed in real-time.

The Information-Assisted Visualization comes into play in the way we present the performance evaluation to the user. Instead of showing a single measurement score which can be difficult to evaluate, we use a novel information-assisted visual evaluation scheme. We map separate IDTW scores to a color table. We then use this color table to show a colored skeleton silhouette, where different colors on limb represent the level of performance of the user. Through a quick glance on the skeleton, the user can easily see where he/she is going wrong and correct it immediately.

1.3 Organization of the Thesis

The rest of the thesis is organized as follows: in chapter 2, we discuss the existing works on visualization, mainly focusing on volume rendering. Chapter 3 presents our data-centric approach towards volume rendering. Chapter 4 presents the proposed KN-SVM and COSVM methods. Chapter 5 presents our image-centric approach, where the KN-SVM method is used to design another volume rendering framework. Chapter 6 presents the information-assisted visual evaluation method for medical rehabilitation. All the novel methods proposed in this thesis are accompanied with detailed experiments on benchmark datasets and comparison with other state-of-the-art methods.

Finally, chapter 7 draws conclusions with some future directions on how to progress further.

Chapter 2

Related Works

2.1 Introduction

Scientific visualization is the analysis and exploration of scientific data that have close ties to real-world objects with spatial properties. The goal is to generate a visual representation of something for which we have spatial information and combine that with data that is less directly accessible. For example, in medical visualization, MRI or CT scan data shows the visual representation of the patient's inner organs with an overlay of necessary information (e.g. intensity of the voxels). The need for visualization emerged from the necessity of informative exploration of data in several domains [21]. Extracting information is the ultimate goal of data exploration. Creating an effective visualization entails much more than simply converting raw numbers into graphs or images. The visualization needs to be simple, quick and powerful, so that it can actually ease the process of gathering knowledge rather than making it even more complex. Hence, the raw data needs to be processed to extract the required information, and the information needs to be organized in a way that can be represented visually. The whole process needs to be fast enough for real-time user interaction.

2.2 Direct Volume Rendering

Our focus on this thesis is volume rendering, which is a popular technique for visualization of scalar data in 3D. Volume rendering is especially popular in the medical domain, where the scalar data of the interior of the human body is obtained by CT or MRI scan. It is also popular in computational fluid dynamics, geology, seismography etc [22]. Volumetric models have also recently become popular in the entertainment industry due to the expressiveness and beauty of the 3D rendered results [22].

Direct Volume Rendering (DVR) is a method for volume rendering where no intermediate representation of the scalar data is necessary [16]. In contrast to the traditional isosurface rendering, it is not required to calculate and construct surfaces from the raw data. The colors and opacity values for each voxel is calculated with the help of a Transfer Function (TF). The TF maps different voxel properties (e.g. intensity) to different color and opacity values. There are several benefits of this direct approach over the isosurface approach [16]:

- Isosurface rendering requires considerable amount of preprocessing for the calculation of isosurface for each voxel. Since DVR works on raw data, this extra time is not required, and rendering results can be acquired faster.

- DVR has greater flexibility in determining how each voxel will contribute to the final rendering. This can be important in case of medical imaging, where the measurement artifacts from instruments can distort the final result for isosurfaces [16]. This is because isosurface is typically a binary classification, either a voxel passes through the surface or it does not. This can result in poor isosurface reconstructions which provides a false sense to the user that there is some roughness around those regions. But results from DVR can easily show that the roughness is due to poor measurements, not the data itself. Further details on the comparison between these two methods can be found in [16].

We focus on the TF aspect of DVR. Traditionally, TFs were one-dimensional, where the color and opacity are derived from intensity value only. The user assigns different color and opacity values to the one-dimensional histogram derived from intensity values [15]. However, volume data is typically complex in nature and naturally, intensity value alone is not a good feature to separate voxels into different groups.

Efforts have been taken to improve the TF generation process. The existing methods to generate TF for DVR can be divided into two categories: data-centric methods and image-centric methods.

2.2.1 Data-Centric Methods

In the *data-centric* approach, the system tries to find the TF solely based on data properties. Besides intensity, other properties of the data (e.g. gradient magnitude) can be used to design multi-dimensional TFs. The work of Kindlmann and Durkin first popularized multi-dimensional TF [16]. Their work used gradient magnitude alongside intensity value to build a 2D histogram, where material boundaries in a volume can be identified as arches. The arches can be assigned color and opacity values manually with the help of the histogram. The limitation of this method was the overlapping of arches for different boundaries, making them indistinguishable. Their later work [23] has included the second derivative to build the histograms. In [24] the *LH-Histogram* was proposed. The LH-histogram is calculated by tracing the voxels along the boundary. The low and high intensity values along the gradient were recorded and used to build a histogram. In this approach, boundaries are represented as blobs rather than arches, which are relatively easier to distinguish [24]. The main drawback with this approach is that the histogram takes a long time to calculate [24]. Rather than using the magnitude of the voxel properties, the statistical relationships among them (mean, standard deviation etc.) were used to generate the histograms in [25]. However, directly using statistical properties is sensitive to noise as mentioned by the authors in [25].

Recently, *feature size* has also been used to design multi-dimensional TFs. In [26], the voxels were assigned a scale field, which is calculated from the relative size of a particular feature for each voxel. A size-based distribution is then presented to the user for TF generation. A region-growing technique was used to find the feature size in [27]. In [28], structure sizes were estimated based on a user-defined intensity range. This structure size was then used to generate a Structure Size Enhanced (SSE) histogram, which in turns was used to specify the TF.

Users generally may not be able to focus on voxels too far apart due to the complexity of volume datasets. As a result, spatial information has been used as features in some recently proposed methods. Local histograms combining intensity and spatial information in the local neighborhood was proposed in [29]. In [30], the α-histogram was proposed, which incorporates the property of spatial coherence to produce a global histogram. Spatial information was combined with the 2D histogram in [31]. A distance-based TF was introduced in [32] which assigns optical properties to structures based on the distance to a selected reference structure in order to hide or emphasize structures. A Volume Histogram Stack (VHS) is introduced in

[33], where the histogram is created by aligning the histograms of the different slices and incorporating spatial information with them. Then a Self-Generating Hierarchical Radial Basis Function Network is used to generate the TF.

Some recent methods have used different clustering algorithms on the histogram data to divide the voxels into different groups and assign TF properties accordingly. Voxels were grouped based on their LH-histogram and spatial information in [34]. A non-parametric TF automation method based on kernel density estimation was proposed in [35]. The estimated density distribution is divided into different ranges, and the user specifies the color and opacity values for each range to generate the final output. A parallel technique based on mean-shift clustering of voxel intensities and spatial values was proposed in [36]. A semi-automatic method based on a combination of mean-shift clustering and hierarchical clustering was proposed in [37], where the user has control over the hierarchical clustering results. Self-Organizing Maps were used in [38] to perform dimensionality reduction on the feature space. Different features of the voxels and the maps were then linked to the color and opacity values to generate the final rendering based on user action.

2.2.2 Image-Centric Methods

Another alternative for TF specification is the *image-centric* approach, where the users are given the option to interact with the data in the image domain i.e. directly with the image slices or the rendered volume itself [39]. The changes to the TF are done in the background. Image-centric approaches also include methods where the TF is picked by the user from a set of TFs based on the output rendering, not based on volume properties. The Design Gallery method [40] is a popular image-centric approach. In this method, several different TFs are generated by varying the input parameters. The user is then presented with the output generated from all the TFs, and the task of final TF selection is left to the user. Gaussian Mixture Models (GMM) were used in [41], where the user can manipulate the different parameters of the models instead of directly manipulating TF parameters. An intelligent user interface has been introduced in [42], where the user can paint on different slices to mark areas of interests. Neural network based techniques are then used to generate the TF. A spreadsheet-like interface based on Douglas-Peucker algorithm is presented in [43].

2.2.3 Color Assignment

Most of the methods discussed above mainly focused on classifying the voxels in the volume data into different groups/regions. However, the final rendering is obtained by assigning some color and opacity values to the voxels. In most of the existing methods, this color and opacity assignment is manual. The visual aesthetics is very important for volume rendering, as the user can only draw required conclusions from the volume data visually. So the colors assigned to different voxel groups need to be differentiable. The concept of *color harmonization* can help here [44]. Harmonic colors are a set of colors who look aesthetically pleasing when visualized together. Using harmonic colors can take the burden of color selection away from the user.

Zhou et al. were the first to use the concept of color harmonization in TF generation [45]. Their method adopted a residue flow model based on Darcy's law for TF generation. Color harmonization was also used in [20], where the K-means++ algorithm was used to divide the voxels into groups based on their intensity values and spatial information. This method also uses automatic assignment of opacity values based on the user's perception.

2.3 Summary

The recent volume rendering methods from literature have been discussed in this chapter. Despite all these efforts, TF specification is still a difficult task. Most of these methods still rely on extensive user interaction for histogram manipulation or some form of cluster control (e.g. number of clusters, variance of clusters, merging and splitting of clusters) in the feature space. An expert from medical or architectural background might not be familiar with these specifications. A completely automated rendering is also not very desirable, as only the user has the necessary domain knowledge to decide which regions in the volume he or she needs to focus on. Hence, the user control over the rendering is necessary, but it also needs to be as simple as possible. The next few chapters explore different machine learning techniques to find this balance between automation and customization.

Chapter 3

Data-Centric Approach Towards Volume Rendering

3.1 Introduction

In this chapter, we present a new method for data-centric TF design approach. As discussed before, in the data-centric approaches, the user does not interact with the data directly. Information is extracted from the data (in the form of histograms, clusters etc.) and presented to the user through a GUI. The user interacts with the GUI to find interesting regions. The output rendering is generated accordingly.

Data-centric TF design is typically a user-controlled process, where the user interacts with different widgets (usually representing feature clusters or 1D/2D histograms) to set color and opacity properties to the feature space. In case of clusering-based TF design techniques, the user can also control some low-level properties like number of clusters, cluster variance etc. Most of the recently proposed DVR methodologies [16, 37, 46, 33] are based on this philosophy.

However, interacting with the feature space is difficult for the end-user, who may not have any knowledge about feature extraction and clustering. Multi-dimensional feature spaces can not represent distinguishable properties such as peaks and valleys which are important for proper TF generation from histogram [35]. As a result, the user requires extensive knowledge about the actual volume and the TF widgets themselves. Also, these kind of widgets try to represent the feature space directly, putting a strict restriction on the type of features used and the dimensionality. Some methods use alternative ways to represent the higher-dimensional feature space through manageable widgets. For instance, in [31], spatial information is encoded in the color values while opacity is derived through intensity and gradient. But these kind of alternatives are restrictive in the sense that only the specific features used in the proposed methods can be used for all datasets. Volume rendering has wide range of applications in different fields, and one set of features useful for a specific application might be completely irrelevant in another. There is a need to make the method independent so that any feature can be represented to the user in a visual form while maintaining the topological relationship between various data distributions. Hence, a good DVR system should be: 1) intuitive and easy to use without extensive technical experience, 2) time efficient by providing immediate results, 3) have separate classification and feature modules independent of each other so that any type of feature can be integrated easily and efficiently and 4) visually pleasing.

A machine learning method that can satisfy all the aforementioned requirements is Self-Organizing Map (SOM) [17]. SOM is an unsupervised clustering algorithm that maps continuous input feature space patterns to a set of discrete output nodes. The nodes are arranged in a pre-defined lattice. Each node has a weight vector associated with it. The input patterns (or feature vectors) are assigned to the closest node in the Euclidean sense. In other words, the nodes "compete" among themselves to be representative of the underlying input patterns [47]. As a result, at the end of the training cycle, the node lattice represents a topology-preserving low-dimensional representation of the original feature space. Because of the unique training procedure of SOM, clusters existing in the original space preserve their relationship in the lower-dimensional space. Because of the regular lattice structure, SOM nodes are easy to visualize. They can also be colored in a very intuitive way by using U-matrices [17], so that the end-user can quickly identify underlying clusters. Since SOM nodes have direct correspondence with the original input data (voxels in this case), the user can be presented with the colored SOM lattice. An end-user can select the clusters he or she deems to be important. In this way, the user will have full control over the regions to be defined as important. Due to the ability to use higher-dimensional features than mere 2D/3D as in the case of widget-based methods, the cluster representation can be sophisticated so that user interest is reflected in the grouping of voxels.

However, mere correspondence between SOM nodes and voxels is not enough for effective volume rendering. We need to generate the color and opacity properties to be assigned to the different clusters selected by the user. The typical trend is to let the user control the color and opacity specifications [31, 35]. Especially the color selection is mostly left out as a user task. However, this requires some visualization knowledge which the end user may not possess. As stated before, in our proposed method, we calculate the opacity and color values automatically from the user selection of voxel groups. The opacity value is generated based on the variance of a group, since variance is related to visibility [20]. For the color values, we take help of *harmonic colors* [48, 45]. Harmonic colors are a sets of colors, the internal relationship among which provides a pleasant visual perception [49]. Since our visual experience is largely dependent on the represented colors and their relationships, using color harmonics can result in a much better visual output than using user-defined colors. We also assign the saturation and brightness of each voxel based on its properties. The combination of feature-based parameterizations and color harmonics takes the difficult task of searching through a virtually infinite color space away from the user. In this way, by simply interacting with the SOM lattice, a user can generate the desired rendering quickly and efficiently.

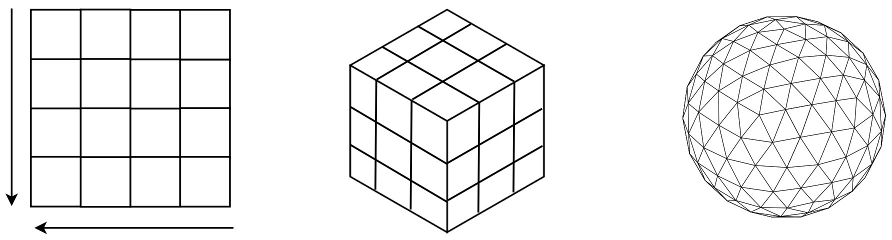
In summary, our proposed system has the following primary contributions:

- Rather than manipulating cluster parameters or optical properties, the user simply interacts with a color-coded SOM lattice representing cluster densities. Due to this visual nature of SOM, there is no need to tweak the cluster parameters and perform operations like split and merge to precisely determine the number of clusters or cluster spread. The user only has to intuitively select or de-select the SOM regions to reveal corresponding structures in a volume.

- The user also does not need to manipulate color and opacity properties. The opacity and color values are automatically generated by combining various voxel group properties with aesthetically pleasing color harmonics. As a result, desired rendering can be achieved quickly and efficiently.

- The proposed model is independent of the type of the features used. As a result, new types of features can be integrated easily and efficiently. Based on the application domain in which the system will be used, the type of feature can be changed. Also, since the feature space is not represented directly like widget-based TF methods, the proposed model is not restricted by 2D/3D features. Higher-dimensional features can also be used.

We use the Spherical SOM structure (SSOM) [50, 51, 52] because of it's restriction-free structure (explained later) . The GPU implementation of our method provides fast interaction



(a) 2D (open boundaries) (b) Cube ("forced" continuity) (c) Spherical (continuous)

Figure 3.1: Depiction of different SOM lattices.

with the SSOM and real-time volume rendering.

3.2 Self-Organizing Map

The Self-Organizing Map (SOM) is an unsupervised clustering method proposed by Kohonen [17] that clusters the data and projects data points onto a lower-dimensional space while preserving the input topology. The data points are projected onto a regular lattice during training, when the weights of the nodes in the lattice are updated. In this way, after training the initial lattice "self-organizes" itself to represent the data distribution. Different coloring methods are then applied on the lattice to color code it so that the cluster regions can be visualized. As stated before, the topology-preserving clustering and easy visualization properties of the SOM makes it an attractive choice for TF generation.

The underlying lattice structure is primarily responsible for the visual representation of the SOM. The SOM learning occurs due to lateral interactions among nodes in the underlying lattice. A discontinuity in the lattice will result in a restricted neighborhood and can lead to inefficient learning [51].

The most popular lattice structure is the 2D SOM, where the nodes are represented by a sheet of connected nodes (Figure 3.1-(a)). Since the SOM training is based on the lateral connections among the nodes, a 3D structure provides an additional dimension of connections and therefore offers increased learning as compared to a 2D lattice [50]. Commonly used 3D SOM consists of the cube lattice structure (Figure 3.1-(b)).

However, all these structures have a restricted neighborhood problem. In [50] it has been shown that the 2D SOM fails to accurately approximate a point cloud due to its open boundaries. It has also been shown that the 3D cubic structure has a rather "forced" continuity, which

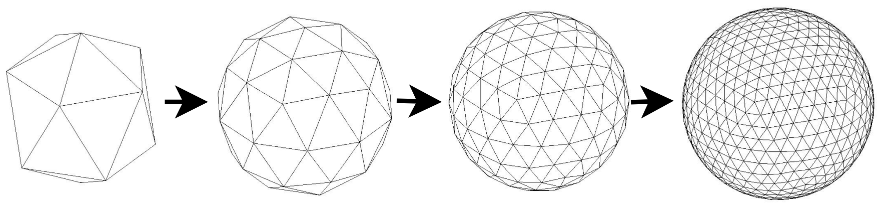


Figure 3.2: The first three levels of subdivision of the basic icosahedron.

results in an unwanted folding effect. These problems were perceivable due to the use of 3D point clouds in [50]. But for higher-dimensional data these kind of training errors will be difficult to detect.

An optimum lattice structure should offer minimum topological discontinuity, as is the case of a spherical structure. The spherical lattice is created by subdividing an icosahedron to obtain a uniform distribution of nodes resulting in a structure with an overall symmetry and continuity. From the basic icosahedron, higher resolution lattices can be created by bisecting every edge of the icosahedron and radially projecting the new vertex to lie on a sphere of unit radius [51]. All the new vertices will still maintain the equilateral triangular structure of the basic icosahedron. The first three levels of subdivision of the basic icosahedron are depicted in Figure 3.2. In this way, the lattice will have a continuous structure with minimum discontinuity among nodes, which can result in better separation among the clusters.

There are several variations of the spherical lattice depending on the underlying data structure used to represent the sphere and the visualization technique used, such as the hyperbolic SOM [53] or the Geodesic SOM [54]. We've used the version introduced in [51] mainly because of it being visually more intuitive and easier to perceive [50].

The next three sections contain discussions on different aspects of the proposed system based on the Spherical SOM (SSOM) structure: 1) Feature extraction, where we discuss about the features used to model a volume for TF generation; 2) SSOM training, where the detailed steps of the Spherical SOM training are described and 3) TF representation and exploration, where we discuss our opacity and color assignment for voxel groups based on color harmonization and the user's perception. We also discuss in detail how simply and efficiently the user can explore our color-coded SSOM lattice to find interesting regions and group voxels together. The color and opacity values are generated accordingly with immediate rendering results through GPU implementation.

3.3 Feature Extraction

As discussed before, multi-dimensional TFs mostly use some form of histogram based on intensity and gradient magnitude. However, one problem with histogram is that it cannot retain the spatial information present in a volume [31]. As present day scanning systems for volume data (CT, MRI etc.) have some inherent noise associated with them, losing spatial information might prove to be costly and may not cluster the volume data in a suitable way for volume rendering. On the other hand, incorporating the spatial information directly is difficult for any histogram-based approach, as the histogram will be even higher-dimensional and there is no easy way to represent it visually. As a result, in most histogram-based approaches, the spatial information is used for color generation only [31]. However, as described before, our proposed model can incorporate higher-dimensional features due to the use of SSOM lattice. As a result, in our proposed model, the spatial information is directly embedded into the feature definition. To emphasize the boundaries between materials [16], we also use the intensity value of each voxel, it's 3D gradient magnitude and the second derivative (obtained through eigenvalues of the Hessian matrix), all of which are popular features for traditional histogram-based TF design [16]. Our 6D feature set for each voxel consists of the following features:

- The X,Y and Z coordinates,
- the intensity value,
- the 3D gradient magnitude. The 3D gradient magnitude for each voxel is defined by:

$$G = \sqrt{G_x^2 + G_y^2 + G_z^2}, \tag{3.1}$$

  where G_x,G_y and G_z are the gradient values along X, Y and Z direction, respectively.

- The second derivative. This feature can be calculated from the Eigen values of the Hessian matrix [55]. The Hessian matrix can be defined as:

$$H = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}, \tag{3.2}$$

  where $I_{xx}, I_{xy}, \ldots, I_{zz}$ are the second partial derivatives. The sum of the three eigenvalues of this matrix $\lambda_1 + \lambda_2 + \lambda_3$ (where λ_i represents one eigenvalue) roughly approximates the information obtained from the second derivative of the intensity values [55]. This sum value is used as a feature in our experiments.

All the features are scaled to fall in the interval of $\{0,1\}$. Ideally, we would still like to emphasize the boundaries of materials over spatial similarity. Hence, the used features are weighted. For our experiments, we use a weight set of $\{0.5, 0.5, 0.5, 1.5, 3, 2\}$ for the aforementioned features. These values were picked by a trial and error method. For the trial-and-error process, the first three weight values were always set to 0.5. The weight values pertaining to the gradient magnitude and the second partial derivatives were changed within the range of $1-5$ with an increment of 0.5. The set of values that resulted in the best separation of clusters (assessed visually) were used. The trial-and-error process was only carried out for one dataset. It was observed that minor changes to these values does not have any adverse effect on the results. In fact, during our trial-and-error method the values in the close vicinity of the used set resulted in almost identical separation of clusters. As long as the boundary features (gradient magnitude and second partial derivatives) are reasonably emphasized, changes in these values does not significantly effect the result. Hence, we used the same set of values for all datasets.

3.4 SSOM Training

The training phase of the SSOM is similar to the classical SOM [17]. Let, the input space of N voxels be represented by $\mathscr{X} = \{\mathbf{x}_i\}_{i=1}^{N}$. Let, the SSOM be represented by M nodes ($M << N$). Each node in the SSOM lattice has a corresponding weight vector $\boldsymbol{\varpi}$. All these weight vectors together represent our SSOM space $\mathscr{W} = \{\boldsymbol{\varpi}_i\}_{i=1}^{M}$. Each node also has a *neighborhood* associated with it. A neighborhood is a set of nodes consisted of the node itself and its neighbors. Let, the neighborhood set for node i be represented by Θ_i^r. Here, r represents the neighborhood spread, $r = 1, \ldots, R$. R is the maximum neighborhood radius, which is set to a value such that it covers half of the spherical space [50].

The input voxels are randomly introduced to the SSOM during training. For each voxel, a *Best Matching Unit (BMU)* among all the nodes is selected. BMU is the node which is closest to the input voxel according to some similarity measure. Euclidean distance is used as distance measure. The update step then takes place, where the weight vector of the BMU and it's neighboring nodes (Θ_{BMU}^r) are updated in a way so that they are pulled closer to the weight of the input voxel. After training, the SSOM weight vectors are arranged in such a way that represents the underlying distribution of the input data (the voxel features in this case).

Besides using the traditional SOM parameters for training, a count-dependent control over how the winning node is being chosen is also used in our system [50]. This parameter is inspired from *Frequency-Sensitive Competitive Learning* (FSCL) [56]. FSCL addresses the problem of nodes of the lattice being under-utilized by introducing a measure of how frequently a node is being selected as the BMU. Each node has a count c_i associated with it (see the *Weight Update* step below), which increases when the node wins. This count in turn is used with the distance measure to "penalize" the winning node i.e. increase its distance measure for the next input vectors [56]. This is especially necessary in our case because in a volume, typically there is almost $80\% - 90\%$ homogeneous regions. But the other $10 - 20\%$ is usually more important, since they are the boundary voxels. Without a count-dependent control, the boundary voxels will be assigned to only a few winning nodes frequently. As a result, eventually homogenous regions will take over the whole map and important regions (boundaries) will not get enough map space to be noticed. In the Experimental Results section, we provide an analysis of the effect of count-dependent parameter on the clustering and the eventual rendering of a dataset.

The step-by-step training algorithm is described below:

- **Initialization:** The weight vectors of the SSOM nodes are initialized first. Random values can be used for initialization, but as pointed out by Kohonen et al. in [17], random initialization will take more time to converge. We have followed the initialization method stated in [17] i.e. initialize the weight vectors with values that lie on the subspace spanned by the eigenvectors corresponding to the two largest principal components of the input data distribution. As explained above, a count vector $\mathscr{C} = \{c_i\}_{i=1}^{M}$ [50] is used to keep track of the hits to each node. This vector is initialized to zero. This is used in the BMU selection step to prevent cluster under-utilization.

- **Training:** For each input voxel $\mathbf{x}$, do the following:
  - **BMU Selection:** Calculate the Euclidean distance of the voxel feature vector $\mathbf{x}$ with all the nodes as follows:

    $$e_i = (c_i + 1)\|\mathbf{x} - \varpi_i\|, \quad i = 1, \ldots, M. \tag{3.3}$$

    BMU is the node for which this distance is the smallest.

– **Weight Update:** Update the weights for the BMU and it's neighboring nodes (defined by Θ^r_{BMU}) as follows:

$$\varpi = \varpi + b(t) * h(s,r) * \|\mathbf{x} - \varpi\|, \tag{3.4}$$

$$c_\varpi = c_\varpi + h(s,r), \tag{3.5}$$

where $\varpi \in \Theta^r_{BMU}$, $b(t) = \alpha e^{-\frac{t}{T}}$ and $h(s,r) = e^{-\frac{r^2}{s*R}}$.

The functions $b(t)$ and $h(s,r)$ control the rate of learning and the neighborhood effect, respectively. $b(t)$ decreases in value as the iteration number $t = 1,2,\ldots,T$ increases. It also depends on the learning rate α. $h(s,r)$ depends on the neighborhood size parameter s, which is user controlled. $h(s,r)$ is a Gaussian function. The further a neighboring node is from a BMU, the less it's weight will be affected.

As discussed before, the count-dependent parameter c_ϖ is increased here to prevent cluster under-utilization. Observing Equation 3.3 and Equation 3.5, we see that the BMU and its neighbors are penalized for winning by increasing the count. In the next update step, the distance measure with these nodes will be higher due to the increase of count. This ensures that one node (or its neighbors) does not win too many times, and the entire map is utilized [56].

- Repeat the training steps for a pre-defined number of iterations (T).

The main control parameters in SSOM training are the learning rate α, the number of iterations T and the neighborhood size parameter s. In case of volume rendering, we are dealing with a huge number of voxels (e.g. for a volume of dimension $256X256X256$, there are over 16 million voxels). Among this voxels, only around $10-20\%$ are boundary voxels, the others represent mostly homogeneous regions. For the voxels belonging to homogeneous regions, the features are very similar. As we will show in the experimental results section, for such high number of voxels with similar features, the SOM typically stabilizes reasonably within only $1-2$ iterations. The α is set to 0.1 in our experiments. The neighborhood size parameter is set to $s = 2$, which is determined through trial and error.

3.5 TF Representation and Exploration

After training of the SSOM is completed, the weight vectors associated with the nodes of the SSOM represent the underlying clustering of the voxels. We map this SSOM to a suitable TF in three steps: 1) present a color coded graphical representation of the SSOM, 2) provide the user with interaction options to select group interesting regions in the SSOM together, and 3) map selected regions of the SSOM to a suitable TF and show the rendering of the volume with the generated TF. Due to our GPU implementation, the user can see the rendering of the volume in real-time while interacting with the SSOM.

To color code the spherical lattice, the U-Matrix approach is used [50], which provides a visual form of the distance between the weight vectors of the nodes. For each node, the average Euclidean distance of its weight vector with the weight vectors of all the immediate neighboring nodes is calculated. These distance measures are then mapped to a color-map for visualization purposes. In this way, a homogeneously colored region will represent a cluster, while the cluster boundaries will incur a change in coloring. An important point to note here is that since volume rendering is an entirely perceptual process, it is not important to strictly define how many clusters we have or whether the cluster boundaries are very well defined or not. The important point is to color the SSOM lattice in such a way that intuitively interacting (explained below) with it will directly result in meaningful rendering. Representation of cluster densities in the form of color-map through U-matrix serves this purpose.

Our target is to keep the user interaction as minimum and efficient as possible. However, only the user has the domain knowledge to know what is important in the volume dataset. Therefore, we only leave the selection of voxels to the user. In our experiments, we use the third-level subdivision of the basic icosahedron (Figure 3.2) which has 642 nodes. For a typical volume, usually the user is interested in only a few groups (2-3) of voxels. For example, a CT scan primarily retains the bone structures. The user will ideally be interested in grouping the voxels based on different bone densities or regions. All our SSOM nodes correspond to some number of voxels (the voxels for which the node was selected as BMU). The U-Matrix based coloring represents existence of potential clusters, but we leave the final decision of grouping the nodes and defining them as a single group to the user. We provide the user with control to group the SSOM nodes together to define different *group of voxels*. The user performs this by dragging a rubber-band control on top of the visualized SSOM lattice. The user is immediately presented with an intermediate rendering based on basic pre-defined colors to provide a rough idea of which voxels were selected as a result of his or her action. When the user is satisfied with the selection, he or she can define the current selection as one group, and do further selections for the next group. As soon as a group is defined, the rendering is updated to show the final result based on the properties of the voxels in the group and color harmonization. After a group is finalized, the visualization for it can be turned on/off temporarily through the user interface, so that the user can do further node selections without any hindrance [1].

For the intermediate rendering, since we do not have a definition of a group yet, we simply map the color channels to different features. One channel is mapped to intensity, while the other two channels are mapped to the 3D gradient magnitude. After a group is defined, the final color and opacity values are generated according to the rules described in the next section.

3.5.1 Color and Opacity Assignment

After the user defines the separate voxel groupings on the SSOM lattice, we generate the color and opacity values to create the final rendering. We've put special attention to the color selection of different groups. In most of the existing methods, the colors are often determined by adhoc assignment or personal preference [20]. Since volume exploration is a lengthy and complex process, visual aesthetics is very important. The contrast among the colors is also important to differ among different types of materials (voxel groups in our case) efficiently. Hence, we apply the theory of color harmonization [49] for our color assignment. Harmonic colors define sets of colors that are aesthetically pleasing to the user [49]. In [44], a color wheel was introduced in which color harmony is described across color hue values. A harmonic set is described by specifying the relative position of the colors on the wheel rather than specific color themselves. The harmonic colors are defined based on user evaluations rather than strict mathematical formulations. Based on this color wheel, several templates can be defined. Figure 3.3 shows the four templates we've used in our method, namely, Type V, Type L, Type T and Type X. All this templates can be rotated at arbitrary degrees to generate a new set of colors (please refer to [48] for the detailed specifications on these templates).

[1]The reader is encouraged to see the video demo at `http://goo.gl/Z4NJD` for a better understanding of the whole process

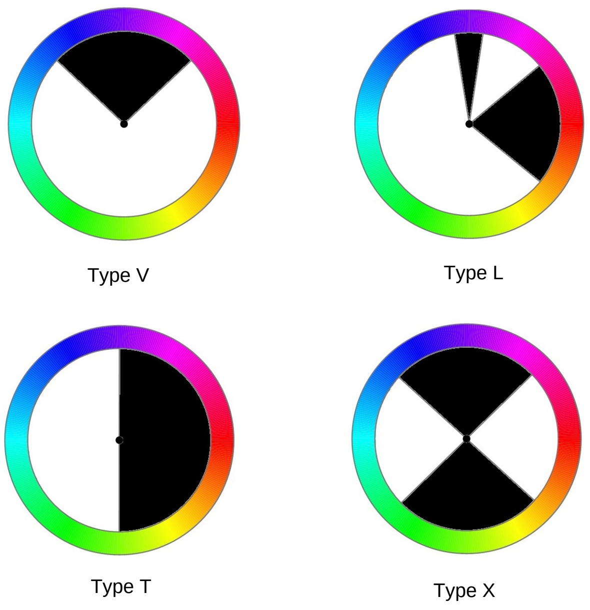


Figure 3.3: Hue Templates

We use these templates to generate the hue values for *HSV* color space. We need separate hue value for each group of voxels selected by the user. We equally space the required hue values along the available hue range so that we've a visually pleasing result as a combination of all of them.

However, if we simply use one hue value for each user selected group, the details will not be revealed. For volume rendering, boundary enhancement is important to reveal the inner structures [16]. As a result, instead of using one hue value, we use a small range with one lower and one higher limit, ξ_l^i and ξ_h^i for each group. For our experiments, the difference between ξ_l^i and ξ_h^i is 5% of the whole hue wheel (or 18°). This still gives us enough hue range to space out all the groups (typically 2-3 as described before) reasonably around the hue templates.

The hue value for each voxel of each group is calculated by the following equation:

$$H_i^v = \xi_l^i + (\xi_h^i - \xi_l^i) * \digamma_v, \; i = 1, \ldots, Z, \tag{3.6}$$

where Z denotes the total number of groups selected by the user. $\digamma_v$ is the *Gradient Factor* and is defined as:

$$F_v = \frac{G_v}{max_{v \in \mathscr{X}} G_v}. \tag{3.7}$$

Here, $\mathscr{X}$ denotes the group that the voxel v belongs to. G_v is the 3D gradient magnitude for each voxel (defined in Equation 3.1). In this way, even for a single group, the hue values will have some variation based on the gradient value instead of having a single hue value assigned to all of them.

In the HSV color space, the S and V values are also very important to define a color. The S and V values respectively determine the saturation and brightness of the color. We generate the S and V values based on the understanding of the user's perception [20]. Voxel groups that have a small spatial variance occupy a smaller viewing area compared to groups that have relatively larger spatial variance. We can intuitively assume that the groups with smaller spatial variance needs to have a more saturated color to highlight them properly [20]. Hence, we calculate the S value for each voxel group according to the following equation:

$$S_i = \frac{1}{(1 + \sigma_i^2)}, \ \ i = 1, \ldots, Z. \tag{3.8}$$

Here, σ_i^2 represents the spatial variance of each voxel group.

Similarly, since the rendering results are usually viewed at a distance from the whole volume, we can assume that voxel groups closer to the center of the volume needs to be brighter so that they are not overshadowed by groups that are further from the center [20]. Hence, The V value for each voxel group is calculated as follows:

$$V_i = \frac{1}{(1 + \mathbf{D}_i)}, \ \ i = 1, \ldots, Z. \tag{3.9}$$

Here, $\mathbf{D}_i$ denotes the distance of the centroid of the $i-$th group to the center of the volume [20]. By assigning the saturation and brightness values this way, we can assign more vivid colors to the voxel groups that are relatively more difficult to highlight.

The last parameter to define the full TF is opacity. Since voxel groups with smaller spatial variances are likely to be obstructed for proper viewing by groups with larger spatial variances, we calculate the opacity values based on spatial variances. We also want to emphasize the boundaries to reveal the detailed structures. Hence, our opacity values for each voxel are calculated using the following equation:

$$O_i^v = \left(1 - \frac{\sigma_i^2}{\max(\sigma_i^2)}\right) * (1 + \digamma_v), \;\; i = 1, \ldots, Z. \tag{3.10}$$

Here, σ_i^2 is the spatial variance of the $i-$th voxel group. $\digamma_v$ is the Gradient Factor as defined in Equation 3.7. Inspecting Equation 3.10, we can see that voxel groups with smaller spatial variances will be assigned higher opacity values. Moreover, all the opacity values will go through boundary enhancement by being multiplied by $(1 + \digamma_v)$. The opacity values are scaled to fall between 0.2 and 1.0 to ensure that none of the voxel groups become completely transparent.

After calculating the *HSV* and opacity values, we convert them into the final *RGBA* texture. The *HSV* triplets are converted into *RGB*, and the opacity value is appended to generate the *RGBA* quadruples. This is passed to the rendering stage to generate the final rendering.

For the rendering stage, we use the Visualization Toolkit (VTK) [57]. The *RGBA* texture passed to VTK is rendered in GPU. As a result, the user can see the rendering changes in real time while selecting and assigning voxels into different groups.

3.6 Experimental Results

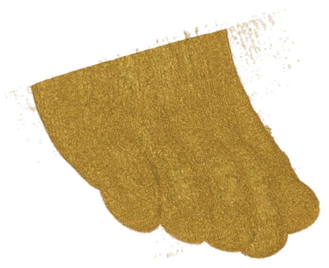
In this section, we provide detailed analysis and results of our proposed system on several volume datasets. We used five volume datasets [58], CT scans of a Foot, the Visual Male Head dataset, the Carp dataset, the Engine dataset and the Lobster dataset. Generally, volume datasets have a lot of vacant regions (air around the object). If we build the SSOM directly on the dataset, these regions will generate a misleading clustering. A simple region growing algorithm [59] can be used to separate the object from these regions first, where a voxel in the vacant region can be used as seed to separate the object from the background. However, the voxels in the vacant region typically have zero (or very close to zero) intensity values. Hence, a simple thresholding with a low intensity value can separate the background. In practice, the region growing option provides marginally better separation. But the thresholding method is several times faster than the region growing approach. Hence, in our experiments we opted for the thresholding method to separate the object from the vacant regions. These separated voxels were then fed to the SSOM training algorithm.

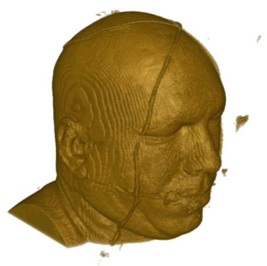
After the SSOM training, the user is presented with a visual form of the spherical lattice color coded with the U-matrix approach as described before. The user can then perform selection/de-selection on the lattice, and the corresponding voxels are shown with an intermediate rendering. When the user is satisfied with the selection, he or she can assign the current selection as a group of voxels. This group is then assigned the color and opacity values according to the equations described in Section 3.5.1 for final rendering. The user can then move on and select new group of voxels. At each step of group definition, the rendering is updated to reflect the new color and opacity values. To see how simple and intuitive the whole process is, we highly recommend the reader to see the video demo (located at `http://goo.gl/Z4NJD`). The video demo shows a brief walk-through of the whole system, especially emphasizing on the effectiveness of our color and opacity assignment techniques.

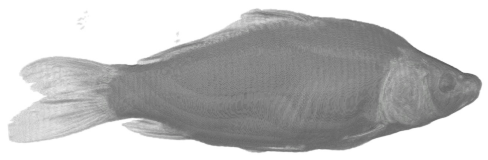
Figure 3.4 shows intermediate renderings of all the datasets when no grouping is performed (i.e. all the nodes are treated as a single group). Please note that the color assignments for these intermediate renderings are arbitrary. The color channels are mapped to different features and scaled to fall at random color regions to generate these renderings. These renderings are presented here to provide the reader with an idea on how these datasets might look without any voxel grouping or color processing. As we will show subsequently, our new color and opacity assignment based on color harmonization and the user's perception can reveal the details easily and efficiently.

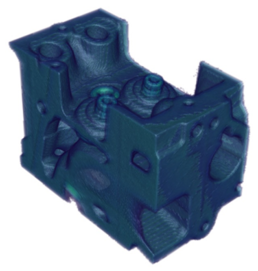
First, we demonstrate the effect of the count-dependent parameter defined in Section 3.4. As described before, due to the dominance of homogeneous regions in typical volume datasets, we need to make sure that the boundary voxels are getting enough map space. Hence, the count-dependent parameter is used so that cluster under-utilization is eliminated and boundary voxels are not overshadowed by other voxels [56].

Figure 3.5 shows the intermediate renderings of the voxels corresponding to the user selection, both with the count-dependent parameter and without it. As we can see, without the parameter, the voxels representing the inner structure of the bones are not converged into a cluster properly. Other homogenous voxels also become part of the selection and the rendering is unclear. By using the count-dependent parameter, the voxels can be grouped in a much easier way, and the user selection clearly corresponds to the inner structures. For this reason, we used the count-dependent parameter in our training algorithm.


(a) Foot


(b) Visual Male Head


(c) Carp


(d) Engine

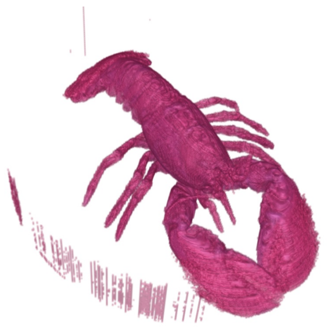

(e) Lobster

Figure 3.4: Intermediate rendering of all the datasets without any voxel grouping or color harmonization.

Next, we show the empirical convergence behaviour of the SSOM with the Foot dataset. As stated before, typical volume datasets have a small number of (around $10-20\%$) boundary voxels. The other voxels belong to homogenous regions, and have similar feature characteristics. As a result, the SSOM stabilizes very quickly, within $1-2$ epochs. Perfect convergence is also not necessary for our system, as the final result is visual rather than quantitative.

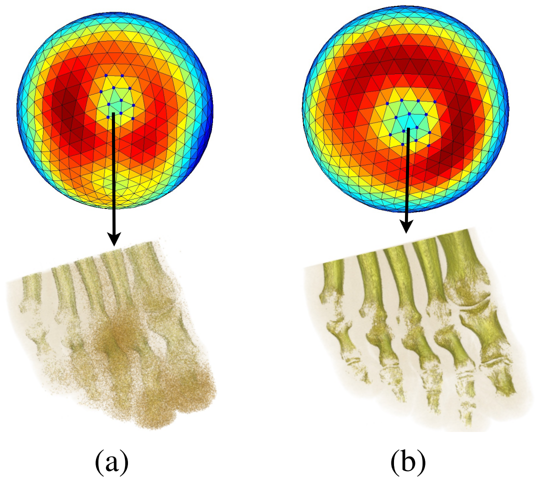


Figure 3.5: Intermediate rendering of the Foot dataset corresponding to user selection when the training is performed a) without the count-dependent parameter. b) With the count-dependent parameter.

Figure 3.6 shows a plot of the change in the weight vector as number of epochs increases [50]. This experiment was performed on the Foot dataset. The change in the weight vectors is obtained as a scaled distance between the current weight vectors and the previous one. As we can see, after Epoch 1, the change in weight vectors is not significant. The map does not necessarily converge, but we can say that the map is “stable” enough for our system, as the changes are only minor after the first iteration.

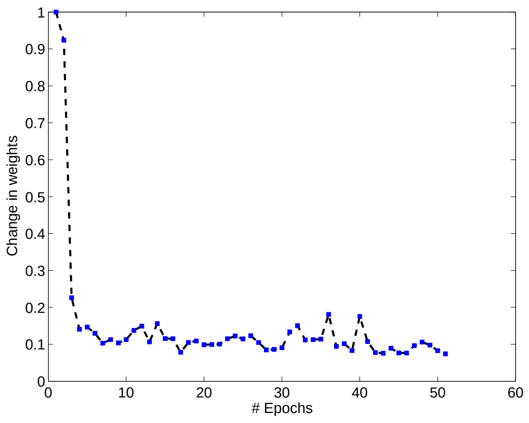


Figure 3.6: Convergence characteristic of the SSOM for the Foot dataset.

Figure 3.7 show the visualization of the SSOM lattice for the Foot dataset after one and five epochs, respectively. As we can see, even visually the maps are almost similar with minor differences. This again justifies our claim that 1-2 iterations of the training algorithm is good enough for the stabilization of the map.

Finally, we show the usefulness of our system in action. Figure 3.8 shows the final render results of different voxel groups defined by the user. The voxel groups can be seen on the spherical lattice, with different groups of nodes separately colored (white, blue and black dots). As can be seen, due to high-dimensional feature clustering, the structures in the dataset can be revealed quickly and efficiently. Group 1 (corresponding to the nodes colored in white) reveal the inner structures of the bones, while Group 2 (corresponding to the nodes colored

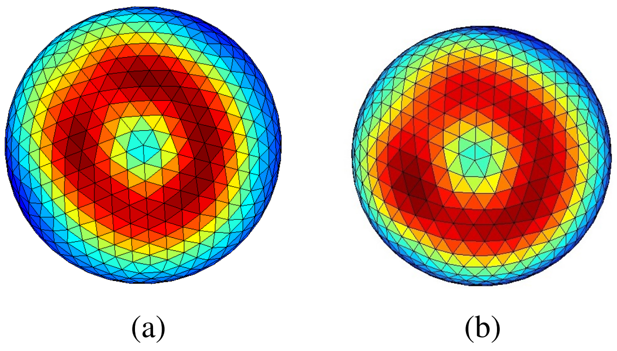


Figure 3.7: SSOM visualizations for the Foot dataset a) after one epoch. b) After five epochs.

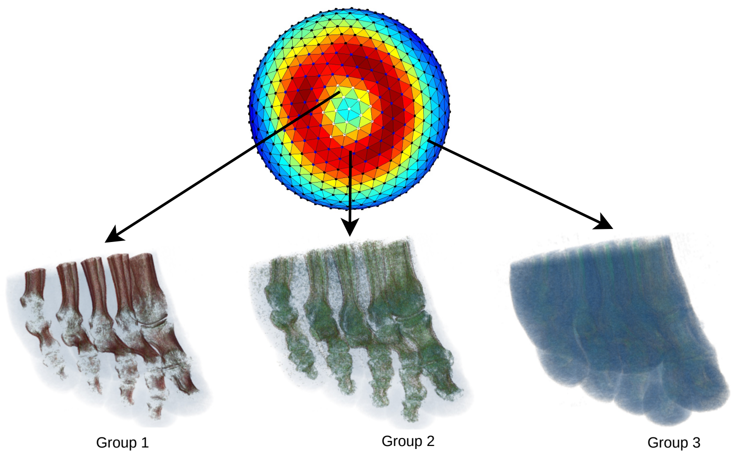


Figure 3.8: Different user-selected voxel groups and corresponding renderings of the Foot dataset. White, blue and black dots refer to Group 1, Group 2 and Group 3 (pointed out by the arrows), respectively.

in blue) emphasize the joints among the bones. All the other voxels are assigned to Group 3 (corresponding to the nodes colored in black), which roughly represents the outer layer of the foot.

The effectiveness of our color and opacity assignment techniques can be seen in Figure 3.9. Figure 3.9-(a) shows Group 1 and Group 2 together, while Figure 3.9-(b) shows all 3 groups together. As can be seen, due to using color harmonization, all these groups are distinguishable even when they're rendered on top of each other. For this dataset, we used the T-type template (Figure 3.3) with default setting i.e. starting at $0°$. Since our opacity assignment depends on the spatial variance (Equation 3.10), the inner groups of voxels are assigned higher opacity values (less transparent). Moreover, due to the use of boundary enhancement with gradient factor, the resulting renders are revealing the structures clearly.

Another interesting observation is the spread of the voxel groups selected by the user. In the map, the blue colors suggest the existence of a cluster, while a shift towards red denotes change of cluster. However, as we can see from Figure 3.8, Group 2 is defined around group 1 and on a red patch in general. Ideally, a perfect combination of SOM algorithm and feature set should separate the clusters very clearly. But as we stated before, volume rendering is a complex process and the principal quality assessment is how the user perceives the output. If we picked the voxel groups automatically based on cluster density, a desirable output will be difficult to achieve. But by leaving the final group definition as a user task, we provide the user

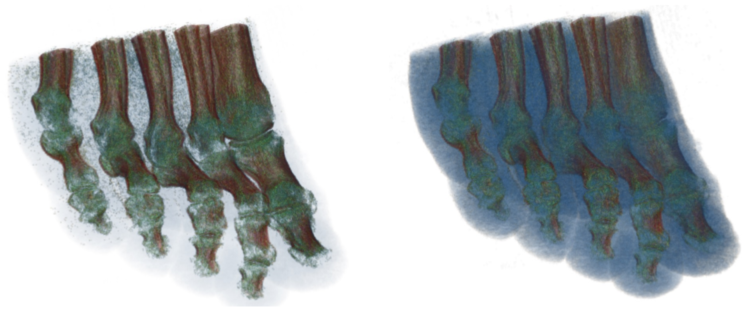


Figure 3.9: Rendering of the Foot dataset a) Group 1 and Group 2 together. b) All Groups together.

with enough flexibility, but hide the complexity of feature extraction, voxel correspondence, color and opacity assignments by making them automated. The only knowledge the user needs is the detection of a small cluster. Typically, a small cluster (blue patch) surrounded by a shift towards other clusters (red patches) suggest that a small group of voxels reside in this cluster which are different from other regions of the volume. So the user can work his/her way out from there and define the voxel groups as he or she wants to see it. With this flexibility, our system provides a robust platform to generate different kinds of renderings from a single dataset.

Figure 3.10 shows the results from our next dataset, the Visual Male head. X-type template with default setting (i.e. starting at 313.2°) was used for this dataset. Here, we see that there are two user-defined voxel groups. Group 1 (corresponding to the nodes colored in white) reveal the detailed skull structure. Group 2 consists of the other voxels to represent the outer layer (corresponding to the nodes colored in black). Here, the skull structure is of interest, which can be separated very easily through our system. Figure 3.11 shows the two groups on top of each other. Again, as we can see, due to our intelligent color and opacity assignments, the outer layer is more transparent and the nested features are still visible.

Figure 3.12-3.13 shows the results of the other datasets. These renderings were obtained with 2 user-defined groups and the following hue settings:

- Carp: L-Type hue template starting at 144°.
- Engine: V-Type hue template starting at 90°.
- Lobster: T-Type hue template starting at 18°.

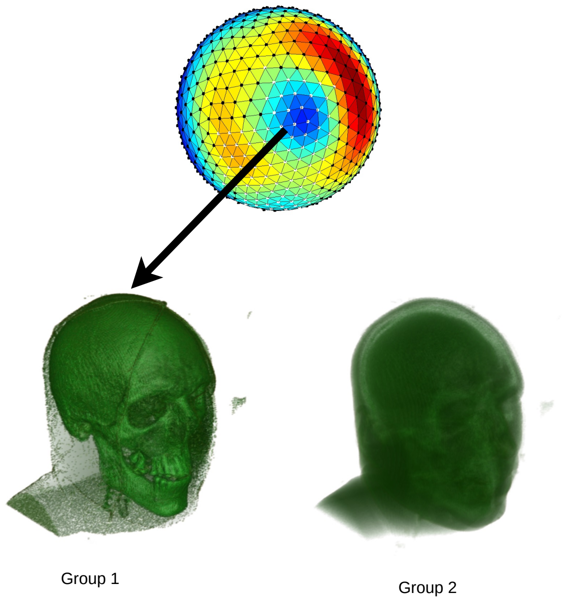


Figure 3.10: Different user-selected voxel groups and corresponding renderings of the Visual Male Head dataset. White and blue dots refer to Group 1 and Group 2 (pointed out by the arrows), respectively.

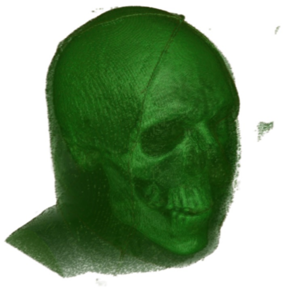


Figure 3.11: Rendering of the Visual Male Head dataset (all groups together).

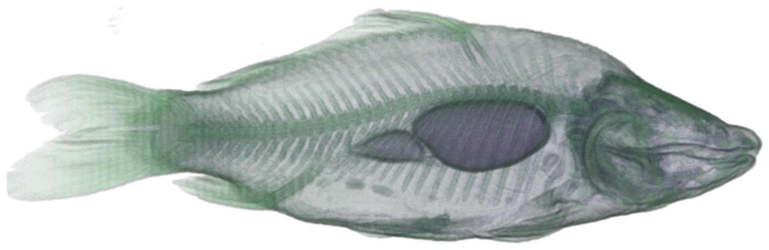
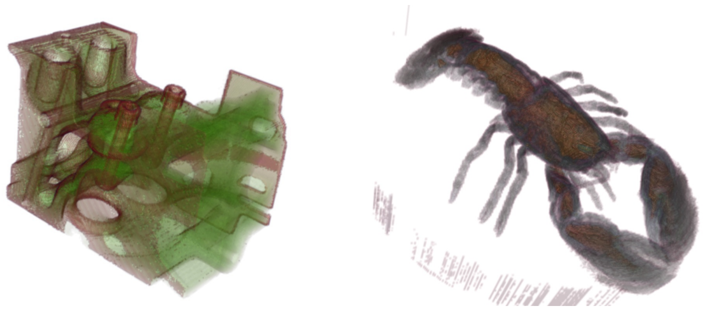


Figure 3.12: Rendering of the Carp dataset with 2 groups.



(a) (b)

Figure 3.13: (a) Rendering of the Engine dataset with 2 groups. (b)Rendering of the Lobster dataset with 2 groups.

Comparing these results with the renders in Figure 3.4 (obtained without any grouping or automatic color assignment), we can see that the flexibility of user-defined voxel grouping on the SSOM lattice and the robustness of our automatic color and opacity assignment result in the exposure of detailed structures in all the datasets.

The size of each dataset, training times and number of training iterations used are listed in Table 3.1. Please note that the training of a SSOM has to be done only once for each dataset and does not effect the rendering process of our proposed method, which is required to be real-time. Also, as shown before, due to the high number of voxels with similar features, $1-2$ iterations of training is good enough for stabilization of the map. Such low number of iterations is reasonable here, since the perfect convergence of map is not very critical, as long as we can have a color-coded SSOM where different cluster regions and the borders between them are visually distinguishable.

Table 3.1: The size of the volume datasets, number of iterations for training and the required SSOM training times (in *seconds*).

Dataset Name	Size	Iterations	Training Time
Foot	256X256X256	1	442.5
Visual Male Head	128X256X256	2	1524.6
Carp	256X256X512	2	2395.9
Engine	256X256X256	1	369.7
Lobster	301X324X56	1	111.11

To further strengthen the usability of our system, we conducted a user survey, where we compared our proposed system with an off-the-shelf traditional TF editor [60]. A point to note here is that to our knowledge, there is no existing method to quantitatively compare volume rendering techniques, since the final interpretation is user dependent. Some measures have been proposed recently to tackle this problem [61]. Even those measurements had to undergo extensive user studies [61] to justify their applicability. The problem with evaluation of volume rendering systems lies in the fact that there are too many different parameters from system to system to come up with a fair and viable comparison platform. Most of the proposed methods make use of their own unique interfaces to make the user interaction easier and more efficient. The TF generation process also varies significantly. As a result, we have resorted to the commonly accepted user satisfaction and efficiency as comparison tools [62, 63].

For our user survey, 10 users were carefully chosen to represent varying level of knowledge and familiarity with visualization tools and techniques. Users 1-5 are the most familiar with graphics tools and techniques. User 6-8 have had previous experience in using complex software tools, not necessarily graphics-related. Users 9-10 have little to no experience with similar tools. There were two phases in the survey:

- **Training**: In this phase, the users were familiarized with the interface of the systems. This phase was continued till the user was comfortable with the various options (e.g. selecting voxels, defining voxel groups in our proposed system and selecting points from histogram in the traditional TF editor).
- **Evaluation**: The users were provided with the raw visualization of the Foot dataset without any voxel grouping (Figure 3.4-(a)) and a reference image with three voxel groups (Figure 3.9-(b)). The task was to play with the different options in the system and gener-

ate a similar rendering as the reference image from the raw visualization. In our proposed system, the color and opacity values are automatically generated through color harmonization. For the traditional TF editor, the user was free to chose arbitrary color and opacity values. Since the reference image was generated from our system, to be fair to the traditional method, matching the color and opacity was not a requirement, as long as all three voxel groups were distinguishable to a satisfactory level for the user.

During these two phases, the time required was recorded for each user (Table 3.2). After the evaluation, the users were asked to rate each system on a Likert scale [64] of 5, with 5 being the best (completely satisfied) and 1 being the worst (not satisfied). There were two categories for rating, the interaction with the system and the quality of the output.

Table 3.2 summarize the results from this user survey. As we can see, our proposed method performs significantly better in all aspects. For the training phase, we see that all users needed more time to adapt to the traditional TF editor than to our proposed system. The time difference is especially significant in case of Users 9-10, who happens to be the least experienced with visualization systems. To interact with the traditional TF editor, you need some knowledge of histogram and color/opacity mapping. Since these users were not familiar with these concepts, it took them more time to adapt. However, our proposed system is simple and intuitive. No low level parameter is exposed to the user. As a result, the training for our system was faster. Even with other experienced users (1-8), we see that the training of our proposed system was faster.

In case of interaction time, we see that our proposed system could generate satisfactory output very quickly, while with the traditional editor it took more time. With the traditional editor, on top of histogram manipulation, the user had to assign the color and opacity values manually. As a result, it took longer to generate a satisfactory output.

The same improvements are noticeable in case of user ratings. We see that our system got an average rating of 4.4 for both interaction and output satisfaction, while the traditional system only got 3.35 and 2.95 respectively. The traditional TF was especially unsatisfactory in terms of output quality. This is expected, as the traditional editor only makes use of intensity histograms. Our proposed system uses high-dimensional features. As a result, the voxel groups can be identified easily and efficiently. In fact, the users were not able to differentiate the 3 voxel groups in the Foot dataset accurately with the traditional TF editor. As we can see in Figure 3.14-(c), the voxel group near the bone joints are not distinguishable. But with our

Table 3.2: Times recorded for user training and interaction (in *seconds*) and the ratings (on a scale of 5) by the users for interaction and output quality (proposed method in **bold**).

User	System	Training Time	Interaction Time	Interaction Rating	Output Rating
User 1	Proposed	**110**	**130**	**4.5**	**4**
	Traditional	170	250	3.5	2.5
User 2	Proposed	**90**	**125**	**4**	**5**
	Traditional	150	190	3	3
User 3	Proposed	**120**	**130**	**4**	**4**
	Traditional	177	200	3.5	3.5
User 4	Proposed	**107**	**140**	**4.5**	**4**
	Traditional	150	220	4	3.5
User 5	Proposed	**100**	**100**	**5**	**4**
	Traditional	142	198	3	3
User 6	Proposed	**100**	**130**	**5**	**4.5**
	Traditional	210	340	3.5	3
User 7	Proposed	**140**	**140**	**4.5**	**4.5**
	Traditional	220	310	3.5	2.5
User 8	Proposed	**125**	**105**	**4**	**4**
	Traditional	190	350	4	3.5
User 9	Proposed	**170**	**220**	**4**	**5**
	Traditional	580	610	3	2.5
User 10	Proposed	**150**	**200**	**4.5**	**5**
	Traditional	450	570	2.5	2.5
Average	Proposed	**121.2**	**158**	**4.4**	**4.4**
	Traditional	243.9	323.8	3.35	2.95

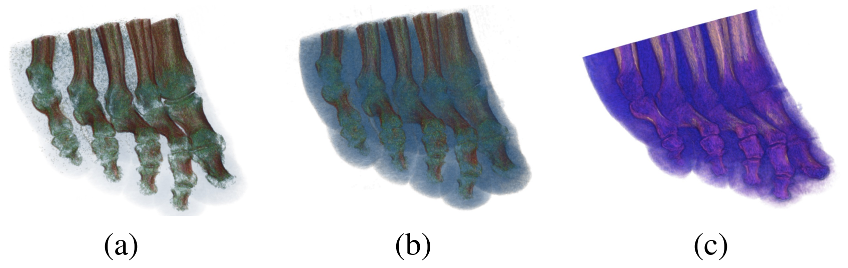


Figure 3.14: Rendering of the Foot dataset from different TF editors a) Proposed (2 Groups). b) Proposed (3 Groups). c) Traditional.

proposed system, the group is easily identifiable (Figure 3.14-(a)). Our proposed system is also flexible in terms of generating different visualizations, as voxel group visibility can be turned on or off with just one click. No such operation is possible with the traditional TF system.

3.7 Immersive Visualization in the CAVE

We have also extended the proposed method to be usable in the CAVE automatic virtual environment [65]. Traditional 2D displays lack the realistic sense of depth and size. Lack of judgment in depth or size can result in erroneous interpretation by the end user [66]. Advanced 3D displays like the CAVE can provide a sense of immersion for a better analysis of the data.

The CAVE used in this system consists of 4-screens; left, right, front and floor. Each screen is operated by a driver, where each driver consists of a computer connected to a projector. Since our target is complete immersion where the user resides in the CAVE during the whole visualization experience, we will have to make the SSOM visualization part portable. Hence, we project the SSOM lattice on an iPad. Figure 3.16 shows the UI on the iPad screen. The server communicates with the iPad and listens for user selection of voxel groups. The server also sends the corresponding RGBA texture information to the drivers in real time. The drivers render the data correspondingly. The stitching and accurate placement of different screens together and the head tracking for the 3D glasses to provide an immersive experience is maintained by VR Juggler [67]. The rendering is done through OpenSceneGraph [68].

Figure 3.17 shows the immersive volume visualization process in action. As we can see, the user controls the voxel grouping through the iPad while experiencing the result of the visualization immersively inside the CAVE. This provides the user with a better sense of depth perception which, in turns, can result in better interpretation of the results.

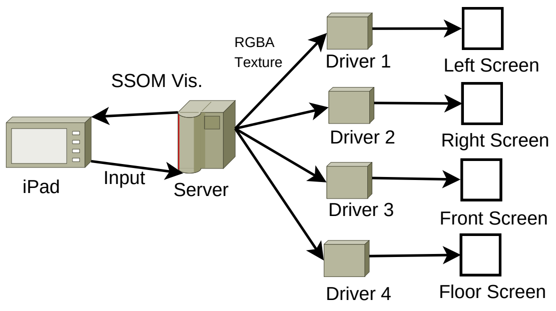


Figure 3.15: Overall system architecture for Immersive Volume Visualization

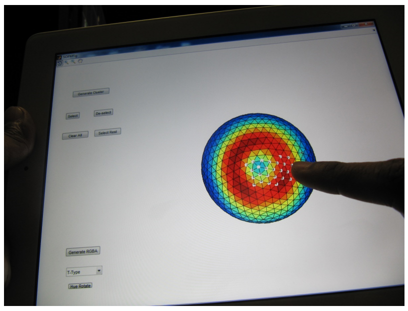


Figure 3.16: The SSOM lattice on the iPad

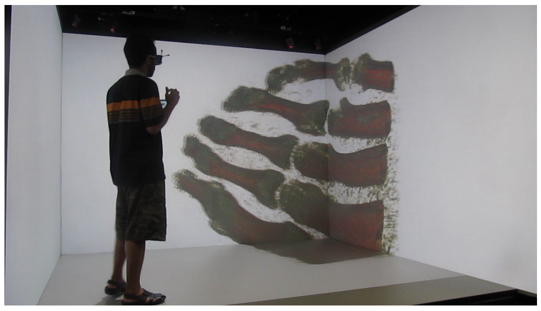


Figure 3.17: Rendering of the Foot dataset on the CAVE

3.8 Summary

In this chapter, we have proposed a new intuitive way of data-centric direct volume rendering with the help of Spherical Self-Organizing Maps and color harmonization. The user interacts with the SSOM lattice to find interesting regions and group SSOM nodes together. The TF is generated using color harmonization and the user's perception, which results in an aesthetically pleasing rendering where the detailed structures inside a volume are easily distinguishable. Real-time rendering of the generated TF on the volume dataset provides instant feedback to the user. The proposed system is intuitive to interact with and robust in nature, since any feature can be used with the SSOM lattice. Experimental results on five volume datasets verify the feasibility of our proposed approach. We have also extended the proposed method to be usable in the CAVE automatic virtual environment, which can provide the user with a better sense of depth perception and immersion.

Chapter 4

Novel Supervised Classifiers

4.1 Introduction

In the next few chapters we will lay down the basics to devise an image-centric approach towards volume rendering. In the image-centric approach, the user interacts with the image/slices of the volume data directly. Based on the user interaction, the volume rendering is generated. The image-centric approach gives the user direct control over which part of the volume is to be highlighted. Both data-centric and image-centric methods have their own pros and cons. A comparative discussion among these two approaches is presented later in chapter 5.

We planned a straight-forward image centric approach: the user selects some voxels from the volume slices; the system is trained based on the user selection and feature extraction. This training data is used to classify the rest of the voxels. Our automated color assignment scheme as discussed in the previous Chapter is then used to render the volume data.

It is apparent from the above simplified description that our image-centric approach will make use of supervised classification. Supervised classification is the task of finding a function which relates inputs and targets. The objective is to learn a model of dependency of the targets on the inputs. The ultimate goal is to be able to make accurate predictions of unseen input values. The non-linear classifiers have two stages [69]: (i) a fixed non-linear mapping transforms the data into a feature space and then (ii) a linear classifier is used to classify them in the feature space. The mapping is achieved through kernel functions [69]. There are two popular group of kernel-oriented methods. One is the class of classifiers based on the Kernel Fisher Discriminant Analysis (KFD) [70], the other is based on the Kernel Support Vector Machine

(KSVM) [71].

These two different groups of classifiers work on opposing philosophies [72]. The group of classifiers stemmed from the KFD are discriminant-based. The discriminant function is calculated based on the philosophy that minimizing the within-class distance while maximizing the between class distance will provide the best classification performance. The discriminant function is calculated through the use of between-class and within-class scatter matrices. Since the calculation of these matrices involve all the training data points, we can say that this group of classifiers make use of the "global" information available.

On the other hand, KSVM is based on the idea of maximizing the margin or degree of separation in the training data. There are many hyperplanes which can divide the data between two classes for classification. One reasonable choice for the optimal hyperplane is the one which represents the largest separation or margin between the two classes. KSVM tries to find the optimal hyperplane using support vectors. The support vectors are the training samples that approximate the optimal separating hyperplane and are the most difficult patterns to classify [71]. In other words, they are consisted of those data points which are closest to the optimal hyperplane. As KSVM deals with a subset of data points (support vectors) which are close to the decision boundary, it can be said that the KSVM solution is based on the "local" variations of the training data.

Since there is no prior way to know whether the global or local information will be most useful for classification, we attempt to combine these two classes of classifier. The basic idea is to incorporate the scatter matrices of KFD-bsed classifiers into the optimization problem of KSVM. Then, the scatter matrices can "guide" the direction of the separating hyperplane of KSVM into an optimum direction.

Based on some variations of this basic idea, we have proposed two novel classifiers. The first one is the Kernel Nonparametric Support Vector Machine Classifier (KN-SVM), which combines the Kenrel Nonparametric Discriminant (KND) and the KSVM classifier. The same idea is used in the field of novelty detection [19] to devise our second classifier named Covariance-Guided One-Class Support Vector Machine (COSVM). These two classifiers are described in detail in the next two sections with experimental results comparing against othet state-of-the-art classifiers. Finally, in the next chapter, our proposed KN-SVM classifier is used to devise a novel image-centric volume rendering approach.

4.2 Kernel Nonparametric Support Vector Machine (KN-SVM)

4.2.1 Motivation

The Nonparametric Discriminant Analysis in kernel space [73, 74, 18, 70, 75] (KND) is a more robust extension to the Kernel Fisher Discriminant Analysis (KFD) [70]. The advantage of KND over KFD is the relaxation of normality assumption [18]. KND measures the between-class scatter matrix on a local basis in the neighborhood of the decision boundary in the higher dimensional feature space. This is based on the observation that the normal vectors on the decision boundary are the most informative for discrimination [76]. In case of a two-class classification problem, these normal vectors are approximated by the $\kappa - NN$'s from the other class for one point. Although KND gets rid of the underlying assumptions of KFD and results in better classification performance, no additional importance is given to the boundary samples. In other words, the margin criterion (as calculated in KSVM) is not considered here. Moreover, it is not always an easy task to find a common and appropriate choice of $\kappa - NN$'s on the decision boundary for all data points to obtain the best linear discrimination [77].

On the other hand, KSVM only considers the support vectors close to the decision hyperplane to build the decision boundary [69]. It has been shown in the literature that maximum margin based classifiers like the KSVM typically perform better than discriminant (or average margin) based methods like the KND [78] due to their robustness and local margin consideration. However, KSVM can perform poorly when the data varies in such a way that data points exist far from the classification boundary [79]. This can be the case especially when the data is of high dimension. This is because KSVM does not take into consideration the global properties of the class distribution (as in the case of KND). This limitation of KSVM can be avoided by incorporating variational information from the KND which will control the direction of the separating hyperplane of KSVM. In that way we will have a maximum margin based classifier which is not sensitive to skewed data distribution like KSVM.

Several methods exist in literature which have addressed these issues inherent in discriminant based and maximum margin based methods. The ellipsoidal kernel machine was proposed in [80], where a geometric modification is proposed for data normalization by considering hyperellipsoids instead of hyperspheres in the classical KSVM method. Similarly, in [81], radius/margin bound has been used to iteratively optimize the parameters of KSVM efficiently. In [82], a kernel-based method has been proposed which essentially calculates the KFD scatter matrices based on the support vectors provided by KSVM. While these methods were backed by experimental improvements, most of them are a combination of multiple locally optimal algorithms to separately solve the discriminant based problem and margin maximization rather than providing one algorithm with one unique globally optimum solution.

Although the method proposed in [79] is superior to the previously described methods in the sense that it is based on only one convex optimization problem, it does so by introducing new constraints to the optimization problem. New constraints means new Lagrangian variables [71], which in turns can degrade the computational time. The Gaussian Margin Machine proposed in [83] tries to find the least informative distribution that classifies training data correctly by maintaining a Gaussian distribution of weight vectors. The drawback with this method is the expensive objective function involving log determinants in the optimization problem.

We propose a novel KN-SVM model which combines the KND and KSVM methods. In that way, a decision boundary is obtained which reflects both global characteristics (realized by KND) of the training data in feature space and its local properties (realized by the local margin concept of the KSVM). Being a kernel-based model, KN-SVM can deal with nonlinearly separable data efficiently. Rather than introducing new constraints like [79], our method modifies the objective function of KSVM by incorporating the scatter matrices provided by KND. Our proposed model forms a convex optimization problem because the final matrix used to modify the objective function is positive-definite. As a result, the method generates one global optimum solution. Because of this global extremum, existing numerical methods can be used to solve this problem easily and efficiently.

We also show that our method is a variation of the KSVM optimization problem, so that even existing SVM implementations can be used. The experimental results on real and artificial datasets show the superiority of our method.

4.2.2 KSVM and KND

Let $\mathscr{X}_1 = \{x_i\}_{i=1}^{N_1}$ and $\mathscr{X}_2 = \{x_i\}_{i=N_1+1}^{N_1+N_2}$ are two different classes of data samples constituting an input space of $N = N_1 + N_2$ samples and the associated tags are represented by $\mathscr{T} = \{t_i\}_{i=1}^{N}$, where $t_i \in \{0,1\},\ \forall i = 1,2,\ldots,N$. Since real-life data has inherent non-linearity, KSVM tries to map the data samples to a higher dimensional feature space $\mathscr{F}$, where linear classification might be achieved. Let, the two classes are mapped to higher dimensional feature classes $\mathscr{F}_1 = \{\Phi(x_i)\}_{i=1}^{N_1}$ and $\mathscr{F}_2 = \{\Phi(x_i)\}_{i=N_1+1}^{N}$ by the function Φ. Our target is to learn the weight vector

$\mathbf{w} = (w_0, w_1, \ldots, w_N)$ for functions of the form: $y(x;w) = \sum_{i=1}^{N} f_i^x w_i + w_0 = \Phi^T(x_i)\mathbf{w} + w_0$, where $\Phi(x) = (f_1^x, f_2^x, \ldots, f_N^x) : \mathscr{X} \to \mathscr{F}$ describes the non-linear mapping from the input space to the feature space for the input variable x [69]. The *kernel trick* [71] is used in case when the dimension of $\mathscr{F}$ is very high, where a kernel function $\mathscr{K}$ calculates the inner products of the higher dimensional data samples: $\mathscr{K}(x_i, x_j) =< \Phi(x_i), \Phi(x_j) >, \forall i, j \in \{1, 2, \ldots, N\}$.

In the feature space, KSVM tries to find the optimal decision hyperplane. The optimal hyperplane is the one with the largest margin, or, in other words, the plane which has largest minimal distance from any of the samples. Maximizing the distance of samples to the optimal decision hyperplane is equivalent to minimizing the norm of $\mathbf{w}$. As a result, this becomes part of the objective function. However, it might be the case that the problem is non-linear even in the higher dimensional space. To solve this, the margin constraint is relaxed or *slacked*. Also, a penalty factor is introduced in the objective function to control the amount of slack. This penalty factor is of the form of a loss function, usually a hinge loss function [71]. Incorporating all these, the KSVM optimization problem can be written as:

$$\begin{aligned} \min_{\mathbf{w} \neq 0, w_0} & \left\{ \frac{1}{2}\mathbf{w}^T\mathbf{w} \right. \\ & \left. + C\sum_{i=1}^{N} max(0, 1 - t_i(\Phi^T(x_i)\mathbf{w} + w_0)) \right\}, \end{aligned} \tag{4.1}$$

Here, $max(0, 1 - t_i(\Phi^T(x_i)\mathbf{w} + w_0))$ is the hinge loss function [84]. For correctly classified training samples, this function does not incur any loss. For misclassification, the loss factor is controlled by C. Since the weight vector $\mathbf{w}$ resides in the feature space, it cannot be calculated directly. Instead, the Lagrangian dual problem is solved [85]. The optimal weight vector for this problem is a linear combination of the data points and is of the form $\mathbf{w}^* = \sum_{i=1}^{N} t_i \alpha_i^* \Phi(x_i)$, where $\{\alpha_i\}_{i=1}^{N}$ are the Lagrangian variables. The decision function for any test sample x is obtained by:

$$g(x) = \sum_{i=1}^{N} t_i \alpha_i^* \mathscr{K}(x, x_i) + w_0^*, \tag{4.2}$$

where w_0^* is computed using the primal-dual relationship [85], and where only samples with non-zero Lagrange multipliers α_i contribute to the solution. The corresponding data samples are called Support Vectors (SVs). These points are the crucial samples for classification.

Therefore, KSVM considers only those data points which are close to the decision hyperplane and are critical to find the decision boundary. In other words, KSVM only considers the local variations in data samples. The overall distributions of the training samples are not taken into consideration. Incorporating some kind of global distribution (e.g. results from classifiers like KND) can provide better classification.

Before describing the Kernel Nonparametric Discriminant (KND) method, we introduce the Linear Discriminant Analysis (LDA) first. LDA tries to optimize the projection direction by maximizing the between-class distance while minimizing the within-class distance. The best projection direction v^* can be described by the following optimization problem:

$$v^* = \arg \max_{v} \frac{v^T S_b v}{v^T S_w v}, \tag{4.3}$$

where the within-class scatter matrix is defined as:

$$S_w = \frac{1}{N_1 + N_2}(N_1 S_1 + N_2 S_2), \tag{4.4}$$

where S_1 and S_2 are the covariance matrices for the two classes. The between-class scatter matrix is defined as:

$$S_b = (m_1 - m_2)(m_1 - m_2)^T, \tag{4.5}$$

where m_1 and m_2 are the empirical means of the two classes. Problem (4.3) can be solved by finding the eigenvalues and eigenvectors of $S_w^{-1} S_b$ [86]. v^* is formed by the eigenvector corresponding to the largest eigenvalue.

Despite being a popular method, the performance of LDA degrades when the underlying class distribution is not normal. In [18], the Nonparametric Discriminant Analysis (NDA) is proposed to remove the normality assumption in the LDA. The principle behind the method is to calculate the between-class scatter matrix of the LDA on a local basis. The NDA can be extended to the feature space $\mathscr{F}$ based on the ideas developed in extending the LDA to the Kernel Fisher Discriminant (KFD) [70]. We call this the Kernel Nonparametric Discriminant (KND).

Instead of calculating the simple mean vectors, the nearest neighbor mean vectors are calculated to formulate the between-class scatter matrix of the NDA. In our feature space, this vector can be defined as:

$$M_m^{\kappa}(\Phi(x_i)) = \frac{1}{\kappa}\sum_{j=1}^{\kappa}\Phi(x_i)_{NN}(j), \tag{4.6}$$

where, $\Phi(x_i)_{NN}(j)$ defines the j^{th} nearest neighbor from data point x_i of class m. κ is the free parameter which defines how many neighbors to consider. This parameter needs to be optimized for each dataset. Now, let us define two matrices $L_1(\Phi(x_i))$ and $L_2(\Phi(x_i))$. We will use the kernel trick to formulate these matrices. In that case, the matrices are calculated on a component by component basis, where, a component of $L_1(\Phi(x_i))$ is defined as:

$$\begin{aligned}&(L_1(\Phi(x_i)))_j = \mathscr{K}(x_j,x_i) - (M_2^{\kappa}(\Phi(x_i)))_j,\\ &\forall i \in \{1,2,\ldots,N_1\}, \forall j \in \{1,2,\ldots,N\},\end{aligned} \tag{4.7}$$

and a component of $L_2(\Phi(x_i))$ is defined as

$$\begin{aligned}&(L_2(\Phi(x_i)))_j = \mathscr{K}(x_j,x_i) - (M_1^{\kappa}(\Phi(x_i)))_j,\\ &\forall i \in \{N_1+1,N_1+2,\ldots,N_1+N_2\}, \forall j \in \{1,2,\ldots,N\}.\end{aligned} \tag{4.8}$$

With these formulations, the between-class scatter matrix in the feature space can be defined as:

$$\begin{aligned}\nabla = &\frac{1}{(N_1+N_2)}\sum_{i=1}^{N_1}\Psi_i L_1(\Phi(x_i))L_1(\Phi(x_i))^T\\ &+\frac{1}{(N_1+N_2)}\sum_{i=N_1+1}^{N_1+N_2}\Psi_i L_2(\Phi(x_i))L_2(\Phi(x_i))^T.\end{aligned} \tag{4.9}$$

Here, Ψ_i are the weighting functions to nullify the effects of samples that are far from the boundary. It is defined as follows [18]:

$$\Psi_i = \frac{min\{d(\Phi(x_i),\Phi(xNN_{1i}^{\kappa}))^{\gamma}, d(\Phi(x_i),\Phi(xNN_{2i}^{\kappa}))^{\gamma}\}}{d(\Phi(x_i),\Phi(xNN_{1i}^{\kappa}))^{\gamma} + d(\Phi(x_i),\Phi(xNN_{2i}^{\kappa}))^{\gamma}}, \tag{4.10}$$

where γ is a control parameter which can range from zero to infinity, and $d(\Phi(x_i), \Phi(xNN_{ji}^{\kappa}))$ is the Euclidean distance from x_i to its $\kappa - NN$'s from class $\mathscr{X}_j$ in the kernel space. γ controls how rapidly the value of weighting function falls to zero as we move away from the classification boundary.

The motivation behind KND is the observation that essentially the nearest neighbors represent the classification structure in the best way. For small values of κ, the matrices in Equation 4.7 and 4.8 represent the direction of the gradients of the respective class density functions in the feature space [87]. If the weighting functions are not used, samples with large gradients that are far from the boundary may pollute the necessary information. Hence, these gradients with combination of the weighting functions form the between-class scatter matrix ∇, which preserves the classification structure.

The KND does not make any modifications to the within-class scatter matrix. As a result, the formula for within-class scatter matrix Δ is similar to the KFD, and can be written as follows:

$$\Delta = \mathbf{K}_1(I - 1_{N_1})\mathbf{K}_1^T + \mathbf{K}_2(I - 1_{N_2})\mathbf{K}_2^T, \tag{4.11}$$

where $\mathbf{K}_1$ is a $N \times N_1$ Kernel matrix for the class $\mathscr{X}_1$ and $\mathbf{K}_2$ is a $N \times N_2$ Kernel matrix for the class $\mathscr{X}_2$, with

$$\begin{aligned} &(\mathbf{K}_1)_{nm} = \mathscr{K}(x_n, x_m), \\ &(n,m) \in \{1,2,...,N\} \times \{1,2,...,N1\}, \end{aligned}$$

and

$$\begin{aligned} &(\mathbf{K}_2)_{nm} = \mathscr{K}(x_n, x_m), \\ &(n,m) \in \{1,2,...,N\} \times \{N1+1,...,N\}, \end{aligned}$$

respectively. I is the identity matrix and 1_{N_1} and 1_{N_2} are the matrices with all entries $\frac{1}{N_1}$ and $\frac{1}{N_2}$, respectively.

With these definitions of ∇ and Δ, the KND method proceeds the same way as the LDA i.e., by computing the eigenvectors and eigenvalues of $\Delta^{-1}\nabla$. Since the higher dimensional feature space $\mathscr{F}$ is of dimension N, the matrix Δ is needed to be regularized before calculating the inverse. This is achieved by adding a small multiple β of the identity matrix I [70]. Hence, the eigenvectors and eigenvalues of $(\Delta+\beta I)^{-1}\nabla$ are computed, and the eigenvector corresponding to the largest eigenvalue forms the optimal decision hyperplane. If we assume that the eigenvector corresponding to the largest eigenvalue is represented by Υ, then, a new sample point x can be projected onto the decision hyperplane by computing $\sum_{i=1}^{N} \Upsilon_i \mathscr{K}(x_i,x)$.

The most significant advantage of the KND over the KFD is the removal of the normality assumption. Because of this, KND is a robust classifier which can perform better in case of real-life datasets. However, finding the optimum number of nearest neighbors is not an easy task. Also, KND considers the global variations in the data distribution by considering the κ-nearest neighbors. Unlike the KSVM, the points that are crucial to classify (the local variations) are not given any special consideration. This can result in degradation of performance.

4.2.3 The KN-SVM method

It is clear from previous sections that both the KSVM and the KND approach the classification problem from different perspectives. As indicated before, the KSVM relies on local data variations while the KND uses global data variations. This set of information is crucial for classification, specially in case of real world datasets, as the data distribution is not known in advance. In our proposed KN-SVM method, we try to utilize both the local and the global variational information obtained from these two classifiers. Our objective is to keep the method close to the KSVM approach as much as possible, since the KSVM optimization approach is more robust than the discriminant approach of KND in the sense that it is less sensitive to data distribution or the small sample size problem. We modify the KSVM optimization problem (Equation 4.1) to incorporate the scatter matrices from the KND (Equation 4.9 and 4.11). Here, for simplicity, we treat the kernel function as definition of a set of basis functions, rather than as a definition of a dot product in some space [88]. Hence, the weight vector $\mathbf{w}$ can be defined as $\mathbf{w} = (w_0, w_1, \ldots, w_N)$. Similarly, the high-dimensional mapping of a data point x_i can be defined as $\Phi(x_i) = (\mathscr{K}(x_1,x_i), \mathscr{K}(x_2,x_i), \ldots, \mathscr{K}(x_N,x_i))$.

KND computes $(\Delta+\beta I)^{-1}\nabla$ and takes the eigenvector corresponding to the *largest* eigenvalue to form the optimal decision hyperplane. However, KSVM optimization problem (Equation 4.1) is a *minimization* problem. Therefore, we plug in $\Delta(\nabla+\beta I)^{-1}\Delta$ into Equation 4.1. Note that here we are inverting ∇ as opposed to Δ in KND, since it is a minimization problem.

Hence, our method can be described by the following convex optimization problem:

$$\min_{\mathbf{w}\neq 0, w_0} \left\{ \frac{1}{2}\mathbf{w}^T(\eta\Delta(\nabla+\beta I)^{-1}\Delta)\mathbf{w} + \frac{1}{2}\mathbf{w}^T\mathbf{w} \right. \\ \left. +C\sum_{i=1}^{N} max(0, 1-t_i(\Phi^T(x_i)\mathbf{w}+w_0)) \right\}. \tag{4.12}$$

If we compare this problem with Equation 4.1, we see that the first term is added to the optimization problem to incorporate the KND scatter matrices. The coefficient $\eta\Delta(\nabla+\beta I)^{-1}\Delta$ changes the orientation of the weight vector **w** by incorporating global variational information obtained by the KND. The parameter η is the key control parameter to find out the optimal weight vector orientation. η can take values from 0 to ∞. The second term $\frac{1}{2}\mathbf{w}^T\mathbf{w}$ is the traditional KSVM term to retain the local variational information. For a more compact form, we can combine the first and the second term. As a result, the optimization problem for the KN-SVM has the following form:

$$\min_{\mathbf{w}\neq 0, w_0} \left\{ \frac{1}{2}\mathbf{w}^T(\eta\Delta(\nabla+\beta I)^{-1}\Delta + I)\mathbf{w} \right. \\ \left. +C\sum_{i=1}^{N} max(0, 1-t_i(\Phi^T(x_i)\mathbf{w}+w_0)) \right\}. \tag{4.13}$$

Problem (4.13) is a convex quadratic optimization problem. This is because the matrices Δ and ∇ are positive-definite by definition (being the within-class and between-class scatter matrices). Moreover, the inverse of a positive-definite matrix is positive-definite [89]. So $(\nabla+\beta I)^{-1}$ is positive-definite. We also know that if two matrices A and B are positive-definite, then ABA is positive-definite [89]. Hence, the term $(\eta\Delta(\nabla+\beta I)^{-1}\Delta+I)$ in Equation 4.13 is positive-definite. This clearly makes KN-SVM a convex optimization problem with only one global optimum solution. This is a significant advantage, since numerical methods can be used to solve our problem easily and efficiently while retaining the advantages of both KND and KSVM.

Figure 4.1 shows a (hypothetical) geometric interpretation of how our proposed method may achieve better classification results. For simplicity, we assume that the depiction is in the higher-dimensional feature space, where the data points are linearly separable. As we can see, the KSVM decision hyperplane is determined by the two hyperplanes $P1$ and $P2$.

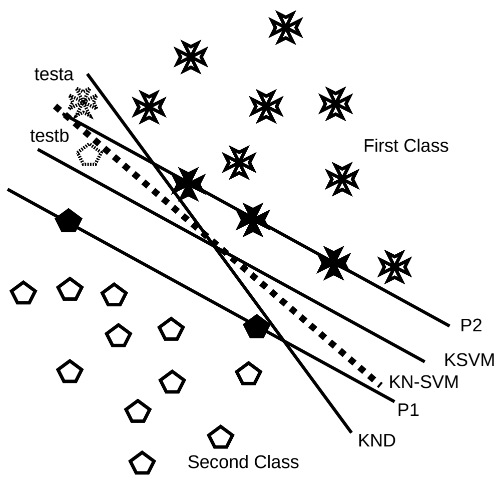


Figure 4.1: Hypothetical Geometric Depiction of KN-SVM

$P1$ and $P2$ are the planes that contain the support vectors. Hence, the KSVM hyperplane is constructed considering only local variations in the data, i.e., the support vectors. The KND decision plane, on the other hand, considers the global variations in data when building the between-class scatter matrix. As a result, this plane differs from the one achieved through the KSVM. The hyperplane of our proposed KN-SVM method (depicted by a dotted line) is a balance between these two. By incorporating the scatter matrices of the KND into the KSVM optimization problem, we change the direction of the decision hyperplane. The amount of change of direction is controlled by the parameter η (Equation 4.13). Theoretically, by properly tuning this parameter through cross-validation, better classification results can be achieved. As we see in our hypothetical depiction, two test points, $testa$ and $testb$ are presented to the three different classification methods. KSVM can only classify $testa$ correctly, while KND can classify only $testb$. By changing the direction of the decision hyperplane properly, our method can correctly classify both of these points. Although this is a hypothetical scenario but later we will observe from the experimental results that by reasonable parameter tuning, our method can provide better results.

4.2.4 Solving the Optimization Problem

Since our optimization problem is similar to the KSVM optimization problem, we can solve it in a similar way, i.e., by using Lagrange multipliers [71]. If the positive Lagrange multipliers are α_i, for $i = 1, \ldots N$, then, the dual of our KN-SVM problem in terms of Lagrange multipliers is:

$$
\begin{aligned}
& max\left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} t_i t_j \alpha_i \alpha_j \Phi^T(x_i) \Theta^{-1} \Phi(x_j) \right\}, \\
& s.t. \ \sum_{i=1}^{N} t_i \alpha_i = 0,\ 0 \le \alpha_i \le C, \ \ \forall i = 1, \dots N,
\end{aligned}
\tag{4.14}
$$

Here, $\Theta = \eta\Delta(\nabla + \beta I)^{-1}\Delta + I$. Let, $\{\alpha_i^*\}_{i=1}^N$ represents the solution to this optimization problem. Then the optimal weight vector $\mathbf{w}^*$ can be found by using the following equation:

$$
\mathbf{w}^* = \sum_{i=1}^{N} t_i \alpha_i^* \Theta^{-1} \Phi(x_i). \tag{4.15}
$$

Also, the offset w_0^* can be found using the Karush-Kuhn-Tucker conditions [90]. The final decision boundary of the KN-SVM for a test point x is:

$$
y(x; w) = \sum_{i=1}^{N} t_i \alpha_i^* \Phi^T(x_i) \Theta^{-1} x + w_0^* = 0. \tag{4.16}
$$

However, obtaining the KN-SVM solution this way requires an entirely new implementation to test this method. The following lemma gives us an easier alternative to implement this method:

Lemma 4.2.1 *The KN-SVM method formulation is equivalent to:*

$$
\min_{\hat{\mathbf{w}} \neq 0, w_0} \left\{ \frac{1}{2} \hat{\mathbf{w}}^T \hat{\mathbf{w}} + C \sum_{i=1}^{N} max(0, 1 - t_i(\hat{\Phi}^T(x_i) \hat{\mathbf{w}} + w_0)) \right\}, \tag{4.17}
$$

where

$$
\hat{\mathbf{w}} = \Theta^{1/2} \mathbf{w}, \tag{4.18}
$$

$$
\hat{\Phi}(x_i) = \Theta^{-1/2} \Phi(x_i) \ \ \forall i = 1, \dots, N \tag{4.19}
$$

and

$$\Theta = \eta\Delta(\nabla + \beta I)^{-1}\Delta + I. \tag{4.20}$$

Proof Substituting Equations 4.18-4.20 into equation 4.17 we get the original KN-SVM problem (Equation 4.13).

This lemma gives us a significant advantage from the implementation viewpoint. This essentially means that we can use the existing SVM implementations [91] provided we can calculate the terms $\Theta^{1/2}$ and $\Theta^{-1/2}$. This can easily be done through the help of eigenvalue decomposition of the matrix Θ [92]. Θ is a square matrix of size $N \times N$ by definition. Let $v_1, v_2, \ldots, v_N$ be the eigenvalues of Θ and $s_1, s_2 \ldots s_N$ be the corresponding set of eigenvectors. Let V denotes the $N \times N$ diagonal matrix with all the eigenvalues as diagonal i.e., $V_{ii} = v_i$. Let S denotes the $N \times N$ matrix where the ith column is s_i. Then we have:

$$\Theta S = SV. \tag{4.21}$$

Alternatively:

$$\Theta = SVS^{-1}. \tag{4.22}$$

Now, we can calculate $\Theta^{1/2}$ and $\Theta^{-1/2}$ as follows:

$$\Theta^{1/2} = SV^{1/2}S^{-1}, \tag{4.23}$$

$$\Theta^{-1/2} = SV^{-1/2}S^{-1}. \tag{4.24}$$

As V is a diagonal matrix, calculating $V^{1/2}$ or $V^{-1/2}$ reduces to just calculating $v_i^{1/2}$ or $v_i^{-1/2}$ for each eigenvalue v_i, which is trivial.

After calculating $\Theta^{1/2}$ and $\Theta^{-1/2}$, we can compute the terms in Equations 4.18-4.20. After that, any existing KSVM implementation can be used to obtain the results from our method. In our experiments, we have used the KSVM implementation provided with MATLAB™[91]. The algorithm used to solve the optimization problem in this implementation is based on the interior-reflective Newton method described in [93].

Table 4.1: List of parameters to be optimized for each method.

Parameter	KN-SVM	KSVM	KND	KFD	Gaussian
Weight Parameter (η)	✓	-	-	-	-
Nearest Neighbors (κ)	✓	-	✓	-	-
RBF Width Parameter (σ)	✓	✓	✓	✓	-
KSVM Regularization (C)	✓	✓	-	-	-

4.2.5 Experimental Results

In this section we evaluate the proposed KN-SVM method against four other contemporary classifiers, namely, the KSVM, KND, the Kernel Fisher Discriminant (KFD) [70] and the Gaussian method[18].

For kernelization of the data, we use the Gaussian RBF Kernel $\mathcal{K}(x_i,x_j) = e^{-\|x_i-x_j\|^2/\sigma}$. This kernel is proven to be robust and flexible [69]. Here, σ represents the positive "width" parameter. For KND and KFD, after finding the optimal eigenvector, Bayes classifier was used for conducting the final classification.

There are several control parameters for each of the methods we implemented. A summary of the parameters for each method can be found in Table 4.1. The parameter γ for KND (Equation 4.10) does not effect the results significantly and, hence, it was set to 1 throughout the experiments.

The Gaussian method fits the two classes to Gaussian distributions during training, and Naive Bayes classification [86] is used during testing. For this process, no parameter tuning is necessary, as the parameters are estimated during training. This classifier is included to tease out the advantages of using kernel mapping. Also, since the Naive Bayes classification assumes all the features to be uncorrelated [94], the advantage of considering the correlation among features while computing the weights (e.g. KSVM or KN-SVM) will also be apparent by comparing with this method.

The involved parameters were optimized using exhaustive search to try all possible combinations. Although the parameter optimization is a lengthy process, this needs to be done only once for a new dataset, and, hence, does not contribute to the actual classification performance. If the optimization needs to be faster, efficient methods like coordinate descent technique [95] can be used at the cost of a small degradation in accuracy values.

The number of parameters to tune for the KN-SVM method is 4, while it is 2 for KSVM and KND. It might seem that an accurate fit of the parameter values is necessary for KN-SVM to perform well, specially if we have a small training dataset. But as we will see from the results, KN-SVM performs better compared to other methods by tuning over a limited range of parameter values we have used (e.g. we use a set of only 20 κ values and 20 η values for parameter tuning to obtain the results of Table 4.2). Since this is a combination of KSVM and KND, the parameters compensate each other, and the fit doesn't necessarily have to be perfect. Also, for small training set, we tackle the problem of poor performance due to inaccuracies in matrix inversion by adding a regularization term before inverting (Equation 4.12).

We have applied the classification algorithms on 11 real-world and artificial datasets.The datasets are obtained from the Benchmark Repository used in [96]. Namely, the datasets are: Flare-Sonar, Breast-Cancer, German, Heart, Banana, Diabetes, Ringnorm, Thyroid, Twonorm, Waveform and Splice. These datasets are obtained from the UCI, DELVE and STATLOG repositories [97]. Some of these datasets are originally multi-class. In such cases, some of the classes were (randomly) merged to convert it into a two-class classification problem. 100 partitions are then generated for each dataset, where about 60% data is used for training and the rest for testing [96]. For our experimental results, we randomly picked 5 out of these 100 partitions (5 partitions each for training and 5 each for testing). Additionally, we repeated this random picking process 5 times to achieve the average result. This randomness was introduced to ensure that no method has a coincidental advantage over the others. For parameter tuning, 5-fold cross validation on the training dataset was performed for each model (i.e. 4 out of the 5 picked training partitions were used for training and the remaining one for validation at each stage of cross validation).

Table 4.2 contains the average accuracy values and the standard deviations obtained over all the runs. We see that the KN-SVM method outperforms the KSVM, KND, KFD and the Gaussian methods in all of the cases except for Breast-cancer, where it is the second best. Since the KN-SVM combines the local and global variations provided by the KSVM and the KND, respectively, it can classify the relatively difficult test samples. Also, being a variation of the KSVM and KND, this method is free from any underlying distribution assumption, and, hence, can provide better results. In case of Breast-cancer, our method improves on the result provided by the KSVM, but falls behind the KND. Even in this case, better result might be achieved by testing with a different set of weight parameter (η) values. However, since theoretically the parameter can take any value from 0 to ∞, the possibilities are endless. To reduce the time of optimization, we had to restrict ourselves to only a few values. Still, as we can see, these limited

Table 4.2: Average percentage classification accuracy and standard deviation (in parentheses) of each method for the 11 data sets (best method in **bold**, second best *emphasized*). The last two rows contain the average cpu time for each method (in *seconds*) and the t-test confidence interval, respectively.

Dataset	KN-SVM	KSVM	KND	KFD	Gaussian
Flare-Sonar	**67.7** (0.47)	66.9 (0.41)	*67.1* (0.65)	66 (0.40)	51.4 (0.2)
Breast-Cancer	*78.9* (1.96)	77.4 (2.1)	**79.3** (2.00)	77 (2.37)	72.4 (0.16)
German	**78.1** (0.40)	*77* (0.38)	76.3 (0.68)	75.7 (0.51)	73.4 (0.03)
Heart	**86.5** (2.21)	*85.4* (2.3)	81.7 (1.58)	82.9 (2.13)	84.3 (0.2)
Banana	**89.8** (0.25)	*89.6* (0.29)	*89.6* (0.22)	89.5 (0.20)	60.1 (0.17)
Diabetes	**78.6** (0.50)	*77.7* (0.69)	75.7 (0.90)	77.3 (1.03)	75.2 (0.04)
Ringnorm	**98.5** (0.04)	*98.4* (0.04)	98.3 (0.03)	97.4 (0.07)	**98.5** (0.05)
Thyroid	**97.3** (0.6)	96.5 (1.02)	*97.1* (0.64)	96.8 (0.49)	92.7 (0.09)
Twonorm	**97.7** (0.04)	*97.6* (0.05)	96.5 (0.32)	96.9 (0.08)	*97.6* (0.04)
Waveform	**90.7** (0.15)	*90.5* (0.15)	89.3 (0.19)	90 (0.12)	85.2 (0.02)
Splice	**88.9** (0.41)	*88.7* (0.38)	88.5 (0.41)	88.4 (0.36)	87.2 (0.02)
Avg. time	4.04	4.05	2.95	2.92	0.09
Confidence	-	99.8	97.6	99.9	97.2

values are good enough for almost all the datasets. This establishes the fact that our method can be used in practical applications. To measure the statistical significance of the results, we paired up the KN-SVM method with the other methods and performed paired t-tests on the accuracy values [98]. The paired t-test determines whether or not two paired sets of measured values are significantly different. The last row of Table 4.2 provides the confidence intervals (in %) obtained from the performed t-tests. This confidence interval quantifies the probability of the paired distributions being the same. The higher the confidence interval, the lower is the probability that the underlying distributions are statistically indifferent. As we can see, all the confidence intervals are almost 100%, which proves that the KN-SVM method indeed provides statistically significant accuracy improvements.

If we compare the results between the KND and KFD, we see that in some cases, the KFD provides better classification results than the KND. This is due to the fact that the optimal nearest neighbor parameter for the KND (the $\kappa - NN$'s) is not always easy to find. But since our method combines the KND with KSVM, the optimality of this parameter is not as crucial as it is in the KND.

We also see that the Gaussian method mostly provides inferior results compared to the other methods. This is because in the Gaussian process, the classes are being fitted to Gaussian distributions. The Gaussian method also considers features to be uncorrelated [94]. These assumptions may not necessarily be true for most of these datasets. Also, there is no kernelization being performed in this case. As mapping to a higher dimension with kernel can make the classification problem easier to solve, the Gaussian process suffers from inferior performance.

The computational complexity of the KSVM scales with $\mathscr{O}(N^2)$ for one iteration [69]. The KND and KFD scales with a computational complexity of $\mathscr{O}(N^3)$ (dominated by the inversion of the within-class scatter matrix [99]). The Guassian method scales with the complexity of $\mathscr{O}(NM)$ [100], where M is the number of attributes (dimensionality) of a data point. Since there is no kernel mapping in the Gaussian method, it has the lowest computational complexity. KND, KFD and the Gaussian method requires only one run as there is no iterative process involved.

In the KN-SVM, the complexity for the inversion of Θ (i.e. $\eta\Delta(\nabla+\beta I)^{-1}\Delta+I$) scales with $\mathscr{O}(N^3)$. However, this inversion process can be considered to be part of pre-processing, as it is needed to be done only once before start of the training. Therefore, the computational complexity of our proposed KN-SVM can be considered similar to that of the KSVM, i.e., $\mathscr{O}(N^2)$ per iteration. This can also be seen from the obtained results (second last row of Table 4.2), where we see that the average computational time of our method is on par with that of KSVM.

4.2.6 Summary

We have proposed a novel classification method named KN-SVM. The KN-SVM method incorporates the local variational information from the KSVM and the global information from the KND. Being a combination of these two methods, KN-SVM is a robust classifier, free from any underlying assumption regarding class distribution. Our method is also capable of tackling the small sample size problem. Being a convex optimization problem, our method provides a global optimum solution and can be solved efficiently by using numerical methods. Besides, we have shown that our method can be reduced to the classical KSVM model so that existing KSVM implementations can be used. The experimental results on some benchmark datasets verifies the superiority of our method, where we compare KN-SVM with the KSVM, KND, KFD and the Gaussian method.

4.3 Covariance-guided One-Class Support Vector Machine (COSVM)

4.3.1 Motivation

In this section, we explore the application of the same philosophy of combining classifiers in the field of one-class classification. One-class classification, novelty detection, outlier detection or concept learning [101, 102] is the process of separating one particular type of data from the others. The difference with traditional two-class or multi-class classification is that only one class of data is available for training (deemed as *targets*). The objective is to distinguish any other data points from the targets. The "other" data points are typically called *outliers*. One-class classification is necessary for several reasons: 1) Outlier data may not be available or very costly to measure. For example, it is possible to measure the necessary features for a nuclear plant operating under normal circumstances. However, it is too dangerous or impossible to measure the same features in case of an accident. In this case, a classifier has to be trained based only on the data for normal circumstances (the target class). 2) In some cases, the available outlier data might be too small or badly sampled with unknown priors and ill-defined distributions [103]. 3) Another scenario where one-class classification can be of importance is the comparison of two datasets [103]. Usually, a classifier is trained on a dataset by a complex procedure. It will be beneficial if this training information can be reused. To solve a similar problem, the new dataset can be compared with the old dataset instead of repeating the whole training process.

For these reasons, one-class classification has found its application in several fields such as engine fault detection [104], medical diagnosis [105], nuclear testing [106], web page classification [107] and network intrusion detection [108].

The key limitation in one-class classification is that only one class of labeled dataset is available during training. The existing approaches use different techniques to estimate the necessary parameters to classify future data points as targets or outliers. Based on the techniques used, they can be divided into three categories; reconstruction based, density-based and boundary-based methods [109].

In reconstructive classifiers, a model regarding the data generation process is assumed first. In the training phase, the parameters of this model are estimated according to the target dataset. During classification, a reconstruction error for the incoming data point is calculated. The less the error, the more accurate is the model. Examples of reconstructive classifiers are K-Means [110], Self-Organizing Map (SOM)[17], etc.

Density-based one-class classifiers are based on the estimation of the probability density function (PDF) of the target class [111]. The PDF is estimated through the training data using statistical methods. For example, a Parzen density estimator is used in [112, 113]. Gaussian distribution is used in [114]. After density estimation, thresholding is used to label data points as belonging to the target class or the outliers.

The boundary-based classifiers are built upon the idea of the traditional two-class SVM [71]. Instead of using the complete training data to estimate the distribution (as done in density-based classifiers), only the boundary data points are used to guess the area where the future target data points could reside. The boundary points around the target class are used to classify an incoming data point. Example of popular boundary-based one-class classifiers are the Support Vector Data Description (SVDD) [103] and One-class Support Vector Machines (OSVM) [19].

The problems with these existing categories of one-class classification methods are that none of them consider the full scale of information available for classification. In density-based methods, solely the overall class probability distribution is used. The first problem with this approach is that a large number of samples are required for density estimation. Also, this kind of methods estimate the density based on the training data. But the latter represents the area of available targets only, not the complete distribution. The true distribution is unknown and can be unpredictable for a real-world problem. Density-based methods only focus on high density areas [103], and reject areas with lower training data density, although they represent valid targets. This limitation can reduce the performance of a classifier.

On the other hand, in boundary-based methods, only boundary data points are considered to build the model. These points do not completely represent the overall class. Solutions to boundary-based methods are only calculated based on the points near the decision boundary, regardless of the spread the remaining data. In [79], it has been shown that solutions to boundary-based methods like OSVM can be misled by the spread of data, since these methods tend to separate the data along large spread directions. A more reasonable method would be to simultaneously make use of the maximum margin criterion [69] while controlling the spread of data [79].

Also, unlike multi-class classification problems, the low variance directions of the target class distribution are crucial for one-class classification. In [115], it has been shown that projecting the data in the high variance directions (like PCA) will result in higher error (bias), while retaining the low variance directions will lower the total error. Boundary-based methods do not put any special emphasis on these low variance directions. In fact, these methods preferentially separate data along large variance directions [79]. However, finding the optimal number of directions to retain is also not possible because of the basic bias-variance dilemma [116]. This dilemma arises from the fact that the estimated covariance is not accurate due the limited number of training samples. Hence, we need to reduce the estimation error by taking projections along some variance directions. However, taking these projections before training will increase the total error (bias) since we are losing important characteristics from the training data. It is nearly impossible to find out the perfect projectional directions which will generate the lowest total error in case of all datasets.

The principal motivation behind our proposed method is to use the robustness of the boundary-based classifiers while emphasizing the small variance projectional directions. We want to use the maximum margin based solution while optimally controlling the projectional direction. However, we don't want to lose any data characteristics by directly taking projections in specific directions before training. Generally, the estimated covariance matrix represents necessary information regarding the required variational directions. Hence, our approach incorporates the covariance matrix into the well-known One-class Support Vector Machine (OSVM) method [19]. The OSVM method is essentially a one-class interpretation of the classic SVM problem. We call our proposed method the Covariance-guided One-Class Support Vector Machine (COSVM) method.

In our proposed COSVM method, we plug the covariance matrix into the optimization problem of OSVM. The covariance matrix is estimated in the kernel space [70]. The estimated covariance matrix has the required information for all projectional directions, both along high variance and low variance. However, the OSVM optimization problem is a *minimization* problem. Hence, we can intuitively assume that incorporating the covariance matrix into the optimization problem of OSVM as an additional term will result in emphasizing the low variance directions. The additional term will essentially act as a penalty factor. The high variance directions will be penalized more than the low variance directions. Hence, after training, the weight vector will be adjusted in such a way that the low variance directions are assigned more weight (importance) than the high variance directions. This technique does not increase the overall computational complexity of the OSVM method. COSVM still results in a convex optimization problem with one global optimum solution which can be found efficiently using existing numerical methods. The degree of emphasis on the covariance matrix can be elegantly controlled through one parameter only (details in Section 4.3.3). This provides a quick control over the bias-variance dilemma. The performance of our classifier can be tuned very easily by changing this single parameter. We also provide an analytical approach to tune the parameter.

Our covariance term is incorporated into the dual problem of OSVM. Exploiting the dual problem allows us to use the Kernel trick for efficient calculation (explained in detail in Section 4.3.3. Since COSVM keeps the basic formulation of the OSVM problem unchanged, it can be implemented through the existing OSVM packages with minimal coding. In fact, in our experiments, we have used the SVM-KM toolbox [117] for OSVM, and implemented the COSVM method with only a few additional external functions.

4.3.2 One-Class SVM

Let $\mathscr{X} = \{x_i\}_{i=1}^{N}$ represent the training dataset of N samples. Since real-world data has inherent non-linearity, SVM-based methods try to map the data samples to a higher dimensional feature space $\mathscr{F}$, where linear classification might be achieved. Let, the target class is mapped to a higher dimensional feature class $\mathscr{F} = \{\Phi(x_i)\}_{i=1}^{N}$ by the function Φ. In One-class SVM (OSVM)[19], the strategy is to learn a function y that returns $+1$ in a small region consisting of all the training data points, and returns -1 for any other point. Mathematically, the target is to learn the weight vector $\mathbf{w}' = (w'_1, \ldots, w'_N)$ and the offset ρ for function y of the following form:

$$y(x;\mathbf{w}') = \begin{cases} 1, & \mathbf{w}'^T\Phi(x) - \rho \geq 0. \\ -1, & \text{otherwise.} \end{cases} \tag{4.25}$$

where $\Phi(x) = (f_1^x, f_2^x, \ldots, f_N^x) : \mathscr{X} \to \mathscr{F}$ describes the non-linear mapping from the input space to the feature space for the input variable x [69]. In practice, $\mathscr{F}$ is not calculated directly. The *kernel trick* [71] is used to calculate the mapping, where a kernel function $\mathscr{K}$ calculates the inner products of the higher dimensional data samples: $\mathscr{K}(x_i, x_j) =< \Phi(x_i), \Phi(x_j) >, \forall i, j \in \{1, 2, \ldots, N\}$.

OSVM tries to find the hyperplane that separates the training data from the origin with maximum margin. It can be modeled by the following optimization problem:

$$\begin{aligned} &\min_{\mathbf{w}'\neq 0,\rho} \ \frac{1}{2}\mathbf{w}'^T\mathbf{w}' - \rho + \frac{1}{\nu N}\sum_{i=1}^{N}\xi_i, \\ &s.t. \ \ \mathbf{w}'^T\Phi(x_i) \geq \rho - \xi_i, \ \xi_i \geq 0 \ \forall i = 1,\ldots N. \end{aligned} \tag{4.26}$$

Here, ξ_i are the slack variables to the optimization problem. The solutions to this problem $\mathbf{w}'^*$ and ρ^* will result in evaluating the objective function of the form of Equation 4.25 to 1 for most of the training dataset. $\nu \in (0,1]$ is the key parameter that controls the fraction of outliers and that of support vectors (SVs). It has been shown in [19] ν is simultaneously the upper bound on the fraction of outliers and the lower bound on the fraction of SVs. By controlling the value of ν in the training phase, one can control the confidence on the training dataset directly. If the training dataset is very reliable, ν can be set to a low value so that the whole training dataset is accepted. On the other hand, if it is not known whether or not the training dataset truly represent the target class, ν can be set to some higher value.

To solve the OSVM optimization problem, the dual problem is formulated first. We use Lagrange multipliers to find the dual problem [19]. By introducing the Lagrange variables, problem (4.26) becomes the following:

$$\begin{aligned} L(\mathbf{w}',\rho,\xi,\alpha',\beta) &= \frac{1}{2}\mathbf{w}'^T\mathbf{w}' - \rho + \frac{1}{\nu N}\sum_{i=1}^{N}\xi_i \\ &- \sum_{i=1}^{N}\alpha'_i(w'^T\Phi(x_i) - \rho + \xi_i) - \sum_{i=1}^{N}\beta_i\xi_i. \end{aligned} \tag{4.27}$$

Setting the derivatives to the primal variables to zero, we obtain:

$$\mathbf{w}' = \sum_{i=1}^{N}\alpha'_i\Phi(x_i), \tag{4.28}$$

and

$$\alpha'_i = \frac{1}{\nu N} - \beta_i \leq \frac{1}{\nu N}, \ \sum_{i=1}^{N}\alpha'_i = 1. \tag{4.29}$$

Substituting Equation 4.28 and Equation 4.29 into Equation 4.27, we find the dual problem of OSVM:

$$\begin{aligned} &\min_{\alpha'} \alpha'^T \mathbf{Q} \alpha' \\ &s.t.\;\; 0 \le \alpha'_i \le \frac{1}{\nu N},\; \sum_{i=1}^{N} \alpha'_i = 1. \end{aligned} \tag{4.30}$$

Here, for clarity, we have used the vectorized form of $\alpha' = (\alpha'_1, \ldots, \alpha'_N)$. $\mathbf{Q}$ is the kernel matrix for the training data i.e.:

$$\begin{aligned} &\mathbf{Q}(i,j) = \mathscr{K}(x_i, x_j), \\ &i = 1, \ldots, N;\;\; j = 1, \ldots, N. \end{aligned} \tag{4.31}$$

Now, $\mathbf{w}'$ can be recovered using Equation 4.28.

As stated before, OSVM is a boundary-based method which only considers the boundary data points (SVs) to build a model of the training data distribution. The small variance projectional directions do not get any special consideration, which can result in better classification performance. In fact, it has been shown in [79] that maximum margin methods like OSVM tend to separate the classes along the high variance directions. To keep the robustness of the OSVM classifier intact while emphasizing the small variance directions, we incorporate the covariance matrix into the objective function of the OSVM optimization problem, as discussed in the next section.

4.3.3 The COSVM method

The purpose of our proposed method is to provide more importance towards the low variance directions. We do this by incorporating the estimated covariance matrix of our target class.

Here, to avoid any confusion with the original OSVM as described above, we will use a different set of notations to derive our covariance term. For this term, we assume that the solution weight vector is $\mathbf{w}''$. Later when merging the covariance term with OSVM, we will unify the notations together.

The intuitive motivation behind incorporating the covariance matrix was based on the observation that the OSVM optimization problem is a minimization problem (Equation 4.26).

The estimated covariance matrix of the training data contains all projectional directions, from high variance to low variance. We can assume that if we plug the covariance matrix into the optimization problem of OSVM, during the optimization algorithm, the components representing high variance projectional directions will be penalized more than components representing low variance directions. Hence, when the optimization problem is finally solved, the weight vector $\mathbf{w}''$ will be adjusted in a way that low variance directions are emphasized more.

Before proceeding, we have to estimate the covariance matrix in the kernel space. Let, the covariance matrix in kernel space (or the "kernel covariance matrix") is denoted by Σ_Φ:

$$\Sigma_\Phi = \sum_{i=1}^{N} (\Phi(x_i) - m_\Phi)(\Phi(x_i) - m_\Phi)^T, \tag{4.32}$$

where m_Φ is the mean of the training data calculated in feature space:

$$m_\Phi = \frac{1}{N} \sum_{i=1}^{N} \Phi(x_i). \tag{4.33}$$

With this definition, we can naively try to incorporate the covariance matrix as an additional term $\mathbf{w}''^T \Sigma_\Phi \mathbf{w}''$ in the objective function of the optimization problem of OSVM (Equation 4.26) (with a little abuse of notation, please note that eventually when we merge OSVM with our covariance term, we will have one unified weight vector). However, there are two issues with incorporating the covariance matrix directly into the primal optimization problem:

- We want to keep the changes to the original OSVM problem as minimal as possible, so that the existing OSVM implementations can be easily used. The dual problem is the one that is solved through some optimization algorithm for OSVM, not the primal one. Hence, it makes more sense to incorporate the covariance matrix directly in the dual problem.

- Although Equation 4.32 provides a form of the covariance matrix in kernel space, this form is not directly computable. We have to use the kernel trick as described before to represent the additional term $\mathbf{w}''^T \Sigma_\Phi \mathbf{w}''$ in terms of dot products only.

Hence, we plug the covariance matrix into the dual problem rather than the primal one.

Since the form of the kernel covariance matrix presented in Equation 4.32 cannot be directly integrated, we need to use the kernel trick to represent $\mathbf{w}''^T\Sigma_\Phi\mathbf{w}''$ in terms of dot products only [70]. From the theory of reproducing kernels [118], we know that any solution $\mathbf{w}''$ must lie in the span of all training samples. Hence, we can find an expansion of $\mathbf{w}''$ of the form:

$$\mathbf{w}'' = \sum_{i=1}^{N} \alpha_i''\Phi(x_i). \tag{4.34}$$

Now our target is to find a form of $\mathbf{w}''^T\Sigma_\Phi\mathbf{w}''$ in terms of dot products only. By using the definitions of Σ_Φ (Equation 4.32), m_Φ (Equation 4.33) and the kernel function $\mathscr{K}(x_i,x_j) =< \Phi(x_i),\Phi(x_j) >, \forall i,j \in \{1,2,\ldots,N\}$, we can derive the dot product form as follows:

$$\begin{aligned}
\mathbf{w}''^T\Sigma_\Phi\mathbf{w}'' &= \left(\sum_{i=1}^{N}\alpha_i''\Phi^T(x_i)\right)\left(\sum_{j=1}^{N}(\Phi(x_j)-m_\Phi)(\Phi(x_j)-m_\Phi)^T\right)\left(\sum_{k=1}^{N}\alpha_k''\Phi(x_k)\right) \\
&= \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\alpha_i''\Phi^T(x_i)(\Phi(x_j)-m_\Phi)(\Phi(x_j)-m_\Phi)^T\alpha_k''\Phi(x_k) \\
&= \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\left(\alpha_i''\mathscr{K}(x_i,x_j)-\frac{1}{N}\sum_{l=1}^{N}\alpha_i''\mathscr{K}(x_i,x_l)\right)\left(\alpha_k''\mathscr{K}(x_k,x_j)\right. \\
&\qquad\qquad\qquad\qquad\qquad\qquad\left.-\frac{1}{N}\sum_{m=1}^{N}\alpha_k''\mathscr{K}(x_k,x_m)\right) \\
&= \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\left(\alpha_i''\alpha_k''\mathscr{K}(x_i,x_j)\mathscr{K}(x_k,x_j)-\frac{2\alpha_i''\alpha_k''}{N}\sum_{l=1}^{N}\mathscr{K}(x_i,x_j)\mathscr{K}(x_k,x_l)\right. \\
&\qquad\qquad\qquad\qquad\qquad\qquad\left.+\frac{\alpha_i''\alpha_k''}{N^2}\sum_{l=1}^{N}\sum_{m=1}^{N}\mathscr{K}(x_i,x_l)\mathscr{K}(x_k,x_m)\right) \\
&= \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\left(\alpha_i''\alpha_k''\mathscr{K}(x_i,x_j)\mathscr{K}(x_k,x_j)-\frac{\alpha_i''\alpha_k''}{N}\sum_{l=1}^{N}\mathscr{K}(x_i,x_j)\mathscr{K}(x_k,x_l)\right) \\
&= \alpha''^T\mathbf{Q}\mathbf{Q}^T\alpha'' - \alpha''^T\mathbf{Q}1_N\mathbf{Q}^T\alpha'' \\
&= \alpha''^T\mathbf{Q}(I-1_N)\mathbf{Q}^T\alpha'' \\
&= \alpha''^T\Delta\alpha''
\end{aligned} \tag{4.35}$$

In the above derivation, the vectorized form of $\alpha'' = (\alpha_1'',\ldots,\alpha_N'')$ is used in the end for simplification. $\mathbf{Q}$ is the kernel matrix as defined in Equation 4.31. I is the identity matrix and

1_N is a matrix with all entries $\frac{1}{N}$.

Δ is the transformed version of Σ_Φ to be used in the dual form:

$$\Delta = \mathbf{Q}(I - 1_N)\mathbf{Q}^T. \tag{4.36}$$

This form of kernel covariance matrix Δ is only in terms of the kernel function and can be calculated easily using the kernel trick.

To explain the motivation behind incorporating the covariance matrix further, we can expand $\alpha''^T\Delta\alpha''$ as follows:

$$\begin{aligned}\alpha''^T\Delta\alpha'' &= \sum_{i=1}^{N}\alpha''^T((\mathbf{Q})_i - M)((\mathbf{Q})_i - M)^T\alpha'' \\ &= \sum_{i=1}^{N}\left(\alpha''^T((\mathbf{Q})_i - M)\right)^2\end{aligned} \tag{4.37}$$

Here $(\mathbf{Q})_i$ defines each column of the kernel matrix $\mathbf{Q}$ (Equation 4.31) i.e. $(\mathbf{Q})_i = (\mathscr{K}(x_1,x_i),\mathscr{K}(x_2,x_i),\ldots$
$1,\ldots,N$.

M is the “kernel mean”, which is an N dimensional vector. Each component of M is defined as [70]:

$$(M)_j = \frac{1}{N}\sum_{i=1}^{N}\mathscr{K}(x_i,x_j),\ \ \forall j = 1,\ldots,N. \tag{4.38}$$

We can see from Equation 4.37 that the term $\alpha''^T\Delta\alpha''$ provides a measure of closeness of the projected data points in the kernel space [72]. The lower this term is, the closer the data points are. As a result, if we *minimize* our objective function with $\alpha''^T\Delta\alpha''$ as an additional term, the low variance directions among the data points will indeed be emphasized.

Now we can incorporate it into the OSVM dual problem. To merge the two sets of notations for OSVM and our covariance term, we will replace α'' and α' with a single notation α. Hence, our target covariance term to incorporate into the OSVM dual problem is $\alpha^T\Delta\alpha$.

However, A method to control the contribution of the covariance matrix is necessary since we still want a balance between the importance given to the kernel matrix and the covariance matrix. We control the factor by which the kernel matrix $\mathbf{Q}$ and the covariance matrix contributes to the objective function through our control parameter η. The parameter should be used in such a way that changing only that value will provide a quick control over how the degree of emphasis should be spread between the OSVM part of our proposed method and the covariance matrix part. We can easily do this by replacing the term $\alpha^T \mathbf{Q} \alpha$ in Equation 4.30 by our new term $\alpha^T \eta \mathbf{Q} \alpha + \alpha^T (1-\eta) \Delta \alpha$. The major changes are the inclusion of the parameter η and the additional term with the covariance matrix.

With this replacement, our proposed Covariance-guided OSVM (COSVM) method can be described by the following optimization problem:

$$\begin{aligned} &\min_{\alpha} \alpha^T \eta \mathbf{Q} \alpha + \alpha^T (1-\eta) \Delta \alpha \\ &s.t. \;\; 0 \leq \alpha_i \leq \frac{1}{\nu N}, \;\; \sum_{i=1}^{N} \alpha_i = 1. \end{aligned} \tag{4.39}$$

If we compare this with Equation 4.30, we see that we are changing the objective function of the optimization problem to incorporate kernel covariance matrix Δ.

The COSVM formulation tries to maximize the margin of the separating hyperplane while simultaneously minimizing the scatter of the data projected to the normal direction of the hyperplane [72]. The tradeoff, i.e. the extent of “contribution” of our kernel matrix $\mathbf{Q}$ and the covariance matrix Δ in Equation 4.39 is controlled by the parameter η which can take values from 0 to 1. A value of 0 results in ignoring the kernel matrix completely and optimizing based on only the covariance matrix, while a value of 1 results in the opposite. For real-world problems, a value in-between will strike the perfect balance between our kernel matrix and the small variance directions obtained through the covariance matrix. In this way, the problem with OSVM providing preferential treatment towards the high variance projectional directions [79] will be alleviated and the lower variance directions will be provided more importance. This in turns can result in better performance, as we will see from the experimental results.

The COSVM optimization problem from Equation 4.39 can be written in a more concise form as below:

$$\min_{\alpha} \alpha^T (\eta \mathbf{Q} + (1-\eta)\Delta)\alpha \tag{4.40}$$
$$s.t.\ \ 0 \leq \alpha_i \leq \frac{1}{vN},\ \sum_{i=1}^{N} \alpha_i = 1.$$

The proposed method still results in a convex optimization problem since both the kernel matrix $\mathbf{Q}$ and the covariance matrix Δ are positive definite [119, 120]. As a result, the solution to this optimization problem will have one global optimum solution and can be solved efficiently using numerical methods.

Fig. 4.2-4.5 show schematic depictions of how our propose method changes the direction of the separating hyperplane in the kernel space optimally in four different cases and how it can result in better classification results. These figures are presented to show the adaptability of COSVM to different cases where the optimal decision hyperplane lies in different variance directions.

Fig. 4.2 shows the case where the optimal decision hyperplane for the example target data is parallel to the direction of high variance. COSVM can easily adapt to this case by setting the value of the control parameter η to 1 which means completely ignoring the covariance matrix term in the COSVM method (Equation 4.40). As a result, the low variance directions will not be given any special consideration in this case. This shows that our method can only do better than the original OSVM method but not otherwise.

On the other hand, Fig. 4.3 is the case when the optimal decision hyperplane will be the one parallel to the direction of low variance. In this case, η can be set to 0 in COSVM, which will only consider the covariance matrix and ignore the kernel matrix (the OSVM term). As a result, the decision hyperplane will be aligned properly in parallel to the direction of low variance.

Fig. 4.4 and Fig. 4.5 shows the general cases when $0 < \eta < 1$. These cases are usually more frequent for real-world datasets since the optimal decision hyperplane is highly unlikely to be entirely parallel to the direction of low variance or high variance. We see that by properly tuning the value of η, we can align the hyperplane in directions that will result in less overlap between the linear projections of the target data and the outlier data. In Fig. 4.4, the OSVM hyperplane (represented by dotted line) results in linear projections that result in some overlap between the example target and outlier data points (circled with dotted boundary). But due to

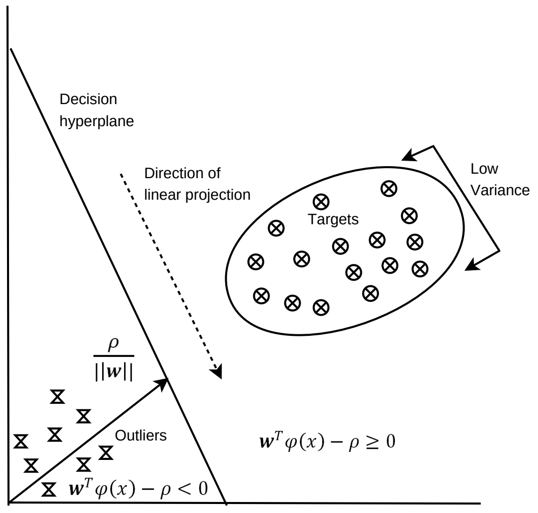

Decision hyperplane
Direction of linear projection
Low Variance
Targets
$\frac{\rho}{||\boldsymbol{w}||}$
Outliers
$\boldsymbol{w}^T\varphi(x) - \rho \geq 0$
$\boldsymbol{w}^T\varphi(x) - \rho < 0$

Figure 4.2: Case 1: Schematic depiction of the decision hyperplane for COSVM when the optimal linear projection would be along the direction of high variance. Optimal control parameter value for COSVM in this case is $\eta = 1$.

the extra importance given to the lower variance directions, the hyperplane for COSVM (the solid line) is pulled towards the direction of low variance. As a result, the linear projections resulting from the COSVM method (circled by solid boundaries) does not result in any overlap at all. In Fig. 4.5 we see another case where COSVM actually results in some overlap. Still, by using optimal η value, COSVM can reduce the overlap by a considerable margin when compared to the huge overlap from OSVM projections.

One important step for achieving better classification with COSVM is finding the appropriate value for η. Since in most cases, one-class problems do not have outlier examples, the value of η can't be tuned via cross validation. We use an indirect approach to optimize η which will be explained in detail in the next section.

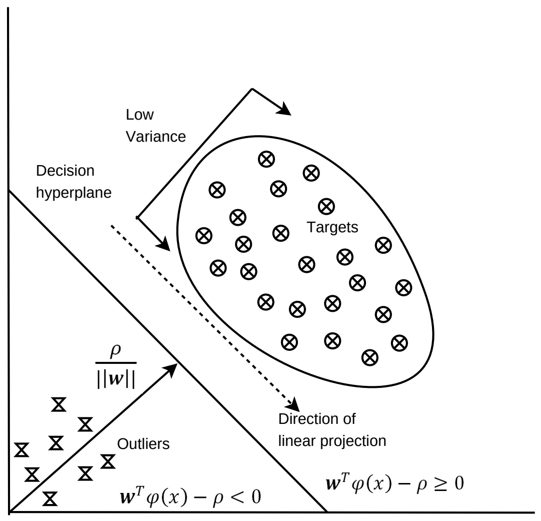


Figure 4.3: Case 2: Schematic depiction of the decision hyperplane for COSVM when the optimal linear projection would be along the direction of low variance. Optimal control parameter value for COSVM in this case is $\eta = 0$.

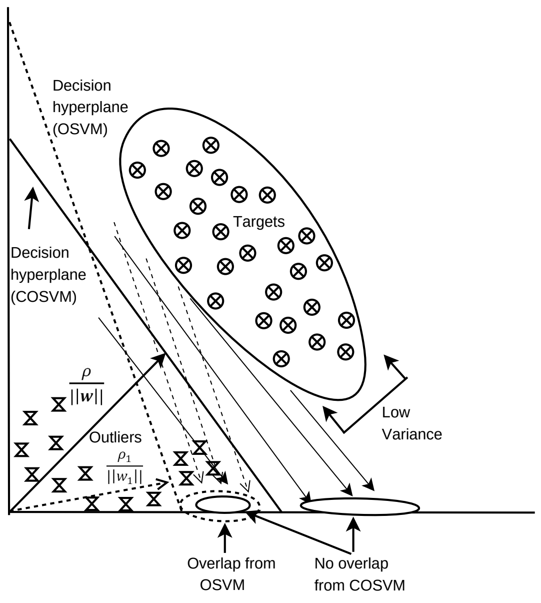


Figure 4.4: General Case 1: The optimal parameter value lies in between 0 and 1 ($0 < \eta < 1$). The linear projection direction for OSVM (depicted by dotted arrows) results in overlap between the example target and hypothetical outlier data (circled by dotted boundary), while an optimally tuned COSVM projection direction (depicted by solid arrows) does not result in any overlap (circled by solid boundaries).

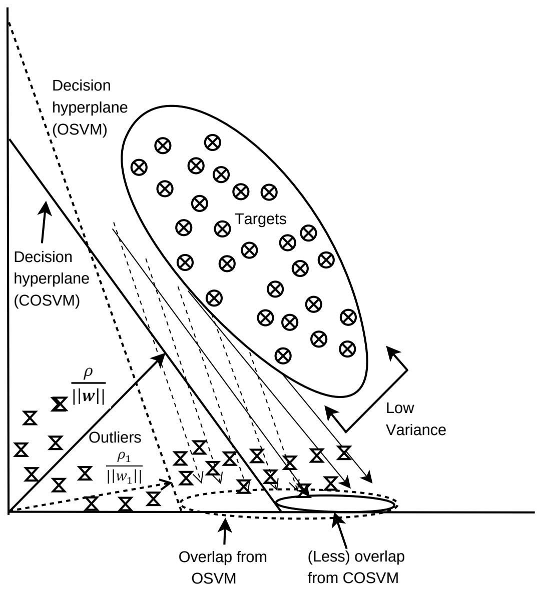


Figure 4.5: General Case 2: The optimal parameter value lies in between 0 and 1 ($0 < \eta < 1$). The linear projection direction for OSVM (depicted by dotted arrows) results in huge overlap between the example target and hypothetical outlier data (circled by dotted boundary), while an optimally tuned COSVM projection direction (depicted by solid arrows) results in much less overlap (circled by solid boundary).

4.3.4 Experimental Results

In this section, we present detailed experimental analysis and results for our proposed method, performed on both artificial and benchmark real-world one-class datasets, compared against contemporary one-class classifiers.

First, we provide an analysis of the effect of changing the value of η, our key control parameter (Equation 4.40). From this analysis, we decide upon how we will be optimizing the value of η for a particular dataset.

Since there is no straightforward way to optimize the value of η (e.g. cross-validation), we analyze the impact of changing the value of η first. Based on that, we present the method of optimizing η.

To visualize the effect of changing η, we generated an artificial Gaussian 2D dataset consisting of 200 data points. We then trained our proposed method with different η values. Since our main interest is to see the effect of varying η values, we kept the original one-class control parameter v fixed to a value of 0.2. Radial-basis kernel was used for kernelization of the data. This kernel is calculated as $\mathscr{K}(x_i,x_j) = e^{-\|x_i-x_j\|^2/\sigma}$. This kernel is proven to be robust and flexible [69]. Here, σ represents the positive “width” parameter. For this experiment, the value of σ was set to 1. Please note that later when comparing COSVM with other methods, σ was optimized first (Section 4.3.4). For this experiment, we only changed the value of η and kept everything else fixed for analysis.

Fig. 4.6 shows the different target boundaries obtained with different values of η. Fig. 4.6-(a) shows the boundary when $\eta = 1$. Please note that $\eta = 1$ refers to the case of original OSVM (check Equation 4.40 for clarification). Similarly, Fig. 4.6-(f) shows the boundary when $\eta = 0$; completely ignoring the OSVM kernel matrix and only taking the covariance matrix in account. The other four figures (Fig. 4.6-(b-e)) shows the intermediate cases when $0 < \eta < 1$.

As we can see, for intermediate values, the target boundary is being “expanded”. This is because the intermediate values distributes the importance between the OSVM term and the covariance matrix. Due to considering the covariance matrix, the low variance directions are being assigned more importance. As a result, the target boundary expands in those directions.

However, we need to use a stopping criterion to find the optimum η value. We use the fraction of outliers as our stopping criterion. The fraction of outliers is determined by calculating what fraction of the training samples are deemed as outliers by the constructed target boundary. We use a pre-defined lowest fraction of outliers allowed (f_{OL}) as stopping criterion. For a new dataset, we keep slowly decreasing the value of η (starting from 1) and observe the fraction of outliers. When it hits the value of f_{OL}, we stop and use the current η value for that particular dataset. As we will see from the benchmark results, this method of optimization provides superior performance by providing a smooth control.

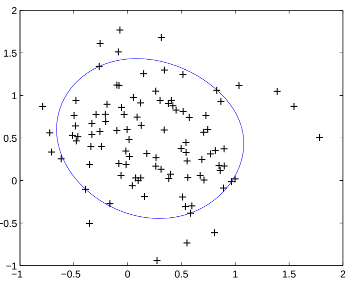
One important point to note here is that there is no direct interconnection between the value of f_{OL} and the value of the OSVM parameter v. The OSVM parameter v theoretically bounds the fraction of outliers from above. The fraction of outliers cannot be more than the preset value of v. On the other hand, f_{OL} is a stopping criterion for optimization of η which provides a lower bound on the fraction of outliers. There is no conflict between these two values, and they can be set independently to fit the purpose of the dataset to be trained. It is also not possible to provide any strict guidance on what these values should be set to. Even for OSVM, the value of v can be set to any value between 0 to 1 [19]. For COSVM, we have the additional parameter f_{OL} which can be set to any value between 0 to v (setting it to a value above v will not be beneficial since v is the theoretical upper bound on the fraction of outliers).

Now we present detailed experimental results to analyze the performance of COSVM compared against contemporary classifiers. We have divided this section into several paragraphs under different headings for clarity; namely, description of the datasets used, the experimental model for impartial results, description of the classifiers to be compared against and finally, discussion of the results.

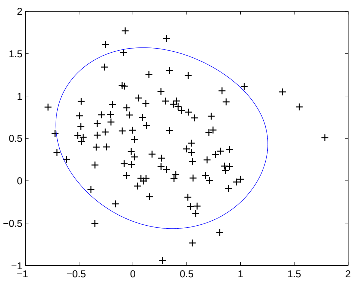
Datasets Used:

We have used both artificially generated datasets and real-world datasets in our experiments to test the robustness of COSVM in different scenarios.

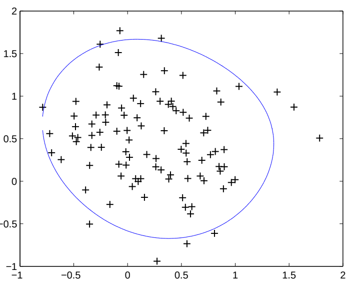
For the experiments on artificially generated data, we have created several sets of 2D two-class data drawn from three different set of distributions: 1) Gaussian distributions with different covariance matrices. 2) Gamma distributions with different shape and scale parameters and 3) Banana-shaped distributions with different variances. Each class contains 500 data points. For each distribution, two different sets were created, one with low overlap and the other with high overlap. The overlaps were varied by tweaking the distribution parameters (e.g. for the Gaussian distributions, the means of the two classes were pulled closer for higher overlap). Fig. 4.7 shows the plots of these generated datasets. Each class of each dataset was used as target class and outliers in turns. As a result, we had a total of 12 datasets. (3 distributions X 2 overlaps X 2 classes).



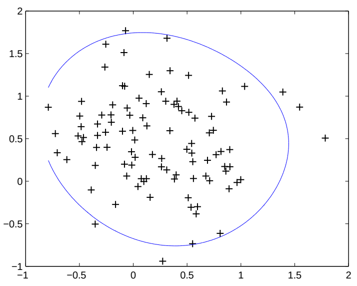
(a) $\eta = 1$



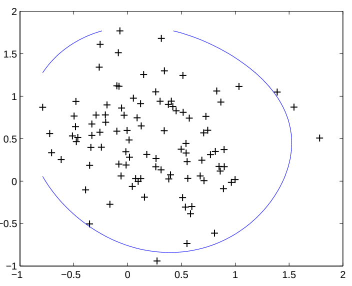
(b) $\eta = 0.8$



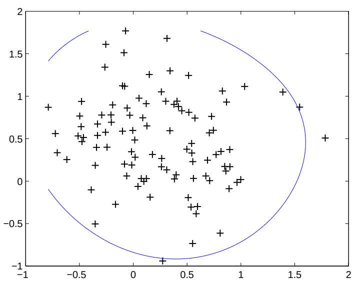
(c) $\eta = 0.6$



(d) $\eta = 0.4$

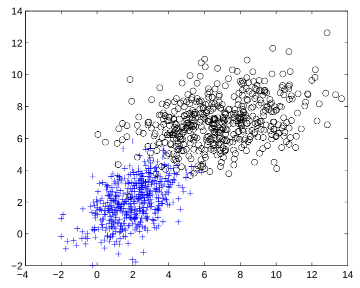


(e) $\eta = 0.2$

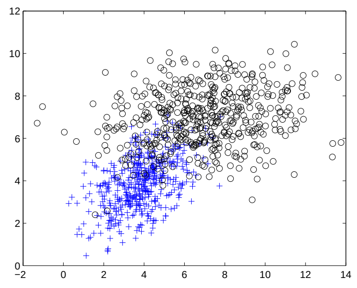


(f) $\eta = 0$

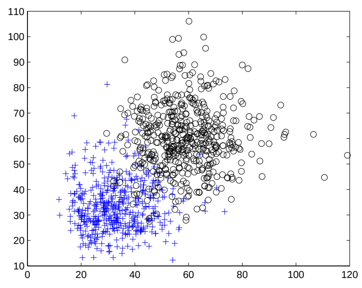
Figure 4.6: COSVM boundaries for different η values, constructed from an artificial Gaussian 2D Dataset.



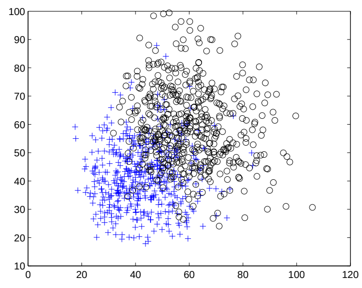
(a) Gaussian (low overlap)



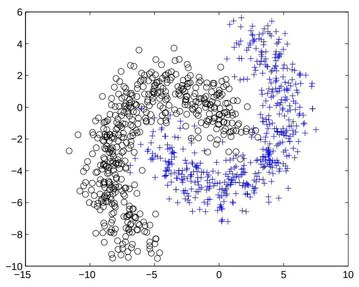
(b) Gaussian (high overlap)



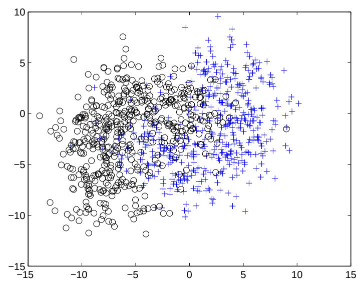
(c) Gamma (low overlap)



(d) Gamma (high overlap)



(e) Banana (low overlap)



(f) Banana (high overlap)

Figure 4.7: Artificial datasets used for comparison. The two shapes denote two different classes generated from a pre-defined distribution. Each class was used as target and outlier in turns.

For the real-world case, we have primarily focused on medical datasets in our experiments, as medical diagnosis is one of the key fields where one-class classification is of utmost importance [105]. We have also included several large scale datasets [121] to compare the training times of COSVM with OSVM. A detailed description of the datasets used can be found in Table 4.3[1].

Table 4.3: Description of Datasets.

Dataset Name	Number of Targets	Number of Outliers	Number of Features
Haberman's Survival	81	225	3
Breast Cancer	241	458	9
Biomedical	67	127	5
Liver (diseased)	145	200	6
Liver (healthy)	200	145	6
SPECT Images (normal)	95	254	44
SPECT Images (abnormal)	254	95	44
Gene Expression (healthy)	22	40	1908
Gene Expression (tumor)	40	22	1908
Adult	4361	2053	122
Protein-1	7379	10387	357
Protein-2	3659	14107	357
Protein-3	4953	12813	357

Most of these datasets were originally collected from the UCI machine learning repository [97]. Since these datasets are originally two-class or multi-class, each dataset was divided into separate classes. Then one class was used as target class and the other ones as outlier. This process was repeated for all classes. Some of the target and outlier sets were too trivial to classify. We have omitted those sets from our results. We have also carefully picked the datasets to be of varying size and dimensions, so that the robustness of COSVM against different feature sizes is tested. As we can see from Table 4.3, the dimensions vary from 3 to 1908, while the training set sizes vary from 22 to 7379.

Experimental Model:

To make sure that the achieved results are not biased or coincidental, we created 10 different training and testing sets for each dataset (both artificial and real). To build a model, about 10% randomly selected data from the target class was removed first. The rest 90% was used as the

[1]The first 9 datasets can be downloaded from `http://prlab.tudelft.nl/users/david-tax`. The last 6 large scale datasets can be downloaded from `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`

training data. The previously removed 10% was added to the outliers and this whole set was used for testing. This approach was repeated 10 times to build 10 different training and testing sets. The final result was achieved by averaging over these 10 models. This ensures that the achieved results were not a coincidence.

For one-class classifiers, usually Receiver Operating Characteristic(ROC) curves are used as a measurement of performance [122]. The ROC curve is created by plotting the True Positive Rate (TPR) vs the False Positive Rate (FPR). ROC curve presents a robust measurement of the classifier performance. It does not depend on the number of training data points or outlier data points, it depends rather on rates of correct and incorrect target detection (TPR and FPR) [123]. To quickly evaluate the performance of a classifier, the Area Under the Curve (AUC) for the ROCs [124] are calculated and presented in our results(Table 4.4 and Table 4.5).

Classifiers:

Our proposed method was compared against the following six contemporary classifiers:

- **Gaussian**: As the name suggests, this classifier models the target data with a single Gaussian density, and uses maximum likelihood estimation for the mean and covariance matrix. A regularization term is usually added to the covariance matrix for high dimensional data. No parameter is needed to be optimized for this classifier.

- **Parzen**: The Parzen density estimation or kernel density estimation technique [125] typically uses a kernel function to estimate the probability density of the training data. Typically, the Gaussian kernel is used for this purpose. The width of the kernel is found through optimizing the likelihood on the training set using leave-one-out procedure [126].

- **k Nearest Neighbors (k-NN)**: This method uses the distance measure from the k Nearest Neighbors to decide whether an incoming data point belongs to the target class or the outliers. The majority of the neighbors decide the belonging class. The parameter k needs to be optimized, which can be done using the leave-one-out density estimation [127].

- **Support Vector Data Description (SVDD)**: The Support Vector Data Description (SVDD) [103] is a boundary-based one class classifier which tries to define the hypersphere covering the training data points which has the minimum volume. The kernel trick can also be used with this classifier.

- **One-class SVM (OSVM)**: Since COSVM was built upon OSVM [19], this classifier has been described in detail in Section 4.3.2. The key difference between OSVM and SVDD is that SVDD tries to estimate the boundary around the training dataset directly with a hypersphere, while OSVM tries to find the maximum margin hyperplane that separates the training dataset from the origin. In this sense, OSVM is more similar to the classical two class SVM. In OSVM, the origin is assumed to be the sole member of the second class.
- **Mahalanobis One-class SVM (MOSVM)**: The Mahalanobis One-class SVM (MOSVM) [128] is an extension of the original OSVM algorithm where Mahalanobis distance is used as a measure instead of Euclidean distance to utilize the information available from data in the form of covariance matrix.

The k-NN, Parzen, Gaussian and SVDD classifiers were implemented with the help of DDtools [129]. For implementation of OSVM, MOSVM and COSVM, the SVM-KM toolbox was used [117]. The radial basis kernel was used for kernelization in SVDD, OSVM, MOSVM and COSVM. The kernel width parameter σ is crucial for these classifiers. σ determines what "scale" of data we will be using for training and testing. For smaller values of σ the data may not scale very well. A good indication of not scaling well is the fraction of SVs obtained from training stage [104]. If all the data points are being considered as SVs, the scaling isn't good and the classifier simply memorizes the data itself. Hence, a reasonable heuristic to optimize the value of σ is this: start with a small value, keep increasing and observe the fraction of SVs. Accept the value of σ for a particular dataset when the number of SVs does not decrease any further [19]. We do this separately for SVDD, OSVM, MOSVM and COSVM to find good values of σ for each dataset.

Similar to our proposed method, the MOSVM has a control parameter $\mathbf{C}$ to control the contribution of the covariance matrix [128]. However, unlike our method, the authors of [128] do not provide any clear guidance on how to set the value of $\mathbf{C}$. Since cross-validation should not be performed in a real problem where outlier data is usually not available, we don't have any easy way to find the optimal value of $\mathbf{C}$. Also, the value of $\mathbf{C}$ is not bound between 0 and 1 like our control parameter η. So, the search space to find optimal $\mathbf{C}$ can be arbitrarily large. To get the best result from this method, we repeat all the experiments with $\mathbf{C} = 10, 100, 200, \ldots, 1000$ and report the best results here (these values were picked by assessing the results reported in [128]). Although performing the experiments this way provides MOSVM with some unfair advantage over the other methods, as we will see from the results, even then the COSVM method mostly outperforms MOSVM.

The method of optimizing the value of our control parameter η was described in detail in Section 4.3.4. The parameter v for OSVM, MOSVM and COSVM was set to 0.2 (Equation 4.26). The similar parameter for SVDD is called *fraction of rejection* which was also set to 0.2 through DDtools [129].

The lowest threshold for the fraction of outliers (f_{OL}) while optimizing η was set to 0.1 (see Section 4.3.4 for details). It is important to note that setting different v and f_{OL} values for different datasets can result in even better performance for these particular experiments. However, we have set the same value of v and f_{OL} for all datasets because it is not possible to define optimal values for these parameters in case of a real-world setting since the future data points will be always unknown. For a practical application, these parameters can be adjusted and the system can be re-trained time-to-time if necessary.

Results and Discussion

Table 4.4 and Table 4.5 contain the average AUC values obtained for the classifiers on the artificial and real datasets, respectively. As we can see, COSVM mostly provides better results in terms of the obtained unbiased AUC values by averaging over 10 different models (details in Section 4.3.4). Specially in case of the real-world datasets, COSVM performs significantly better when compared to other methods in most cases. This strengthens our claim that by emphasizing the low variance directions with the incorporation of the covariance matrix, COSVM can indeed lead to improved performance. To further signify the performance enhancement by COSVM, we present some statistical measurements. We performed paired t-tests on the AUC values [98] by pairing up the COSVM method with each method at a time. The paired t-test determines whether or not two sets of measured values are significantly different. The last rows of Table 4.4 and Table 4.5 provides the confidence intervals (in %) obtained from the performed t-tests. This confidence interval quantifies the probability of the paired distributions being the same. The higher the confidence interval, the lower is the probability that the underlying distributions are statistically indifferent. As we can see, all the confidence intervals are high, which shows that COSVM indeed provides statistically significant accuracy improvements.

Table 4.4: Average AUC of each method for the 12 artificial datasets (best method in **bold**, second best *emphasized*). The last row contains the paired t-test confidence intervals.

Dataset	k-NN	Parzen	Gaussian	SVDD	OSVM	MOSVM	COSVM
Gauss. (low overlap)-1	99.23	99.21	99.17	99.33	99.33	*99.35*	**99.42**
Gauss. (low overlap)-2	98.12	98.17	98.02	97.62	98.20	*98.33*	**98.50**
Gauss. (high overlap)-1	94.04	94.04	94.04	*94.23*	*94.23*	93.82	**95.59**
Gauss. (high overlap)-2	84.17	85.30	**86.32**	84.53	85.18	*85.43*	**86.32**
Gamma (low overlap)-1	**99.55**	*99.48*	84.74	99.10	99.31	99.31	99.31
Gamma (low overlap)-2	**99.38**	**99.38**	85.27	98.85	*99.29*	*99.29*	*99.29*
Gamma (high overlap)-1	88.04	88.09	80.29	88.21	88.41	*88.61*	**89.11**
Gamma (high overlap)-2	88.63	88.13	81.21	88.29	88.28	*88.53*	**89.91**
Banana (low overlap)-1	95.60	*95.89*	95.40	*95.89*	95.71	95.80	**96.70**
Banana (low overlap)-2	94.23	94.80	95.05	95.41	96.11	*96.25*	**96.74**
Banana (high overlap)-1	84.06	*84.59*	82.90	84.00	**84.62**	83.91	**84.62**
Banana (high overlap)-2	76.31	76.86	80.03	80.11	82.54	*82.93*	**84.82**
Confidence	96.07	94.54	99.02	99.50	99.55	99.72	-

Table 4.5: Average AUC of each method for the 13 real-world datasets (best method in **bold**, second best *emphasized*). The last row contains the paired t-test confidence intervals.

Dataset	k-NN	Parzen	Gaussian	SVDD	OSVM	MOSVM	COSVM
Haberman's survival	43.33	44.26	51.36	56.93	*62.85*	62.37	**68.37**
Breast Cancer	82.62	72.96	86.04	*97.35*	96.97	97.05	**98.00**
Biomedical	36.83	40.02	64.66	81.38	*82.65*	82.12	**85.80**
Liver (diseased)	60.02	59.9	59.59	60.93	61.73	*62.19*	**65.16**
Liver (healthy)	50.34	49.69	50.76	62.5	*63.7*	60.63	**64.96**
SPECT (normal)	84.28	*96.45*	93.90	92.32	95.29	96.21	**96.79**
SPECT (abnormal)	19.74	41.29	26.39	57.95	*69.49*	68.85	**72.46**
Gene Exp. (healthy)	61.5	50	*68.125*	55.5	**72.81**	72.50	**72.81**
Gene Exp. (tumor)	68.86	50	60.68	67.84	72.48	*72.72*	**74.6**
Adult	56.44	55.60	59.13	66.45	68.58	*69.10*	**72.96**
Protein-1	50.66	52.36	52.05	67.14	*68.50*	68.11	**71.6**
Protein-2	50.37	53.78	55.04	65.59	68.27	*70.72*	**73.91**
Protein-3	54.3	55.37	55.01	66.11	66.49	*66.72*	**70.07**
Confidence	99.97	100	99.97	99.98	100	100	-

In general, we see that for real-world datasets (Table 4.5), the performance of k-NN, Gaussian and Parzen classifiers are poor when compared to the SVM-based classifiers (SVDD, OSVM, MOSVM and COSVM). This is because of the limitations inherent in these classifiers. Since k-NN classifies a data point solely based on its neighbors, it is sensitive to outliers [130]. The Gaussian classifier has some obvious limitations from the assumption that the underlying distribution is Gaussian, which is not always the case for real datasets. The Parzen classifier is prone to degraded performance in case of high-dimensional data or small sample size [131]. We can easily see this limitation of the Parzen classifier from the poor results on the Gene Expression datasets which has a very high dimension. The SVM-based classifiers are free from all these assumptions and, hence, leads to better result in majority of the cases.

However, in case of the artificial datasets (Table 4.4), we see that these three classifiers (i.e. k-NN, Gaussian and Parzen) are competitive with the SVM-based methods, sometimes even better. This is because the artificial datasets are generated from a pre-defined regular distribution. We see that the Gaussian method performs well for the cases where the dataset was generated from a Gaussian distribution, which is expected. It also performs comparatively well for the datasets generated from the banana distribution. This is because the banana distribution is actually generated by superimposing an underlying Gaussian distribution on a banana shape[132]. However, in case of Gamma distribution, it performs poorly since the distribution does not match with the assumption of the classifier.

We also see that for the low overlap cases (specially Gamma distribution), COSVM does not show any significant improvement, and is sometimes being outperformed. This can happen because in the low overlap cases, the SVM algorithm does not provide any significant advantage, and simpler algorithms like k-NN can perform well. However, in the high overlap cases, the improvement provided by COSVM is noticeable. These results match with the schematic depiction we presented in Section 4.3.3. As we have stated there, because of the importance we provide towards the low variance directions, the separating hyperplane in COSVM will be pulled towards the low variance directions and can reduce the overlap between targets and outliers that way. This is why COSVM improves upon the other methods in case of high overlap. Real-world datasets are expected to have overlaps. This is reflected in the results we got from the real-world datasets (Table 4.5), where the advantage of OSVM is more apparent than in the case of artificial datasets (Table 4.4).

In terms of comparison between SVDD and OSVM, we see that these two are quite competitive. There is no rigorous way to explain the comparative results between these two, as they are built from different philosophies. However, if we compare the results with COSVM, we see that it is clearly the better classifier in almost all the cases. As stated before, COSVM puts emphasis on the low variance directions by using the kernel covariance matrix in the minimization problem. The balance of the emphasis obtained through optimizing the control parameter η results in better performance.

The performance of MOSVM is competitive with SVDD and OSVM. However, in some of the cases MOSVM is outperformed by OSVM (e.g. Banana (high overlap)-1 in Table 4.4 and Liver (healthy) in Table 4.5). As discussed before, the MOSVM method performs inversion of both the kernel matrix and the covariance matrix, which can result in occasional inaccuracies [86]. It might also perform poorly because of suboptimal choice of the control parameter $\mathbf{C}$, since no guidance for optimization or suggested range for the parameter value is provided in [128]. But in COSVM the covariance matrix is plugged into the OSVM minimization problem without changing the basic OSVM formulation. As a result, we see that COSVM can only perform better than OSVM, not worse.

In terms of training computational complexity, COSVM does not add any overhead on top of the original OSVM method except for computing the covariance matrix, which can be done as part of pre-processing and re-used throughout the training phase. The complexity for solving the convex optimization problem imposed by COSVM is $O(N^3)$, where N is the number of training data points [19]. This complexity can be reduced by taking the simplicity of the optimization constraints into account [19]. There are also some existing algorithms for the two-class SVM which can solve the optimization problem in linear [133] or even sub-linear time [134]. In [133], a smoothed version of the so-called hinge-loss function of SVM is presented and then a gradient method is provided to solve the optimization problem. A stochastic primal-dual approach is presented in [134], which can achieve sub-linear training time for SVM. Since our problem is based on OSVM rather than SVM, these approaches are not directly applicable. However, they can be carefully adopted to be used with one-class problems, which can provide better running times for our proposed COSVM approach.

Table 4.6 shows the average training times per model for both the artificial and the real-world datasets. As we expect, COSVM performs almost as fast as OSVM, while providing the advantage of emphasizing the low variance directions.

Table 4.6: Average training times (per model) in *seconds* for OSVM and COSVM for the experiments on the artificial and real-world datasets.

Experiment	OSVM	COSVM
Artificial Datasets	0.114	0.131
Real-world Datasets	1235.7	1246.2

4.3.5 Summary

We have proposed a Covariance-guided One-class Support Vector Machine (COSVM) classification approach. COSVM improves upon the One-Class Support Vector Machine (OSVM) [19] method by emphasizing low variance projectional directions of the training dataset with the incorporation of the covariance matrix into the minimization problem of the algorithm. A parameter controls the degree of emphasis, which can be tuned efficiently for optimum performance. COSVM does not sacrifice any important information and retains the robustness of support vector classification. The proposed method results in a convex optimization problem which can be solved efficiently with existing numerical methods to obtain a unique global optimum solution. Our proposed method does not change the basic formulation of the OSVM method. As a result, it can be easily implemented with the existing OSVM libraries. We've presented detailed experimental results on several artificial and real-world benchmark datasets, where we compare the COSVM method against six other contemporary one-class classifiers. The results show the superiority of COSVM which provides significantly improved performance over the other classifiers.

4.4 Summary

In this chapter, we explored the possibilities of combining distributional information from different groups of classifiers. We argued that both the global information available through discriminant-based classifiers like LDA and local information available through boundary-based classifiers like SVM are important to achieve good classification results. We validated our argument by proposing two novel supervised classification methods. The first one is the Kernel Nonparametic Support Vecot Machine (KN-SVM) classifier, which combines information from Kernel Nonparametric Discriminant (KND) and Kernel Support Vector Machine

(KSVM). The second method was proposed for the field of One-Class Classification. The proposed Covariance-Guided One-Class Support Vector Machine (COSVM) incorporates the covariance information obtained from the training data into the one-class SVM classification method.

For both the classifiers, a control parameter elegantly controls the direction of the decision hyperplane so that optimal classification can be achieved. We have provided detailed experimental results on both real and artificial datasets, where we compare the proposed classifiers with the state-of-the-art. The results clearly validated our initial assumption that combining local and global information can indeed provide us with better classification.

Chapter 5

Image-centric Approach Towards Volume Rendering

5.1 Introduction

In the previous chapter, we have proposed the KN-SVM classifier, which is a novel supervised classification method that combines local and global distributional information from the training data. In this chapter, this method is applied to develop an new image-centric approach towards volume rendering.

To recap, in image-centric volume rendering methods, the user directly interacts with the volume image/slices to select the regions he/she is interested in. The user selection is used as training data after feature extraction. The rest of the volume voxels are then classified accordingly, and a rendering of the volume is shown to the user. A big advantage of the image-centric approach is that the user will directly work in the image domain, which is already familiar to him/her. However, this has some drawbacks too. In the experimental results for our image-centric approach, we will present a qualitative discussion accompanied by a user survey where we compare our proposed data-centric approach and image-centric approach.

In our image-centric approach, the user is first presented with grayscale form of some slices from the volume data, where he/she can do simple selection on voxels to express his/her intention of how the volume should be classified. To further simplify the process, we carefully pick the most representative slices from the volume and only show those to the user. The slices are picked by sorting them based on image entropy, which provides a measure of information present in one slice [135]. Once the user selection is completed, we treat the selected voxels as training data and extract some high-dimensional features. Our proposed KN-SVM classifier is then used to classify the rest of the voxels into different groups. The combination of global and local distributional information results in an accurate classification. Also, due to the robustness of the classifier, only a small number of training samples can provide excellent results, as we will see from the experiments. After the voxel classification, the classes are mapped to different color and opacity values automatically by using the concept of color harmonization [20] as we have done in our data-centric approach. This automated color generation scheme can generate easily distinguishable and aesthetically pleasing visualization of the underlying classes.

5.2 Proposed Method

Figure 5.1 shows the high level system diagram for the proposed method. As can be seen, the *most informative* slices of the volume are extracted and presented to the user. The user selects voxels and assigns them in different classes based on what he/she wants to see. Features are extracted from the voxels and used as training data for the proposed KN-SVM classification. The voxel classes are then assigned different color and opacity values based on color harmonization. The various modules of the proposed method are described in the next sections.

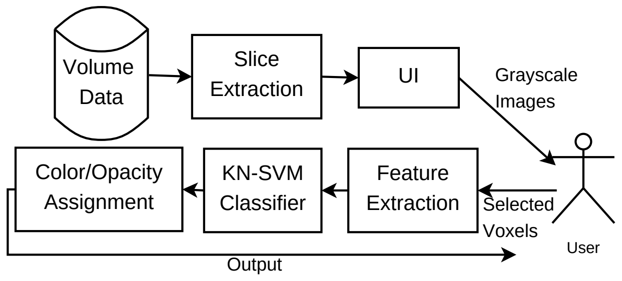


Figure 5.1: System diagram for the proposed method.

5.2.1 Slice Extraction

Since the user will perform selection operations on volume slices, we need to provide the user with the *most informative* slices. A typical volume can have thousands of slices (along X, Y or Z direction). We need to provide the user with the slices that contain the most variation, since we can safely assume that these slices will contain all the structures that the user might be interested in [135]. For this, we calculate the *image entropy* of each slice. In general, for a set of M symbols with probabilities $p_1, p_2, \ldots, p_M$ the entropy can be calculated as follows [135]:

$$H = -\sum_{i=1}^{M} p_i \log p_i. \tag{5.1}$$

For an image (a single slice), the entropy can be similarly calculated from the histogram [135]. The entropy provides a measure of variation in a slice. Hence, if we sort the slices in terms of entropy in descending order, the slices with the highest entropy values can be considered as representative slices. To provide the user with more choices, the three slices with highest entropies along the X, Y and Z directions are shown to the user.

5.2.2 User Input

The user is presented with a GUI (Figure 5.2), which shows the three slices with highest entropy values in X, Y and Z direction. The user can then select different voxels and assign them to different classes, which will be treated as training data for the next steps in the method. As we can see from Figure 5.2, the regions assigned to different classes by the user are marked with different colors. The dataset shown here is the Foot CT scan dataset as we have used in chapter 3. The details of the datasets are also repeated in the experimental results section of this chapter. The objective with the foot datasets is to roughly separate the bones (Class 1), joints (Class 2), and the outer layer (Class 3) in the volume. Please note that although the number of training voxels seems small, our proposed KN-SVM classification method can classify the whole volume from a small training set reliably by combining both local and near-global distributional information as we have discussed in the previous chapter.

5.2.3 Feature Extraction

Unlike histogram-based methods, our approach does not restrict the feature dimension. Therefore, we can use a reasonably high-dimensional feature set. The features need to be picked carefully so that in the classifier stage, there is enough distinction between different classes. If we recall, in our data-centric approach (chapter 3 section 3.3), the features being used were

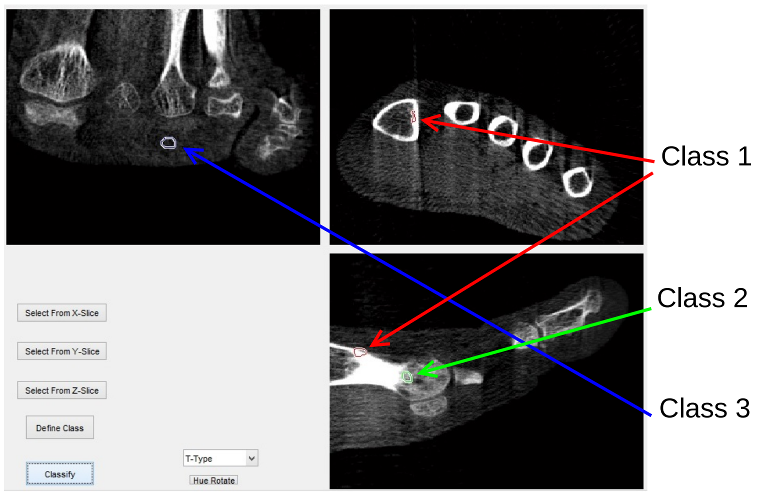


Figure 5.2: The GUI with extracted slices. The red, green and blue selections (circles) by the user refer to voxels assigned to Class 1, Class 2, and Class 3, respectively.

intensity, 3D gradient magnitude, the X, Y and Z locations of the voxels and the second derivative. However, we have found that using the same features for our image-centric approach did not provide satisfactory results. The reason behind this discrepancy is the way the voxels are being classified. In the data-centric approach, the clustering process is unsupervised. But in the image-centric approach, the training voxels are picked by the user from the grayscale representation of the volume slices. Since the grayscale images are generated from intensity values alone, we have found that we need more emphasis on the local variances in intensity to capture the distinctive properties of each class. In other words, the intensity values of the neighboring voxels need to be taken into consideration. As a result, the average neighboring voxel values (in all three directions) are used as separate features in our image-centric approach. We have also found that due the strong localization capabilities of these intensity driven features, the X, Y, Z locations and the second derivative value are not required to be included in the feature set to achieve satisfactory results. Dropping these redundant features, the 5D feature space for the proposed image-centric method consist of:

- the intensity value,
- the 3D gradient magnitude of each voxel, defined by:

$$G = \sqrt{G_x^2 + G_y^2 + G_z^2}, \tag{5.2}$$

where G_x,G_y and G_z are the gradient values along X, Y and Z direction, respectively.

- The average intensity value of the neighboring 8 voxels in X, Y and Z direction, respectively. The three average values along different directions are treated as different features.

These features provide us with reasonable localization of voxel attributes, which helps separating different structures in the volume in the classification stage.

5.2.4 KN-SVM Classifier

Our proposed KN-SVM classifier was described in detail in the previous chapter. It is motivated by combining the merits of both discriminant-based classifier such as KND [70] and the classical SVM. The KND calculates the within-class scatter matrix by considering the κ-nearest neighbors for each training data point. Thus it considers the "global" characteristics of the training distribution. On the other hand, SVM only considers the "local" characteristics (support vectors) to build the separating hyperplane. In KN-SVM, these two are combined by incorporating the within-class scatter matrix and the between-class scatter matrix ∇ of KND into the SVM optimization problem. The optimization problem of KN-SVM was derived in chapter 4 Equation 4.13. The equation is repeated here for reference (for details on the specific terminologies, please refer to chapter 4):

$$\begin{aligned}\min_{\mathbf{w}\neq 0, w_0} & \left\{ \frac{1}{2}\mathbf{w}^T(\eta\Delta(\nabla+\beta I)^{-1}\Delta+I)\mathbf{w} \right. \\ & \left. +C\sum_{i=1}^{N} max(0, 1-t_i(\Phi^T(x_i)\mathbf{w}+w_0)) \right\}. \end{aligned} \tag{5.3}$$

In this equation, η is the control parameter which dictates the amount of contribution from SVM and KND. By using an appropriate value of η, we can control the direction of the separating hyperplane of the classifier and place it in an optimum way.

The value of η is set to 0.3 through experiments in the proposed system. This value was determined through trial and error. It should be noted that slight variations in this value does not result in any drastic change in the volume visualization. Since the problem can be multi-class, we have used the one-vs-one classification scheme with our KN-SVM method.

5.2.5 Color and Opacity Assignment

As discussed in this thesis before, manual color and opacity assignment can be a hindrance towards achieving good volume visualization. An automated coloring method is a necessary component.

If we compare our data-centric and image-centric approaches, we see that the principal difference is how the voxels are classified into different groups. Once this stage is passed, the problem essentially reduces down to color and opacity assignment to different voxel groups. Hence, we have opted for the same color and opacity assignment rules as devised in chapter 3 section 3.5.1. In short, the colors are generated by using color harmonization, which is a concept borrowed from arts [44]. The harmonic colors give us a set of aesthetically pleasing hue values which can be assigned to different voxel groups. The S and V values for the HSV color space and the opacity value are also derived from the user's perception as described in Equation 3.8, 3.9 and 3.10 in chapter 3.

Once we have the color and opacity values, the visualization toolkit [57] was used for fast GPU-based rendering.

5.3 Experimental Results

The same 5 datasets used in our data-centric approach is used to evaluate the proposed method. The datasets are: CT scans of a Foot, the Visual Male Head dataset, the Carp dataset, the Engine dataset and the Lobster dataset. Details of the dataset are repeated in Table 5.1 for convenience. To speed up the classification process, we threshold these datasets with a value close to zero so that the air surrounding the actual data are not passed on to the classifier.

Figure 5.3 shows the result obtained based on the training data shown in Figure 5.2. Here, we see that our KN-SVM method can separate the bones, joints and the outer layer of the foot effectively. Figure 5.3-(b) shows that due to the use of intelligent color and opacity assignment, all three classes can be visualized at the same time and easily distinguishable.

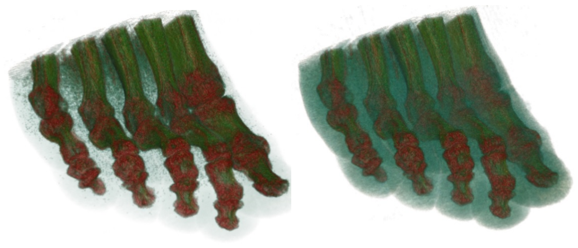


Figure 5.3: Results for the Foot dataset.

Figure 5.4 shows comparison between KN-SVM and the classical SVM (based on the same training data). The bones and joints are shown together. We can see the advantage of KN-SVM when compared to SVM. In the areas pointed by arrows, the SVM was unable to accurately separate the joints from the bones, while the KN-SVM method was successful. This shows the superiority of the combined approach in KN-SVM. Although the KN-SVM result may look noisy in some areas, the target here is to show the effectiveness in separating the bones and joints. The apparently cleaner output from SVM can actually mislead the user in thinking this is accurate.

The rendering results for the other datasets are shown in Figure 5.5, 5.6 and 5.7. All these rendering results were achieved by defining two voxel classes with our GUI. We can see that like our data-centric approach, the image-centric method can provide effective rendering results on these datasets easily and efficiently.

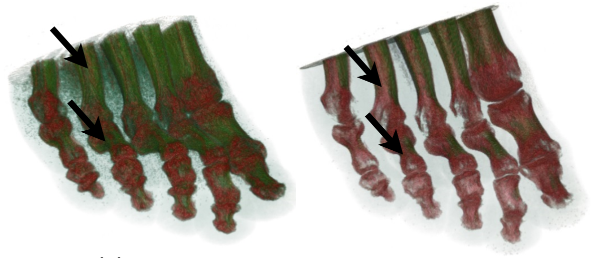


Figure 5.4: Comparison between KN-SVM and SVM.

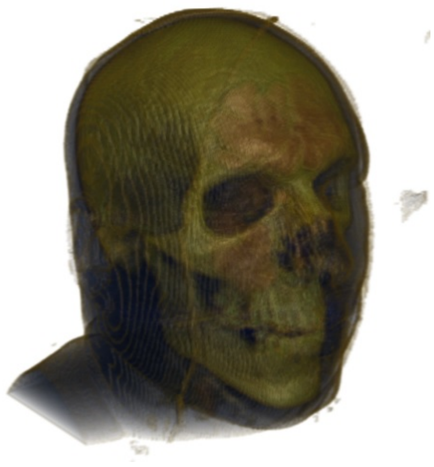


Figure 5.5: Rendering results for the Visual Male Head dataset.

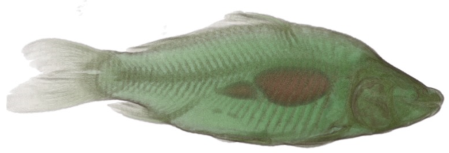


Figure 5.6: Rendering results for the Carp dataset.

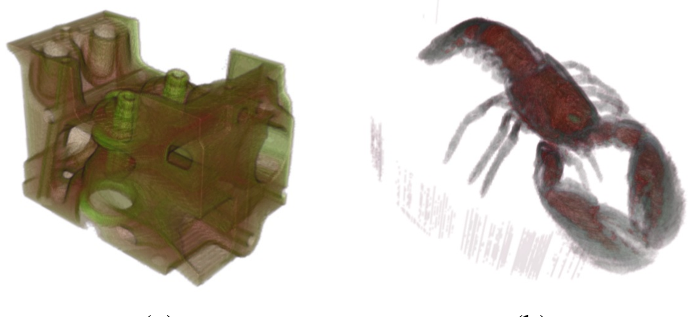

(a) (b)

Figure 5.7: (a) Rendering results for the Engine dataset. (b)Rendering results for the Lobster dataset.

Table 5.1 lists the training sizes and times required for the KN-SVM method (both training and classification). This table reveals a key strength of the image-centric approach. We can see that with the image-centric approach, rendering results can be obtained very quickly without any pre-processing. If we compare the total time with our data-centric approach, we can see that with the data-centric approach, a dataset has to go through the training process at least once, which can take significant amount of time. But with our image-centric approach, a new dataset can be loaded immediately and worked on to achieve satisfactory results quickly and efficiently. We also see the power of the proposed KN-SVM method. From Table 5.1 it can be seen that compared to the whole volume, the number of training voxels is very small. Due to combining the local and global information from the training data, the KN-SVM method can provide satisfactory results even with such a small number of training samples.

Table 5.1: Dataset details and required times (in *seconds*).

Dataset	Size	# Training Samples	Total Time
Foot	256X256X256	401	12.33
Visual Male Head	128X256X256	233	2.27
Carp	256X256X512	299	4.62
Engine	256X256X256	180	1.07
Lobster	301X324X56	97	0.67

5.3.1 Comparison between Data-Centric and Image-Centric Approaches

The data-centric and image-centric approaches are different in principle. As discussed before, volume visualization is mostly a qualitative method. The most important aspect is the user's satisfaction. Hence, we have performed the same user survey as presented in Section 3.6 for the image-centric method. Table 3.2 is reproduced here with the results for our image-centric method appended.

The user survey (Table 5.2) revealed some interesting pros and cons of both the data-centric and image-centric approaches. We can see that in general, the image-centric method is much faster in terms of both training and interaction time. This is because the image-centric method is simpler. The user directly works in the image domain, there is no need to explain any clustering or histogram techniques.

However, we can see that the users rated the image-centric approach much lower than the data-centric one. With the data-centric approach, we can cluster the voxel features and show interesting regions in the cluster domain. Due to high-dimensional clustering, this domain might be able to convey the information available in the volume more concisely. In the image domain, the user typically works on grayscale representation of the volume slices. Although in later stages higher-dimensional features are being used for classification, the training data is selected from the slice representations only. The user might miss some regions that are not apparent until a higher-dimensional clustering is done. Also, the users in general felt that the image-centric approach did not give them the granular level of control that the data-centric approach could provide. In the data-centric approach, they could immediately see which voxels were being selected before the voxel group was defined. But in the image-centric approach, the render was only shown after the voxel group definition is finalized. To summarize, the obvious advantage of image-centric approach is the speed and efficiency in generating a render. But the render might not be as good in quality as can be done in the data-centric approach.

Table 5.2: Times recorded for user training and interaction (in *seconds*) and the ratings (on a scale of 5) by the users for interaction and output quality.

User	System	Training Time	Interaction Time	Interaction Rating	Output Rating
User 1	Data-centric	110	130	4.5	4
	Image-centric	30	15	3.5	3
	Traditional	170	250	3.5	2.5
User 2	Data-centric	90	125	4	5
	Image-centric	25	17	3.5	3
	Traditional	150	190	3	3
User 3	Data-centric	120	130	4	4
	Image-centric	39	21	3.5	3.5
	Traditional	177	200	3.5	3.5
User 4	Data-centric	107	140	4.5	4
	Image-centric	20	20	4	4
	Traditional	150	220	4	3.5
User 5	Data-centric	100	100	5	4
	Image-centric	35	13	4	3
	Traditional	142	198	3	3
User 6	Data-centric	100	130	5	4.5
	Image-centric	40	21	4	3.5
	Traditional	210	340	3.5	3
User 7	Data-centric	140	140	4.5	4.5
	Image-centric	44	29	4	3
	Traditional	220	310	3.5	2.5
User 8	Data-centric	125	105	4	4
	Image-centric	30	31	3.5	3.5
	Traditional	190	350	4	3.5
User 9	Data-centric	170	220	4	5
	Image-centric	50	39	4	3.5
	Traditional	580	610	3	2.5
User 10	Data-centric	150	200	4.5	5
	Image-centric	47	41	4	3
	Traditional	450	570	2.5	2.5
Average	Data-centric	121.2	158	4.4	4.4
	Image-centric	36	24.7	3.8	3.3
	Traditional	243.9	323.8	3.35	2.95

5.4 Summary

In this chapter we have proposed a new image-centric volume visualization approach where the user directly interacts with the data to select interesting structures. Treating the user input as training data, the KN-SVM classifier combined with the concept of color harmonization can generate accurate output showing easily distinguishable structures with aesthetically pleasing colors. Experimental results on several datasets have shown the effectiveness and efficiency of the system. We have also presented a comparative user survey where the advantages and disadvantages of our proposed data-centric and image-centric volume rendering approaches are discussed.

Chapter 6

Information-Assisted Visual Evaluation for Physical Rehabilitation

6.1 Introduction

Until now the thesis has mainly focused on the field of volume rendering and explored the potential of applying novel machine learning techniques to ease the user interaction process. To further explore the power of combining machine learning technologies with intelligent visual interfaces, we investigate the application of information-assisted visualization to the field of physical rehabilitation in this chapter. To our knowledge, this is the first physical rehabilitation method that employs visual elements into the feedback system.

Physical therapy is a popular form of rehabilitation for treatment of patients with different types of medical conditions, such as stroke and Traumatic Brain Injury (TBI) [136]. The therapy sessions are usually conducted in-person with the patient following the instructions of a supervising medical officer.

However, several problems arise from in-person physical therapy:

- the patient needs to find a slot in the typically packed schedule of clinics and hospitals [137];
- in a session with several patients, the continuous monitoring and feedback is generally impractical [138];

- the feedback from a supervisor is only qualitative – there is no way to measure the performance (i.e. recovery rate) of the patient [138].

Hence, there is a need for automated methods that can provide a patient with in-home self-rehabilitation and can evaluate performance without the help of a supervisor. Besides medical rehabilitation, this kind of evaluation process can also be used in social scenarios, such as dance performance evaluation [139] and general physical fitness training.

Camera and marker based motion capture modules have been used in clinics to evaluate the performance of patients [138, 140, 141]. However, these systems need expensive equipment and elaborate setup, none of which are feasible in a home setting [136]. An alternative for home based rehabilitation is to utilize recently released RGB-D devices like the Nintendo Wii or Microsoft Kinect [142, 143]. The Kinect system is particularly useful for such an evaluation platform because of its ability to accurately differentiate particular joint movements. Using a randomized decision forest algorithm, the Kinect can automatically determine the joint centers of the body in real-time [144]. The joint-based representation acquired by Kinect has been tested against conventional motion capture systems and was found to have potential benefits in the clinical setting [145]. Hence we use the Kinect as the RGB-D device of choice in the proposed method.

After extracting the skeletal feature data, the process focuses on time series comparison. We have two sequences (set of features), one from the user and one from the expert. Our target is to use a distance measure to find out how similar the sequences are. Please note that the problem here is different from the more popular gesture recognition problem which aims to classify a gesture into one of many pre-defined classes. We instead want to provide feedback on how the user is performing with respect to the given expert. Toward this end, what we need is an effective distance measure that can be calculated in real-time.

Dynamic Time Warping (DTW) is a popular algorithm to measure the similarity between two different sequences [146]. DTW can measure the distance between sequences of varying speed. The two time series are aligned by local deformation in the time axis through pre-defined step patterns. Due to its flexibility in terms of feature space and alignment rules, DTW is popular in various fields such as gesture recognition [147], speech processing [148], medical imaging [149], and music analysis [150, 151].

However, directly applying DTW in the proposed process would require the two sequences to be available in their complete form [137]. Besides, the DTW algorithm in its original form is not fast enough to provide real-time feedback. Hence, the user would not be able to receive real-time feedback during his/her performance. We propose an Incremental DTW (IDTW) algorithm which can tackle the aforementioned issues of the classic DTW. The IDTW extends the classic DTW in two ways: 1) by calculating the distance score in such a way that an incomplete sequence can be properly compared with a complete sequence [137] and 2) by optimizing the computational time through reusing already calculated components in the DTW algorithm.

The last problem is to convey the feedback to the user. DTW provides us with a single distance measure, which represents the gross similarity score between the user sequence and the expert sequence. However, a single number may not be enough for a user to assess his or her particular limitations. Rather than using a number, the method will be more useful if the information can be communicated visually. This is where IAV comes into play. We have developed a novel way to easily convey performance measures, which we call "information-assisted visual evaluation". The idea is to visualize performance feedback using a skeleton silhouette. The IDTW is calculated separately for each joint. Then, the IDTW scores are mapped to a color table through an exponential function. This color table is used to render the colored skeleton silhouette, where the color of a limb represents the IDTW distance (i.e. how well the user is performing for that limb). In this way, a quick glance at the skeleton silhouette allows the user to easily and meaningfully assess where he/she is going wrong instead of trying to guess it from a single number.

6.2 Related Work

Automated physical rehabilitation methods have received a considerable amount of attention recently. Most of the proposed methods in the literature are based on the popular marker based motion capture approach. A physical therapy evaluation method based on wearable motion sensing units is presented in [138]. The physical exercise movements are divided into template signals. Then a multi-template multi-match DTW algorithm is used to evaluate a particular exercise. The problem with this approach is that the user receives feedback only when the exercise is completed. There is no way to evaluate his/her performance in real-time. Moreover, the correct and incorrect execution templates of the given patient have to be recorded under the supervision of an expert for the method to work properly, since the evaluation stage in this method is essentially a classification problem. As a result, the physical presence of the patient is required.

A shape theoretic framework based on a single marker is proposed in [140]. A marker is placed on the wrist of the patient to gather data on stroke rehabilitation exercises. This work mainly focuses on classification of impaired and unimpaired subjects. Quantitative feedback is only provided as a post processing option.

In [141], the use of a wireless-based system for rehabilitation is explored. A set of wireless nodes equipped with motion sensors is put together to create a wearable device. The patient's movements are then recorded and mapped onto a 3D model. Therapists can then monitor this 3D model remotely. However, automated feedback is not explored in this paper. The final evaluation is still qualitative and requires expert intervention.

In [137], strain sensors are used on a long sleeve shirt to record upper limb rehabilitation exercises. Then an "Open-end" DTW algorithm is used to classify the exercises into several groups such as correct, slow, fast, etc. Truncated input data is used to assess the performance of the proposed DTW algorithm. However, the end result is still only classification of the exercises, and no method of quantitative scoring was explored.

Kinect has also been applied to physical rehabilitation to a limited extent. A Kinect based system to measure finger joint kinematics is presented in [152]. The paper mainly focuses on measurement techniques and sheds some light on the potential of Kinect in home rehabilitation. In [153], a comparison is presented between an optical tracking system and Kinect by evaluating some upper limb motor tasks using a game-based rehabilitation application. In [154], a game based upper limb rehabilitation system is presented. The quantitative evaluation is determined only on the speed of upper limb movement. The quality of movement (i.e. the motion path) and comparisons with expert movements are not explored.

Perhaps the closest to the proposed method is the one in [136], where exercises performed by patients with traumatic brain injuries to that of an expert are compared. The two performances are compared using several measurements, such as cross correlation and DTW. However, the method is limited in the sense that only complete sequences can be matched. There is no option for real-time feedback. Also, no discussion on how to communicate the quantitative measures to the user is presented. The experimental results are mainly limited to classifying different actions.

Our method tries to solve the following two issues that have not been addressed in the aforementioned existing methods:

- Provide real-time feedback to the user on his/her performance. A single rehabilitation exercise session lasts for a few minutes. Only getting a feedback at the end of the exercise may not be helpful. Real-time feedback can help the user to correct his or her mistakes while performing.

- Communicate feedback to the user in a more meaningful and interpretable way. Instead of just providing a number (distance computed through the proposed IDTW algorithm) or a classification result of mere correctness/incorrectness, our method maps the IDTW distances to colors on the virtual skeleton limbs to instantly show where the user is going wrong.

The same fundamentals are equally applicable in other potential fields of application such as dance practice or athletics.

6.3 Proposed Method

Figure 6.1 represents a high-level block diagram of the proposed method. The process flow is as follows:

- The user is presented with a pre-recorded demo of the exercise performed by an expert. The task of the user is to follow the exercise as accurately as possible.

- The user's skeleton data obtained through Kinect is passed through our feature processing stage so that we can compare the two performances.

- The IDTW algorithm compares the user features to the expert features in real-time.

- The output of the IDTW algorithm is projected onto the virtual skeleton silhouette presented to the user using our color mapping technique. As a result, the user can immediately assess his or her quality of performance and try to fix errors accordingly.

The different modules of the proposed framework are explained in detail in the next sections.

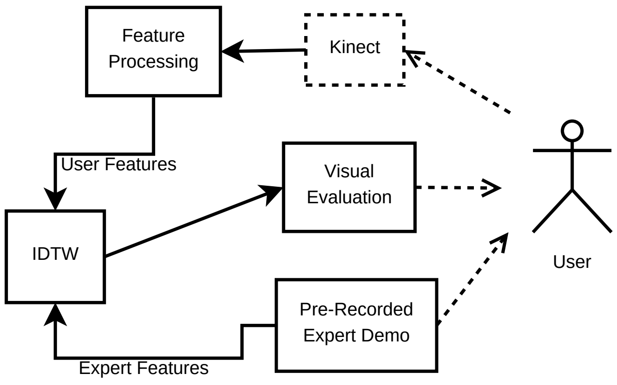


Figure 6.1: Block diagram of the proposed method.

6.3.1 Feature Processing

The Kinect camera [143] can provide a depth image of resolution 640×480 with a speed of 30 frames per second (FPS). The skeletal tracking algorithm built into the Kinect [144] provides a wireframe skeleton of a user that consists of twenty joints which include joints from hands, legs and the torso. Figure 6.2-(a) shows the skeleton silhouette obtained from Kinect. The datastream associated with the skeleton consists of the positions (in world coordinates) of the joints.

In our proposed method, the skeleton data provided by the Kinect is used as input. However, directly using the positional data provided by Kinect may result in an inaccurate or redundant representation of the movements performed by the user/expert because of the following issues:

- Using the absolute positions directly will create some redundancy in the data stream [155]. The joints of the human body move relative to each other (e.g. the movement of the *Wrist Left* joint in Figure 6.2-(a) is relative to the *Elbow Left* joint). A relative representation of the skeleton can help remove the redundancy.

- The user and the expert will most likely be different persons. As a result, using the absolute position will result in some unwanted dissimilarities because the ratios of human limbs vary from person to person. The data needs to be scaled to make it invariant to such differences [156, 157].

- Not all joints are in use for a particular exercise. For instance, for a pectoral stretch, only the upper body is involved, and there is no significant movement of the lower body joints. A filtering process that removes unnecessary joints from processing can facilitate comparison between user and expert features.

The Kinect skeleton can be presented in a hierarchical form, where the joints are connected through a child-parent relation as shown in Figure 6.2-(b). Using this hierarchical representation, we can represent the position of each node in relation to its parent.

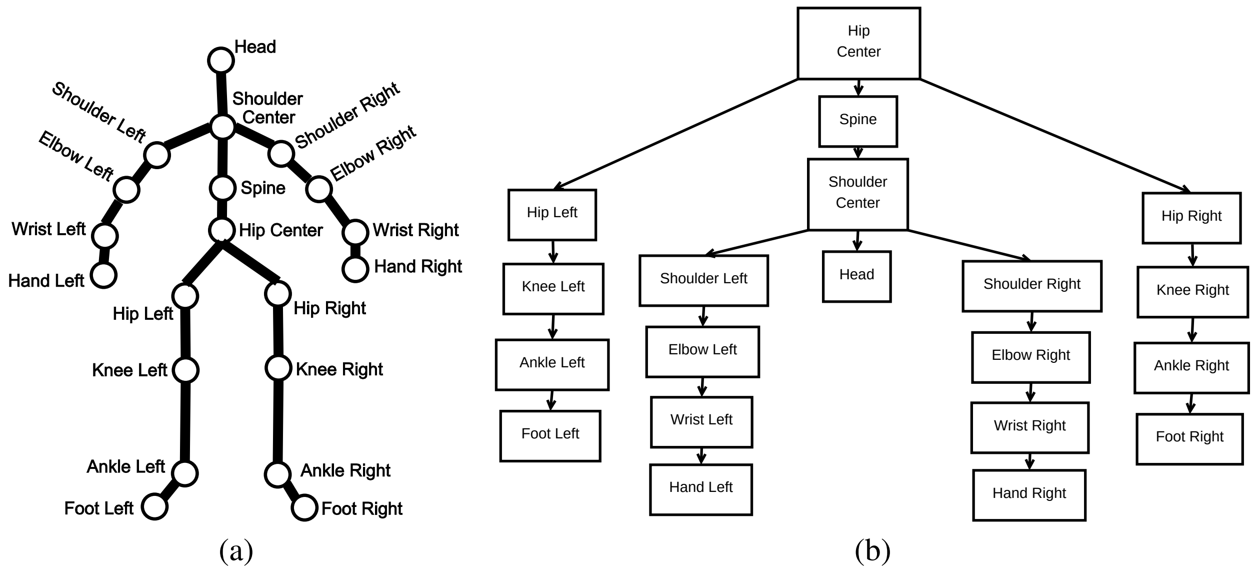


Figure 6.2: a) The Kinect skeleton template. b) Hierarchical representation of the joints.

Let a skeleton be represented by Δ. The skeleton consists of joint coordinates $\Delta = \{\mathbf{r}_l\}, l = 1, \ldots, L$ where $\mathbf{r}_l \in \mathscr{R}^3$ represents the 3D coordinate of the l-th joint, and L is the total number of joints, i.e. $L = 20$.

To represent joint coordinates relative to its parent joint, we subtract the coordinate of the parent joint from every joint. From the hierarchical representation of Figure 6.2-(b) we see that the *Hip Center* joint can be considered as the root of all joints. Hence, we will use the *Hip Center* as the origin of the skeleton, and calculate the relative positions of all the other joints from there.

In the transformed coordinate space, the *Hip Center* joint will be $\mathbf{q}_1 = (0,0,0)$. Let us assume that for each of the rest of the joints in the original coordinate space $\{\mathbf{r}_l\}, l = 2, \ldots, L$, the parent joint is represented by $\mathbf{r}_l^o$. Then, in our method, the transformed coordinate for each joint is calculated as follows [156]:

$$\mathbf{q}_l = \frac{\mathbf{r}_l - \mathbf{r}_l^o}{||\mathbf{r}_l - \mathbf{r}_l^o||_2}, l = 2, \ldots, L. \tag{6.1}$$

where $||.||_2$ is the Euclidean norm. By scaling the transformed coordinate space this way, the normalized features incorporate a certain level of invariance against different body types and limb ratios. As a result, the feedback provided to the user is more robust. We note that since the *Hip Center* joint is considered as the origin, it is not considered for IDTW calculation in our method.

The next issue to tackle is the joint filtering process. As illustrated in Figure 6.3 for normalized Y coordinates, movements of the *Ankle Left* joints are minimal in relation to those of the *Hand Left* for the pectoral stretch exercise performed by an expert (details about the exercises can be found in the experimental results section). Since only the upper body joints are involved for this particular exercise, the other joints can be discarded from further calculation.

To select the relevant joints, we do some pre-processing based on the given expert recording, which serves as the reference. We filter the joints based on the variance of the normalized coordinates in the X,Y and Z directions. More specifically, the following term is calculated for each joint:

$$\beta_l = \sigma_X^2 + \sigma_Y^2 + \sigma_Z^2, l = 2, \ldots, L \tag{6.2}$$

where σ_X^2, σ_Y^2 and σ_Z^2 are the variances in the X,Y and Z directions over the whole recording. A minimum threshold γ is then applied to the values to filter and keep only the relevant joints. For instance, for the pectoral stretch exercise, only the hand joints (*Hand Right*, *Hand Left*, *Wrist Right*, etc.) and the shoulder joints (*Shoulder Right*, *Shoulder Left*, etc.) are relevant. The value of γ can be fixed for different exercises, as the joint selection does not have to be very strict. Most if not all inactive joints can be filtered out with a reasonable value of γ.

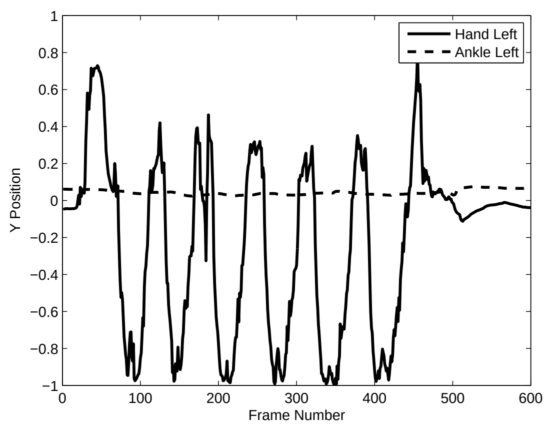


Figure 6.3: Comparison of the Y-positions of *Hand Left* and *Ankle Left* joints for pectoral stretch.

6.3.2 Preliminaries

Before providing the details of DTW and the IDTW algorithm, we define the user and expert sequences first. The sequences consist of a number of Kinect skeleton frames as discussed in the previous section. Let the expert sequence of an exercise be represented by $\mathbf{E} = \{E_1, E2, \ldots E_M\}$ and the user sequence up to the current time by $\mathbf{U} = \{U_1, U2, \ldots U_N\}$. The user sequence consists of N frames that have been acquired till now. Since in our work, the user sequence is incrementally acquired, there could be more incoming frames. As we will see, the proposed IDTW algorithm accommodates this sort of incremental input efficiently.

The *distance* between two frames of these two sequences will be used in the IDTW algorithm. This distance can be defined in terms of the normalized joints of the Kinect skeletons (Equation 6.1). At time instances i and j, the two frames from the above sequences are U_i and E_j, respectively. U_i and E_j can be represented by the normalized joints as $U_i = \{\mathbf{q}_l^{U_i}\}, l = 1, \ldots, K$ and $E_j = \{\mathbf{q}_l^{E_j}\}, l = 1, \ldots, K$. Here, K is the number of joints which survived the joint filtering process for this particular exercise as described in the previous section.

However, if we calculate a single distance between the two frames, the final feedback provided to the user would just be a single number. As discussed before, we use a more informative color mapped skeleton to provide detailed feedback on how the user is performing. We defer the discussion on color mapping to Section 6.3.5. To do the color mapping, the distances have to be calculated on a *per joint* basis rather than as a whole. The distance between the $l-$th joints of frames U_i and E_j can be calculated as:

$$d(U_i, E_j)^l = ||\mathbf{q}_l^{U_i} - \mathbf{q}_l^{E_j}||_2, l = 1, \ldots, K. \tag{6.3}$$

These distances are then used to calculate the IDTW scores, which are also calculated *per joint*. For simplicity, while presenting the IDTW algorithm we omit the index l. However, the reader can assume that the scores are being calculated per joint. We bring the index l back in Section 6.3.5, when we introduce the color mapping technique in detail.

A common practice for optimizing DTW-based algorithms is to use the squared Euclidean distance rather than the Euclidean distance itself [158], as this avoids the additional computation needed to evaluate the square root. Since the distance calculation is part of each DTW iteration, this optimization can significantly speed up the calculation. Using the squared distance changes the absolute value of the DTW output. However, since both functions (square root and squared) are monotonic and concave, the relative measures will still stay the same. The squared distance will be interpretable in the same way as the square root. Hence, we have also opted for this optimization in our implementation.

6.3.3 Dynamic Time Warping

Now we briefly describe the Dynamic Time Warping algorithm. DTW is a template matching algorithm to match a test sequence with a reference sequence (in our case the user sequence with the expert sequence). The objective is to obtain a similarity measure between these two references. One might wonder if simply using a distance measurement like Euclidean distance by aligning the frames of two sequences one by one will suffice. However, there may exist differences in speed between the two sequences, since the user may lag behind the expert and struggle to keep up. If we only restrict the calculation to one-to-one correspondence, the resulting similarity measure will not be accurate (Figure 6.4-(a)). Moreover, the two sequences will most likely be not of the same length. DTW tries to alleviate these problems by compressing or expanding the sequences in the time domain to align them in such a way that minimizes the cumulative difference between the aligned frames (Figure 6.4-(b)).

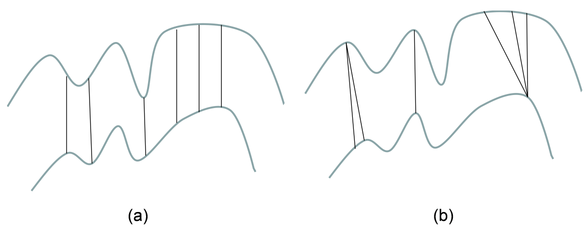


Figure 6.4: (a) Suboptimal similarity measure through one-to-one correspondence. (b) Insertion and deletion by DTW as necessary to compute optimal similarity.

To find the best alignment between two sequences **U** and **E**, an optimal warping path through the grid formed by the frames of the two sequences as shown in Figure 6.5 must be found. Let the warping path be $\mathbf{P} = P_1, P_2, \ldots, P_t, \ldots, P_T$, where $P_t = (i_t, j_t)$. There are many possible paths **P** could take. However, to speed up the computation and keep the path sensible, some constraints are set upon the path:

- Start: The path should start from the origin, i.e., $P_1 = (1,1)$.
- End: The path should cover both sequences completely, i.e., $P_T = (N, M)$.
- Monotonicity: No loop back in time is allowed, i.e., $i_t \geq i_{t-1}$ and $j_t \geq j_{t-1}$.This guarantees that frames are considered sequentially and past frames of the sequences are not repeatedly matched against each other.
- Slope: The slope constraint imposes some step patterns when calculating the next frame on the warping path [159, 160]. A commonly used step pattern is shown in Figure 6.6. These step patterns ensure that the the whole sequence will be considered. It also limits the amount of compression or expansion of a sequence to find the optimal similarity score. Without the step pattern restriction, a very short part of one sequence could be matched with a very long part of the other sequence.
- Warping window: The warping window constraint puts a restriction on how far the path can stray from the diagonal [161]. Essentially, it means that for $P_t = (i_t, j_t)$, $|i_t - j_t| \leq R$. R is called the *Sakoe-Chiba band* [161] (Figure 6.7).

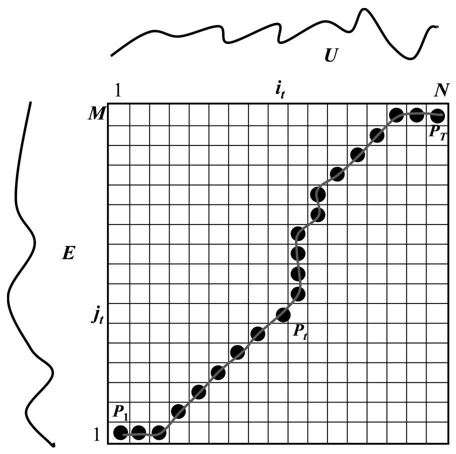


Figure 6.5: Illustration of the warped path on the grid.

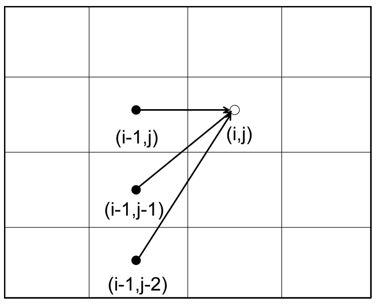


Figure 6.6: Example step pattern restriction when calculating element (i,j) on the warped path.

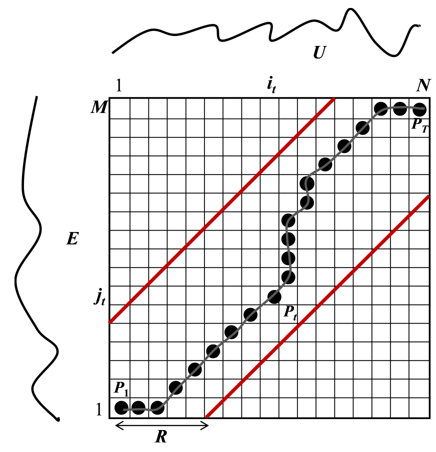


Figure 6.7: Sakoe-Chiba band of width R.

The optimal warping path $\mathbf{P}^*$ is found by minimizing the DTW distance:

$$\mathbf{P}^* = \arg\min_{\mathbf{P}}(\mathbf{D}(\mathbf{U},\mathbf{E})), \tag{6.4}$$

where $\mathbf{D}(\mathbf{U},\mathbf{E})$ is the DTW distance defined as follows:

$$\mathbf{D}(\mathbf{U},\mathbf{E}) = \frac{\sum_{t=1}^{T} d(P_t)\gamma_t}{\sum_{t=1}^{T}\gamma_t}. \tag{6.5}$$

Here, γ_t are the weighting coefficients. The weighting coefficients are needed to normalize the DTW distance with respect to time. Otherwise, short sequences will always be favored over longer sequences in terms of obtaining the minimum similarity measure. $d(P_t)$ is the distance measure between the frames (Equation 6.3). The weighting coefficients are defined along with the chosen step pattern in such a way that they do not depend on the specific route the warping path will take (details will be provided later).

6.3.4 Incremental Dynamic Time Warping

Now, we present the key extensions to the classic DTW that leads to the proposed Incremental DTW (IDTW) algorithm. The main advantages over the classic DTW are twofold:

- *Matching partial sequences*: In our work, the sequence from the user is incrementally growing. The proposed algorithm can match partial sequences with a complete sequence (the expert's) for an accurate similarity score.
- *Optimizing computational time*: For the incrementally growing user sequence, we present strategies to optimize the calculation of the DTW distance by re-using already calculated terms.

Matching Partial Sequences:

The classic DTW algorithm assumes that both of the sequences to be matched are available in full before the start of the algorithm. This is not suitable for our objective of providing real-time feedback to the user. Therefore, the proposed algorithm has to be able to accommodate an partial sequence.

This is achieved through an extension of the existing DTW algorithm [137]. Instead of matching the partial sequence with the full sequence as done in classic DTW, the IDTW algorithm matches it with the best possible starting segment of the reference. This can be achieved by terminating the full sequence at all possible frames, calculating the DTW distances for all of them, and picking the minimum. Let the expert (reference) sequence $\mathbf{E}$ terminated at all possible frames be represented by $\mathbf{E}^j, j = 1,\ldots,M$. In other words, $\mathbf{E}^j$ will be a sequence containing the first j frames of $\mathbf{E}$. Then, the IDTW distance between the user sequence $\mathbf{U}$ and the reference sequence $\mathbf{E}$ can be represented by

$$\mathbf{D}_{IDTW}(\mathbf{U},\mathbf{E}) = \min_{j=1,\ldots,M} \mathbf{D}(\mathbf{U},\mathbf{E}^j), \tag{6.6}$$

where $\mathbf{D}(\mathbf{U},\mathbf{E}^j)$ is defined in Equation 6.5.

Interestingly, there is no need to calculate the DTW distances repeatedly over all possible $\mathbf{E}^j$. It can be calculated efficiently through some minimal changes to how the classic DTW distance is calculated.

In classic DTW, a *cumulative cost matrix* is calculated in order to minimize the similarity score between the two sequences. The cost matrix $\mathbf{G}$ is updated iteratively through the use of a chosen step pattern. In this work, we use the step pattern as depicted in Figure 6.6. DTW is calculated by dynamic programming, where the cost matrix is calculated through recursion [146]. The update formula for the cost matrix can be directly written from the step patterns as:

$$\begin{aligned}\mathbf{G}(1,1) =& d(U_1,E_1),\\ \mathbf{G}(i,j) =& \min(G(i-1,j),G(i-1,j-1),G(i-1,j-2))\\ &+d(U_i,E_j),\end{aligned} \tag{6.7}$$

where $d(U_i,E_j)$ is defined in Equation 6.3 (minus the index l for simplicity).

Once the cost matrix calculation is completed, the classic DTW always considers the value on the top-right corner (i.e. $\mathbf{G}(M,N)$) as the final similarity score. IDTW differs from DTW in this step. Instead of taking the top-right value, IDTW takes the minimum of all the values in the right-most column as depicted in Figure 6.8. Indeed, this value directly corresponds to the value we are looking for in Equation 6.6. This is easily understandable from the nature of the recursion in Equation 6.7 and from the monotonicity constraint described before. For a specific position (i, j) in the cumulative cost matrix $\mathbf{G}$, the values above and right to that position have

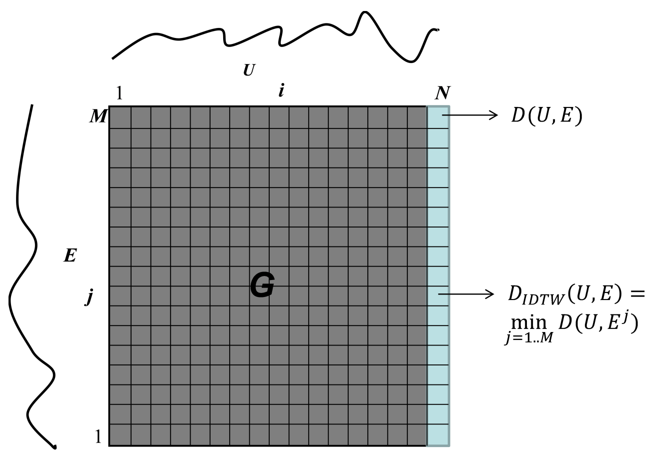


Figure 6.8: Calculation of final cost: DTW vs. IDTW.

no influence. As a result, if the expert sequence were terminated at position j, the final DTW distance value will be the same as that at the (N, j)-th position in the cumulative cost matrix as shown in Figure 6.8.

Finally, we define the weighting coefficients in Equation 6.5. To solve the DTW problem with dynamic programming, we need the recursion defined in Equation 6.7 to be an exact solution of the minimization problem in Equation 6.4. To achieve this, the weighting coefficients should not depend on the specific warping path. In other words, the sum of weighting coefficients in Equation 6.5 should be constant over the specific set of sequences being compared, i.e., $\sum_{t=1}^{T} \gamma_t = C$, where C is a constant. Different step patterns correspond to different values to be used as C [162]. The step pattern defined by Equation 6.7 can be used with C being set to N, i.e., the number of frames in the current user sequence. By calculating the IDTW this way, we can find a better similarity score between the partial sequence from the user and the sequence of the expert.

Optimizing Computation Time:

The worst case computational complexity of the classic DTW is $O(MN)$, where M and N are the length of the sequences as defined before. However, in the proposed framework, the user sequence is incrementally increasing. If we keep repeating the DTW calculation over and over again, the computational complexity will be increasing exponentially.

The complexity can be substantially decreased by observing how the DTW cost is calculated. Due to the monotonicity constraint, the calculations of the cumulative cost matrix **G** never go backward in time. As we can see from Figure 6.9, if the current length of the user

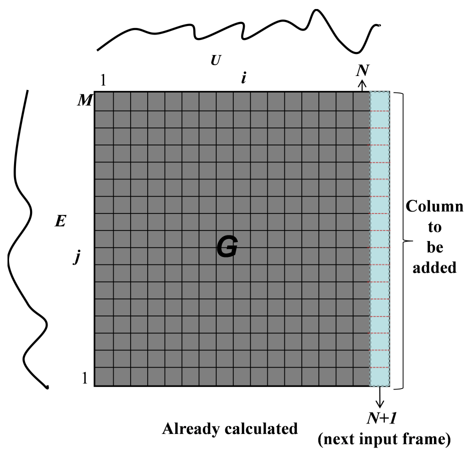


Figure 6.9: Optimization of computational time. Only one column needs to be calculated with each addition of an input frame.

sequence is N, and the $N+1$-th frame comes in, we only need to append one extra column to $\mathbf{G}$. Hence, the complexity of this step can be $O(M)$. With classic DTW, if we repeat the calculation of the whole matrix $\mathbf{G}$, the complexity would have been $O(M*(N+1))$, which is substantially higher. Also, with this optimization, we have a constant number of calculations per frame.

Faster operation of the DTW algorithm can be achieved by applying various techniques such as indexing [163], recursive decimation [164], exploiting similarity and correlation [165]. In the proposed framework, the input frame rate is 30 FPS. As we will see in the experimental results section, our IDTW algorithm is already fast enough to calculate the similarity score in real-time. Hence, further optimization was not a priority for this work. We plan to explore the applicability of these techniques in the future.

Pseudocode of the proposed IDTW algorithm with the aforementioned improvements is presented in Algorithm 1. Here, we see that the next frame in the user sequence is an input to the function. Only one column is appended to the cumulative cost matrix $\mathbf{G}$. This column is then filled according to our recursion rule as described in Equation 6.7. We have also incorporated the Sakoe-Chiba band directly into the loop limit. This saves execution time in comparison to checking the limits inside the loop [158]. Finally, the time normalized minimum value from the newly appended column of the cost matrix $\mathbf{G}$ is returned as the current IDTW distance.

Algorithm 1 The IDTW algorithm

Inputs:
U - The user sequence up to current time (length N).
E - The expert (full) sequence (length M).
G - $M \times N$ cumulative cost matrix up to current time.
V - Next frame in user sequence.
R - Width of Sakoe-Chiba band.
Output:
updated IDTW distance.

```
1: function IDTW(U,E,G,$V$,$R$)
2:     $Q \leftarrow (N+1)$
3:     $U_Q \leftarrow V$
4:     $\mathbf{G}(1\ldots M,Q) \leftarrow array(1\ldots M)$
5:     for $i \leftarrow (\max(1,Q-R),\min(M,Q+R))$ do
6:         $\mathbf{G}(i,Q) \leftarrow \min(\mathbf{G}(i-1,Q),\mathbf{G}(i-1,Q-1),$
               $\mathbf{G}(i-1,Q-2)) + d(U_Q,E_i)$
7:     end for
8:     return $\min(\mathbf{G}(1\ldots M,Q))/Q$
9: end function
```

6.3.5 Information-Assisted Visual Evaluation

The final part of the method is how the performance measurement is communicated to the user. As we discussed before, a sense of immersion from the user's perspective could facilitate in-home rehabilitation. Our automated system aims to provide detailed feedback so that the user can correct him/herself, much like a therapy session under an expert's direct supervision.

If we provide a single IDTW measurement as a score, the user has no way to know where he/she is going wrong. Hence, in the proposed framework we have implemented a *per limb* feedback system, where we loosely refer to the skeletal segment connecting neighboring joints as a limb (Figure 6.2). Rather than calculating the IDTW score as a whole, we calculate separate IDTW scores *per joint*, then convert this to the scores for each limb.

There are 19 limbs in the Kinect skeleton. Although not all of the limbs will be active during a particular exercise, it will still be too many for a user to comprehend if we simply transmit the raw IDTW scores individually. The human perception system is specifically suited to spot visual patterns rather than raw numbers [1]. Hence, we map the IDTW scores to a color table and use the virtual skeleton visualization to convey performance feedback.

To formulate the visual evaluation scheme, first we bring back the index l that was dropped for simplicity after Equation 6.3. Let the IDTW scores for each joint be represented by $\mathbf{D}^{l}_{IDTW}, l = 1, \ldots, K$. Here, K is the number of joints being evaluated for a particular exercise. The distance measure for IDTW is defined in Equation 6.6.

Now, the distance measure for a particular limb can be calculated by simply averaging the two joints that form this limb. Then, the value to be mapped to the color table is calculated as follows:

$$Z = e^{-\nu * (\mathbf{D}^{l_a}_{IDTW} + \mathbf{D}^{l_b}_{IDTW})/2}, \tag{6.8}$$

where l_a and l_b denote the joints that form this particular limb, and ν is a parameter to control the sensitivity of the performance measure. The lower the value of ν, the more sensitive the scoring system will be. In other words, the system will be less tolerant to mistakes by the user if the ν value is lower. In the experiments, ν was set to 20 for all the exercises, which led to satisfactory results throughout.

Finally, the value Z is projected to a color map. The color map used in this work is blue-aqua-green-yellow-red, i.e., the color stays closer to blue if the user is doing well, and shifts towards red as the performance worsens.

Figure 6.10 shows a snapshot of our information-assisted visual evaluation framework when a user performs the leg raise exercise. The limbs involved in this exercise are used in the visual evaluation. Limbs that are not involved (i.e., filtered by the joint filtering process) are colored in white. The effectiveness of the information-assisted visual evaluation system can easily be seen in this screenshot. By a quick look at the virtual skeleton, one can see that most of the active limbs have colors closer to the blue spectrum. Hence, one can conclude that the user is performing well here.

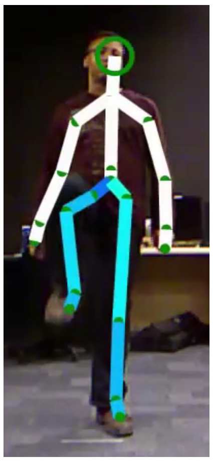


Figure 6.10: The information-assisted visual evaluation scheme.

6.4 Experimental Results

We have evaluated the proposed framework for three different exercises: *leg raise*, *pectoral stretch* and *jumping jack*. The exercises were picked in such a way that different parts of the human body are engaged in different exercises. Hence, they vary in terms of active joints. Leg raise involves the leg joints, while pectoral stretch involves the upper body joints. The whole body is used in jumping jack. Our joint filtering process separates the active joints accordingly by evaluating the recorded expert sequence.

In the joint filtering process, the threshold γ (Equation 6.2) was set to 0.005 for all the exercises. Small changes in the value of this threshold do not significantly affect the overall evaluation method. Hence, this value of γ should be suitable for most cases.

The expert sequences recorded for these exercises are about $15-20$ seconds long (exact running length can be found in Table 6.1). To examine the method's robustness to different users, two other persons besides the person used for recording the expert sequences are tested. Each user was asked to perform each exercise twice. The first time the users were asked to perform it as accurately as possible. This sequence will be called the *good performance*. For the second round, the users were asked to deliberately make some mistakes so that we can test whether our system can provide accurate feedback in real-time. This sequence will be called the *bad performance*.

Figure 6.11 shows a screenshot of the system. The expert sequence is played on the right panel. The user's task is to follow the sequence as correctly as possible. The visual evaluation

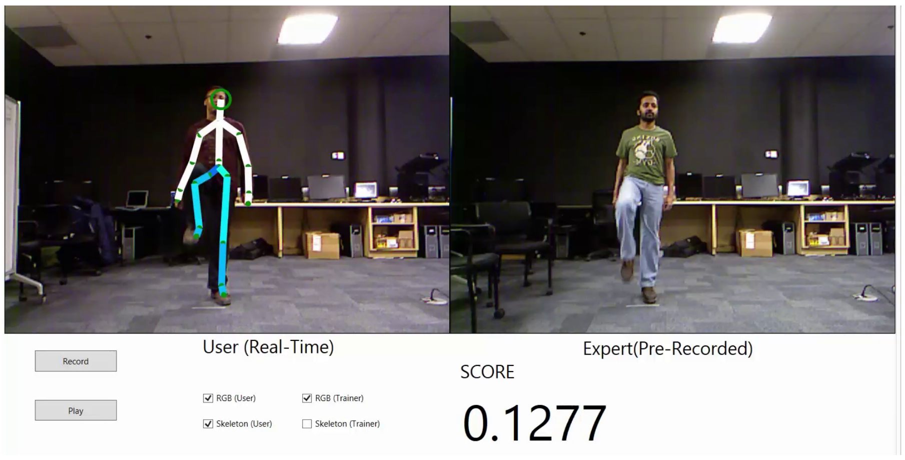


Figure 6.11: Screenshot of the proposed system.

of the user is shown in real-time on the left panel. We have included some screenshots of the system in action later for discussion. However, the best way to comprehend the method is to see the video demonstration, which can be seen at `http://youtu.be/zmIfWmETpiM`.

There is also a component called “Score” as can be seen in Figure 6.11. This score is the sum of IDTW scores for all joints. This score is included for user evaluation, so that the users can qualitatively compare this numerical reporting method to our skeletal visualization method.

The IDTW algorithm was also compared with the classic DTW algorithm in terms of accuracy and computational time. Here, we present some quantitative comparisons between the classic DTW and the IDTW algorithm for all three exercises. For these quantitative results, the expert sequence is compared against one good performance using both the classic DTW algorithm and the IDTW algorithm. The good performance is used to show the accuracy of the IDTW distance function (Equation 6.5) as opposed to the classic DTW distance (Equation 6.5).

Figure 6.12 shows graphical comparisons of the distance scores for classic DTW and IDTW for each of the three exercises. Please note that while plotting these graphs, we have skipped the first 50 frames for all the exercises, because the classic DTW cost is too high for those frames to be accommodated into the graph alongside the IDTW scores.

As seen from the graphs, the IDTW can measure the distance between the two sequences more accurately. Since the sequence being tested is a good performance, we expect the distance measure to be small. The classic DTW fails to provide such a measure and the distances are too high for the majority of the exercise. When both the sequences are available in their full form, the DTW score and the IDTW score are almost identical. This is expected, as with the full sequence the DTW can report the distance accurately. But IDTW excels when the partially available sequence of the user is being compared against the full expert sequence. This feature of the IDTW algorithm is crucial for accurate real-time feedback.

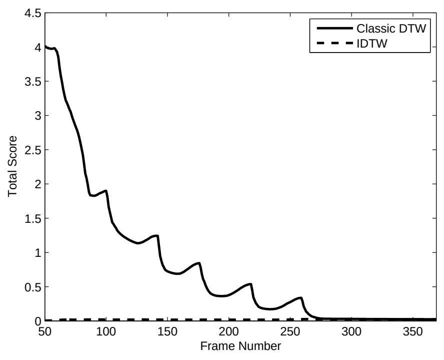
IDTW also comes out as a clear winner when the computational times of the two algorithms are compared. Table 6.1 shows the total computational time for a full comparison of two sequences for all three exercises. The running length of the sequences is also listed. We can see that DTW fails to perform in real-time and hence is not suitable to be used for interactive feedback. Due to the incremental calculation technique in IDTW, the computation is fast enough to be calculated in real-time.

Figure 6.13 illustrates frame-by-frame calculation times. Only the first 250 frames for each exercise are reported to accommodate the graphs properly. As we can see, the classic DTW computational time increases as the frame number increases. This is because the classic DTW is being re-calculated every time. Since the length of the user sequence increases as time progresses, it takes more time to calculate the full DTW per frame. But IDTW makes use of the already calculated cumulative cost matrix and hence only has constant number of calculations per frame. This is evident from the graphs, as the IDTW calculation time stays almost constant throughout the exercises (Figure 6.13).

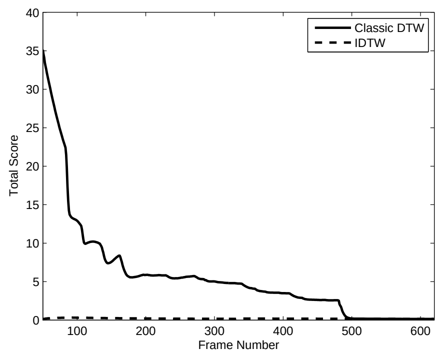
Finally, Figure 6.14 shows a comparison of visual evaluations between good and bad performances for all the exercises. Comparisons for only a single frame are shown here. To see the continuous calculation and visual feedback, the reader is highly encouraged to see the video demo at `http://youtu.be/zmIfWmETpiM`.

Table 6.1: Length of the exercises (in sec.) and total computational times (in sec.) for classic DTW and IDTW.

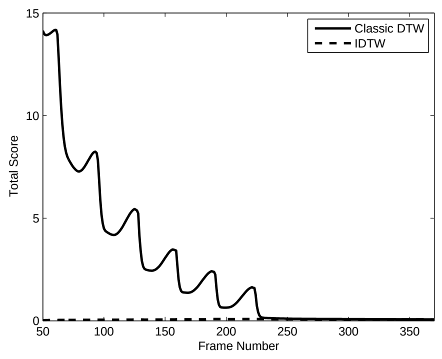
Exercise Name	Length	Total Time (DTW)	Total Time (IDTW)
Leg Raise	13	11.46	0.08
Pectoral Stretch	21.3	72	0.3
Jumping Jack	14.5	27.06	0.19



(a) Leg raise

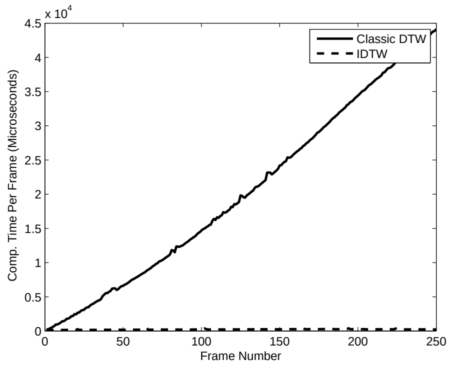


(b) Pectoral stretch

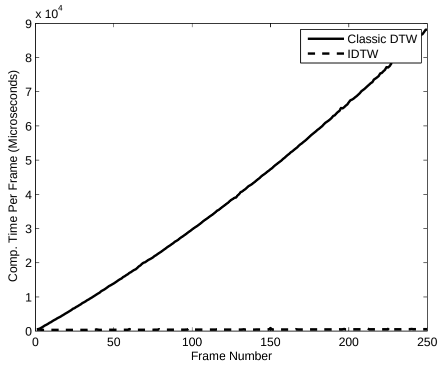


(c) Jumping jack

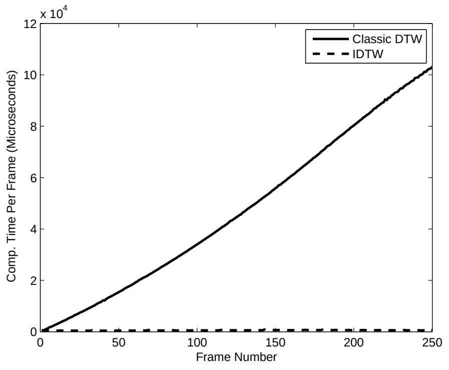
Figure 6.12: Comparison of distance scores between classic DTW and IDTW for different exercises.



(a) Leg raise

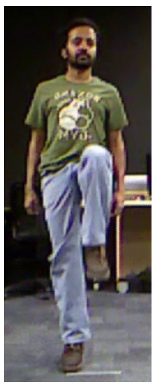


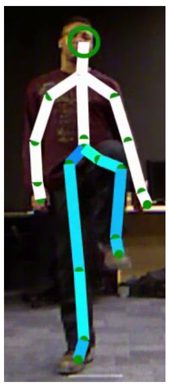
(b) Pectoral stretch

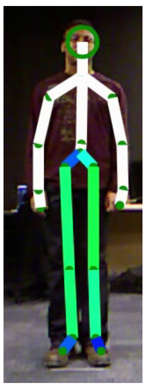


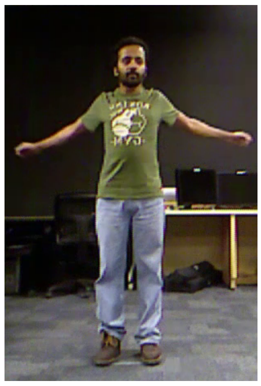
(c) Jumping jack

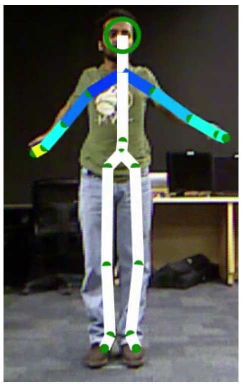
Figure 6.13: Comparison of frame-wise computational costs (in microseconds) between classic DTW and IDTW for different exercises.

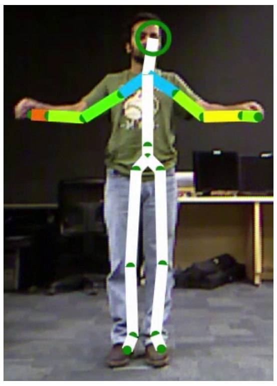

(a) Expert

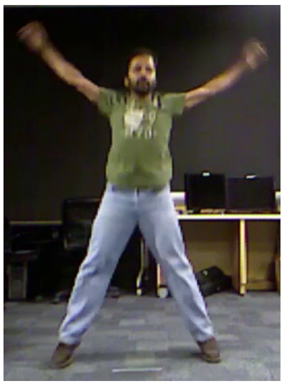

(b) Good performance

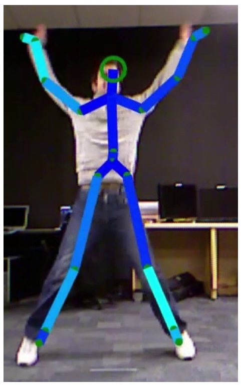

(c) Bad performance


(d) Expert


(e) Good performance


(f) Bad performance


(g) Expert


(h) Good performance

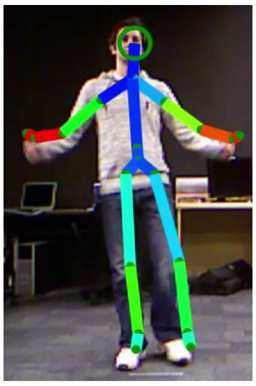

(i) Bad performance

Figure 6.14: Comparison of visual evaluation between good and bad performances for different exercises.

As we can see, the proposed method can report the mistakes by the user in an easily interpretable way. For all the exercises, when the user is performing well, the color of the visual feedback stays close to blue. But when the user is making mistakes, the system can report in detail where the user is going wrong. For instance, in the case of leg raise, the user is not lifting his leg completely. So the color is shifting towards green/yellow. In the case of jumping jack, the user's hand movements are worse than his leg movements. The expert's hands are much further up when compared to the user's. As a result, the limb colors of the hand are shifting towards red while the colors for the leg limbs are shifting towards green/yellow. This form of visual interpretation helps the user quickly correct himself.

Since the total raw score of IDTW was also reported in the framework (see the screenshot in Figure 6.11), we asked the users to evaluate the single number scoring system against our visualization. All the users preferred the visualization over the number-based evaluation. They unequivocally agreed that the sense of immersion with our intuitive visual feedback can be more helpful than a simple score for home-based automated movement evaluation.

6.5 Summary

In this chapter, we propose an information-assisted visual evaluation framework for real-time automated feedback for in-home physical rehabilitation and exercise. Based on normalized features extracted from the Kinect skeleton, we compare the user's feature sequence with a pre-recorded sequence from an expert. The proposed Incremental Dynamic Time Warping (IDTW) algorithm can calculate the distance score between the partial user sequence and the complete expert sequence more accurately than the classic DTW. The proposed algorithm also utilizes the repetitive nature of classic DTW to significantly reduce computation time. Rather than providing a single number as performance feedback, our proposed framework presents a colored virtual skeleton visualization, where different colors of the limbs indicate the level of performance of the user. This form of visual feedback helps the user to easily interpret his/her mistakes and correct it immediately. Experimental results on three different exercises involving different users show that the proposed method can calculate the distance score for a user performance accurately and efficiently. The visual evaluation framework provides a sense of immersion, which is necessary for an in-home rehabilitation system to be effective.

Chapter 7

Conclusion

7.1 Thesis Summary

The purpose of this thesis was to explore the use of machine learning techniques to facilitate the process of information-assisted visualization. Machine learning techniques can be used to design intelligent user interfaces, where required information can be conveyed to the user. By providing the user with concise information and efficient control, the semantic gap between the user and the system can be reduced.

To that end, the thesis primarily focused on volume rendering, which is an important visualization process for many application domains, especially medical data. The present day volume rendering techniques involve complex and tedious process of editing the transfer function that generates the color and opacity values to obtain rendering that shows intended information. To ease this task, we have approached the TF design process from two different angles, the data-centric method and the image-centric method.

In our proposed data-centric approach, the user interacts with a simple color-coded Self-Organizing Map to select interesting regions in the cluster domain. By using harmonic colors and intelligent opacity assignment techniques, we immediately generate the rendering. With our proposed method, the user can visualize the intended results without having to manipulate complex widgets. Experimental results on some benchmark datasets showed the effectiveness of our proposed approach.

For the image-centric approach, first we proposed two novel supervised classification methods, the Kernel Nonparametric Support Vector Machine (KN-SVM) classifier and the Covariance-guided One-Class Support Vector Machine (COSVM) classifier. Both these classifiers are derived from the same philosophy of combining global information from discriminant-based classifiers like LDA with the local information provided by the support vectors in an SVM-based classifier. We have provided detailed experimental results for these classifiers, comparing them with the state-of-the-art.

Our KN-SVM classifier is then used to design the image-centric approach towards volume rendering. In this approach, the user directly works in the image domain i.e. with the volume slices themselves. The user selects training data from the grayscale representation of the slices. This data is used to classify the whole volume. The intelligent coloring method used for our data-centric approach is also used here to generate rendering results quickly and efficiently.

We have used the same datasets for both our data-centric and image-centric approaches so that the results can be qualitatively compared. Moreover, we have provided a comprehensive user survey for these two methods, where the proposed approaches are compared with a traditional visualization tool. These results not only strengthen the cases for both our proposed methods, but also sheds light on the pros and cons of data-centric versus image-centric approaches.

Finally, we have explored the potential of information-assisted visualization in a new field, medical rehabilitation. To our knowledge, visualization has not been used as a tool for communication in this domain. We propose an information-assisted visual evaluation framework for in-home physical rehabilitation. The user can follow a pre-recorded performance by an expert in the proposed system. The instant feedback is obtained by using our novel Incremental Dynamic Time Warping algorithm, which can compare the two sequences (the user's and the expert's) in real-time. The feedback is communicated to the user through a visual form of a skeleton silhouette, where different colors on the limbs show different levels of performance. The obtained results were encouraging, and proves yet again that visual communication can be more effective than mere numbers.

The three proposed systems successfully demonstrate that through careful integration of machine learning techniques with intelligent user interfaces, the semantic gap between the user and the system can be reduced. A simpler interface does not necessarily have to sacrifice the level of control a user can have over the output. An ideal system should be able to strike the perfect balance between automation and customization.

7.2 Future Work

A natural extension to the proposed volume visualization methods would be to incorporate the principles of Knowledge-Assisted Visualization (KAV) [166]. The argument in favor of KAV is that incorporating the user knowledge into the system can improve the results gradually. For the field of volume rendering, the user knowledge can come in the form of user interaction. In both the proposed methods, the user interacts with the GUI to select regions in the volume data. Once the user is satisfied with a particular dataset, these selected regions could be saved and used in future for similar datasets. Although the principle is simple, the steps required to achieve this can be difficult. The main difficulty lies in computing the similarity of volume datasets. Due to their complex natures and inherent noise characteristics, the similarity will not be easy to calculate. However, if an effective method to compute the similarity and segment the volume into similar regions can be devised, the incorporation of KAV into our proposed methods can make them even more efficient and faster.

For our medical rehabilitation system, the focus was mainly on the IDTW algorithm and the information-assisted visual evaluation scheme. More robust features instead of joint positions can be used, which may result in a better evaluation. Also, the visual evaluation scheme is universal and works the same way for all users. A customized evaluation scheme can be more helpful for the user to assess his or her performance. This customized scheme can be derived from an aggregation of user's previous performances, so that his or her particular weaknesses are emphasized properly.

Publications

Journals:

1. **N.M. Khan**, S. Lin, L. Guan and B. Guo. Visual Performance Evaluation of Human Movements Using Incremental Dynamic Time Warping. *IEEE Transactions on Multimedia*. [Submitted]

2. **N.M. Khan**, R. Ksantini, I.S. Ahmad and L. Guan. Covariance-guided One Class Support Vector Machine. *Pattern Recognition*, vol. 47 no. 6 pp. 2165-2177, 2014.

3. **N.M. Khan**, M. Kyan and L. Guan. Intuitive Volume Exploration through Spherical Self-Organizing Map and Color Harmonization. *Neurocomputing*. [Accepted][**Invited Article**]

4. **N.M. Khan**, R. Ksantini, I.S. Ahmad and L. Guan. SN-SVM: A Sparse Nonparametric Support Vector Machine Classifier. *Signal, Image and Video Processing*. [In Press]

Conferences and Workshops:

1. **N.M. Khan**, S. Lin, L. Guan and B. Guo. A Visual Evaluation Framework for In-Home Physical Rehabilitation. *IEEE International Symposium on Multimedia*. [Submitted]

2. **N.M. Khan**, R. Ksantini and L. Guan. Volume Visualization Using Sparse Nonparametric Support Vector Machines and Harmonic Colors. *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2014)*, pp. 6607-6611, 2014.

3. **N.M. Khan**, M. Kyan and L. Guan. ImmerVol: An Immersive Volume Visualization System. *IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA 2014)*, pp. 24-29, 2014.

4. **N.M. Khan**, R. Ksantini, I.S. Ahmad and L. Guan. Incorporating Covariance Information in One Class Support Vector Classification. *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2013)*, pp. 3552-3556, 2013.

5. **N.M. Khan**, M. Kyan and L. Guan. Intuitive Volume Exploration through Spherical Self-Organizing Map. In *Proceedings of the 9th Workshop on Self-Organizing Map (WSOM 2012)*, pp. 75-84, 2012.

6. **N.M. Khan**, R. Ksantini, I.S. Ahmad and L. Guan. A Sparse Support Vector Machine Classifier with Nonparametric Discriminants. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN 2012)*, pp. 330-338, 2012.

Patents:

1. S. Krishna, X. Nan, **N.M. Khan**, N. Dong, J.R.T Bond, L. Guan, M. Kyan, Y. He and E. Biggs. Systems and Methods for a Shared Mixed Reality Experience. *PCT Patent No. PCT/IB2014/061672*, 2014.

Appendix A

Traditional TF Editor

The traditional TF editor used for the user surveys presented in Chapter 3 and Chapter 5 was an off-the-shelf volume visualization software named VolView [60]. A screenshot of the software can be seen in Figure A.1. The TF editor module of the software is shown separately in Figure A.2. As can be seen, a one-dimensional histogram based on the voxel intensity values is presented to the user. The user clicks through the histogram to define points on the TF that will generate the final output. The selected points on the histogram decide how much weight should be assigned to different parts of the histogram. The intensity values that fall between two selected points are weighted based on linear interpolation.

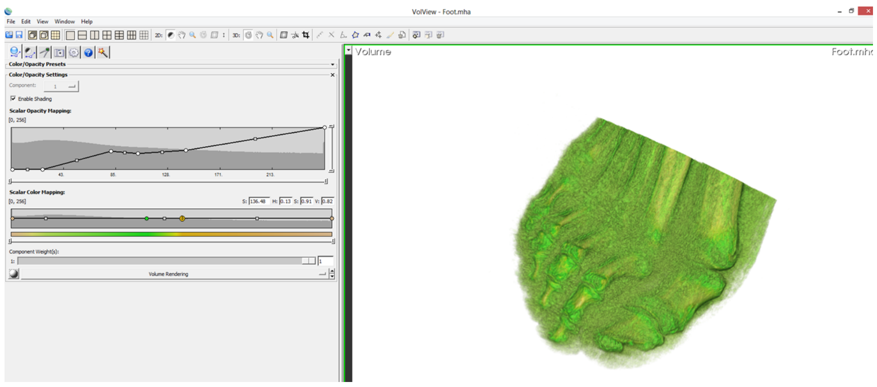


Figure A.1: Screenshot of VolView.

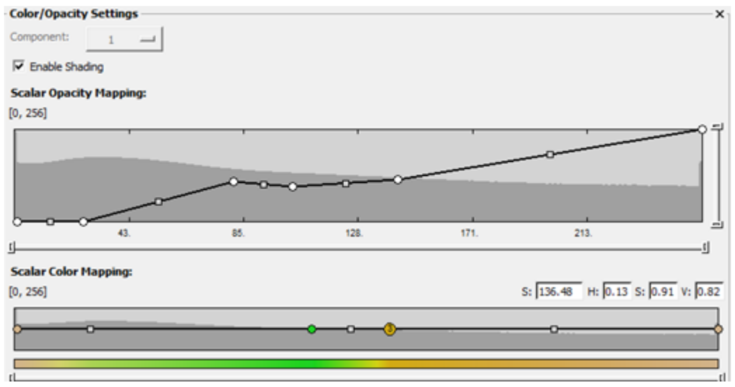


Figure A.2: Close-up of the TF editor module of VolView.

References

[1] W. Aigner, S. Miksch, H. Schumann, and C. Tominski, *Visualization of Time-Oriented Data*, 1st ed. Springer Verlag, 2011.

[2] J. Beyer, M. Hadwiger, A. Al-Awami, W.-K. Jeong, N. Kasthuri, J. Lichtman, and H. Pfister, "Exploring the connectome: Petascale volume visualization of microscopy data streams," *IEEE Computer Graphics and Applications*, vol. 33, no. 4, pp. 50–61, Aug 2013.

[3] T. Schultz and G. Kindlmann, "Open-box spectral clustering: Applications to medical image analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, pp. 2100–2108, 2013.

[4] L. Zhou and C. Hansen, "Interactive rendering and efficient querying for large multivariate seismic volumes on consumer level pcs," in *IEEE Symposium on Large Scale Data Analysis and Visualization*. IEEE Press, 2013, pp. 117–118.

[5] J.-W. Ahn, C. Plaisant, and B. Shneiderman, "A task taxonomy for network evolution analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, pp. 365–376, 2014.

[6] N. Ferreira, J. Poco, H. Vo, J. Freire, and C. Silva, "Visual exploration of big spatio-temporal urban data: A study of new york city taxi trips," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, pp. 2149–2158, 2013.

[7] T. Pham, R. Metoyer, K. Bezrukova, and C. Spell, "Special section on visual analytics: Visualization of cluster structure and separation in multivariate mixed data: A case study of diversity faultlines in work teams," *Computers and Graphics*, vol. 38, pp. 117–130, Feb 2014.

[8] F. Post, G. Nielson, and G.-P. Bonneau, Eds., *Data Visualization: The State of the Art*. Kluwer, 2003.

[9] N. Ezquerra, L.de-Braal, E. Garcia, C. Cooke, and E. Krawczynska, “Interactive, knowledge-guided visualization of 3D medical imagery,” *Future Generation Computer Systems*, vol. 15, no. 1, pp. 59–73, 1999.

[10] J. Gillespie, A. Gholkar, and I. Isherwood, *Three-dimensional computer tomographic reformations: assessment of clinical efficacy*. Boca Raton, FL, USA: CRC Press, Inc., 1991, pp. 103–144.

[11] J. Wijk and J. van, “Bridging the gaps,” *IEEE Computer Graphics and Applications*, vol. 26, no. 6, pp. 6–9, November 2006.

[12] D. Duke, K. Brodlie, D. Duce, and I. Herman, “Do you see what i mean?” *IEEE Computer Graphics and Applications*, vol. 25, no. 3, pp. 6–9, May 2005.

[13] H. Childs, B. Geveci, W. Schroeder, J. Meredith, K. Moreland, C. Sewell, T. Kuhlen, and E. Bethel, “Research challenges for visualization software,” *IEEE Computer*, vol. 46, no. 5, pp. 34–42, May 2013.

[14] M. Chen, D. Ebert, H. Hagen, R. Laramee, R. van Liere, K.-L. Ma, W. Ribarsky, G. Scheuermann, and D. Silverh, “Data, information, and knowledge in visualization,” *IEEE Computer Graphics and Applications*, vol. 29, no. 1, pp. 12–19, January 2009.

[15] M. Levoy, “Display of surfaces from volume data,” *IEEE Computer Graphics and Applications*, vol. 8, no. 3, pp. 29–37, May 1988.

[16] G. Kindlmann and J. Durkin, “Semi-automatic generation of transfer functions for direct volume rendering,” in *Proceedings of the 1998 IEEE symposium on Volume visualization*. IEEE, 1998, pp. 79–86.

[17] T. Kohonen, Ed., *Self-organizing maps*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1997.

[18] K. Fukunaga, *Introduction to Statistical Pattern Recognition, second ed.* Academic Press, 2000.

[19] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[20] Y. Liu, C. Lisle, and J. Collins, "Quick2Insight: A user-friendly framework for interactive rendering of biological image volumes," in *IEEE Symposium on Biological Data Visualization*. IEEE Press, 2011, pp. 1–8.

[21] B. McCormick, "Visualization in scientific computing," *SIGBIO Newsletter*, vol. 10, no. 1, pp. 15–21, March 1988.

[22] M. Hadwigers, J. Kniss, C. Rezk-salama, D. Weiskopf, and K. Engel, *Real-time Volume Graphics*. Natick, MA, USA: A. K. Peters, Ltd., 2006.

[23] J. Kniss, G. Kindlmann, and C. Hansen, "Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets," in *IEEE symposium on Volume visualization*. IEEE Press, 2001, pp. 255–262.

[24] P. Sereda, A. Bartroli, I. Serlie, and F. Gerritsen, "Visualization of boundaries in volumetric data sets using LH histograms," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, pp. 208–218, May 2006.

[25] R. Maciejewski, Y. Jang, I. Woo, H. Janicke, K. Gaither, and D. Ebert, "Abstracting attribute space for transfer function exploration and design," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, pp. 94–107, January 2013.

[26] C. Correa and K. Ma, "Size-based transfer functions: a new volume exploration technique," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, pp. 1380–1387, May 2008.

[27] M. Hadwiger, L. Fritz, C. Rezk-Salama, T. Hollt, and G. G. T. Pabel, "Interactive volume exploration for feature detection and quantification in industrial ct data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, pp. 1507–1514, May 2008.

[28] S. Wesarg, M. Kirschner, and M. Khan, "2D histogram based volume visualization: combining intensity and size of anatomical structures," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, pp. 655–666, May 2010.

[29] C. Lundstrom, P. Ljung, and A. Ynnerman, “Local histograms for design of transfer functions in direct volume rendering,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, pp. 1570–1579, May 2006.

[30] C. Lundstrom, A. Ynnerman, P. Ljung, A. Persson, and H. Knutsson, “The α-histogram: using spatial coherence to enhance histograms and transfer function design,” in *IEEE symposium on visualization*. IEEE Press, 2006, pp. 227–234.

[31] S. Roettger, M. Bauer, and M. Stamminger, “Spatialized transfer functions,” in *IEEE/Eurographics symposium on visualization*. IEEE Press, 2005, pp. 271–278.

[32] A. Tappenbeck, B. Preim, , and V. Dicken, “Distance-based transfer function design: Specification methods and applications,” in *SimVis*. IEEE Press, 2006, pp. 259–274.

[33] M. Selver, M. Alper, and C. Guzeli, “Semiautomatic transfer function initialization for abdominal visualization using self-generating hierarchical radial basis function networks,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, pp. 395–409, May 2009.

[34] P. Sereda, A. Vilanova, and F. Gerritsen, “Automating transfer function design for volume rendering using hierarchical clustering of material boundaries,” in *IEEE Symposium on Visualization*. IEEE Press, 2006, pp. 243–250.

[35] R. Maciejewski, I. Wu, W. Chen, , and D. Ebert, “Structuring feature space: A non-parametric method for volumetric transfer function generation,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, pp. 1473–1480, 2009.

[36] F. Zhou, Y. Zhao, and K. Ma, “Parallel mean shift for interactive volume segmentation,” in *International Conference on Machine Learning in Medical Imaging*. IEEE Press, 2010, pp. 67–75.

[37] B. Nguyen, W.-L. Tay, C.-K. Chui, and S.-H. Ong, “A clustering-based system to automate transfer function design for medical image visualization,” *The Visual Computer*, vol. 28, pp. 181–191, 2012.

[38] F. Pinto and C. M. D. S. Freitas, “Design of multi-dimensional transfer functions using dimensional reduction,” in *Eurographics Symposium on Visualization*. IEEE Press, 2007, pp. 131–138.

[39] S. S. Fang, T. B. Tom, and M. Tuceryan, “Image-based transfer function design for data exploration in volume visualization,” in *Conference on Visualization*, ser. VIS ’98. IEEE Press, 1998, pp. 319–326.

[40] J. Marks, B. Andalman, P. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber, “Design galleries: a general approach to setting parameters for computer graphics and animation,” in *Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 1997, pp. 389–400.

[41] H. Guo, H. Xiao, and X. Yuan, “Multi-dimensional transfer function design based on flexible dimension projection embedded in parallel coordinates,” in *IEEE Pacific Visualization Symposium*. IEEE Press, 2011, pp. 19–26.

[42] F.-Y. Tzeng, “Intelligent system-assisted user interfaces for volume visualization,” Ph.D. dissertation, University of California, 2006.

[43] B. Liu, B. Wnsche, and T. Ropinski, “Visualization by example - a constructive visual component-based interface for direct volume rendering,” in *International Conference on Computer Graphics Theory and Applications*. ScitePress, 2010, pp. 254–259.

[44] J. Itten, *The Art of Color: The Subjective Experience and Objective Rationale of Color*. Van Nostrand Reinhold Company, 1969.

[45] J. Zhou and M. Takatsuka, “Automatic transfer function generation using contour tree controlled residue flow model and color harmonics,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, pp. 1481–1488, November 2009.

[46] Y. Wang, W. Chen, J. Zhang, T. Dong, G. Shan, and X. Chi, “Efficient volume exploration using the gaussian mixture model,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, pp. 1560–1573, 2011.

[47] S. Haykin, *Neural Networks: A Comprehensive Foundation*. New York: MacMillan, 1999.

[48] D. Cohen-Or, O. Sorkine, R. Gal, T. Leyvand, and Y.-Q. Xu, “Color harmonization,” *ACM Transactions on Graphics*, vol. 25, pp. 624–630, 2006.

[49] M. Tokumaru, N. Muranaka, and S. Imanishi, "Color design support system considering color harmony," in *IEEE International Conference on Fuzzy Systems*. IEEE Press, 2002, pp. 378–383.

[50] A. Sangole and A. Leontitsis, "Spherical self-organizing feature map: An introductory review," *International Journal of Bifurcation and Chaos*, vol. 16, pp. 3195–3206, 2006.

[51] A. Sangole, "Data-driven modeling using spherical self-organizing maps," Ph.D. dissertation, Department of Mechanical and Materials Engineering, The University of Western Ontario, 2002.

[52] H. Tokutaka, M. Ohkita, Y. Hai, K. Fujimura, and M. Oyabu, "Classification using topologically preserving spherical self-organizing maps," in *Workshop on Self-Organizing Maps*. LNCS 6731, Springer-Verlag, 2011, pp. 308–317.

[53] H. Ritter, *Self-organizing maps in non-Euclidean spaces*. Springer, 1999.

[54] Y. Wu and M. Takatsuka, "The geodesic self-organizing map and its error analysis," in *Australasian conference on Computer Science*. Australian Computer Society, Inc., 2005, pp. 343–351.

[55] J. Hladuvka, A. Konig, and E. Groller, "Exploiting Eigenvalues of the Hessian matrix for volume decimation," in *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*. Vaclav Skala, 2001, pp. 124–129.

[56] A. Krishnamurthy, S. Ahalt, D. Melton, and P. Chen, "Neural networks for vector quantization of speech and images," *IEEE Journal on Selected Areas in Communications*, vol. 8, pp. 1449–1457, 1990.

[57] W. Schroeder, K. Martin, and B. Lorensen, Eds., *The Visualization Toolkit*. New York, NY, USA: Kitware, 2006.

[58] S. Roettger, "The volume library," 2012, Ohm Hochschule Nurnberg. [Online]. Available: {http://www9.informatik.uni-erlangen.de/External/vollib/}

[59] W. K. Pratt, Ed., *Digital Image Processing*. New York, NY, USA: John Wiley and Sons, 2007.

[60] VolView, “Kitware^{TM},” 2012.

[61] Y. Wu, H. Qu, K.-K. Chung, and M.-Y. Chan, “Quantitative effectiveness measures for direct volume rendered images,” in *IEEE Pacific Visualization Symposium*. IEEE Press, 2007, pp. 1–8.

[62] J. Giesen, K. Mueller, E. Schuberth, L. Wang, and P. Zolliker, “Conjoint analysis to measure the perceived quality in volume rendering,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1664–1671, 2007.

[63] V. Walimbe, V. Zagrodsky, S. Raja, W. A. Jaber, F. P. DiFilippo, M. J. Garcia, R. C. Brunken, J. D. Thomas, and R. Shekhar, “Mutual information-based multimodality registration of cardiac ultrasound and SPECT images: a preliminary investigation,” *The International Journal of Cardiovascular Imaging*, vol. 19, pp. 483–494, 2003.

[64] R. Likert, “A technique for the measurement of attitudes,” *Archives of Psychology*, vol. 22, no. 140, pp. 1–55, 1932.

[65] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti, “Surround-screen projection-based virtual reality: the design and implementation of the cave,” in *20th annual Conference on Computer Graphics and Interactive Techniques*. ACM, 1993, pp. 135–142.

[66] B. Laha, K. Sensharma, J. D. Schiffbauer, , and D. A. Bowman, “Effects of immersion on visual analysis of volume data,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, pp. 597–606, April 2012.

[67] A. Bierbaum, C. Just, P. Hartling, K. Meinert, A. Baker, and C. Cruz-Neira, “VR Juggler: A virtual platform for virtual reality application development,” in *Proceedings of the Virtual Reality 2001 Conference (VR’01)*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 89–96.

[68] D. Burns and R. Osfield, “OpenSceneGraph. A: introduction, B: examples and applications,” in *Proceedings of the IEEE Virtual Reality 2004*. IEEE Computer Society, 2004, pp. 265–.

[69] M. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*. Cambridge, UK: Cambridge University Press, 2000.

[70] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. Mullers, “Fisher discriminant analysis with kernels,” *Neural Networks for Signal Processing*, pp. 41 –48, aug. 1999.

[71] V. Vapnik, *Statistical Learning Theory*. New York, USA: John Wiley & Sons, 1998.

[72] T. Xiong and V. Cherkassky, “A combined SVM and LDA approach for classification,” in *International Joint Conference on Neural Networks*, 2005, pp. 1455–1459.

[73] G. Baudat and F. Anouar, “Generalized discriminant analysis using a kernel approach,” *Neural Computation*, vol. 12, no. 10, pp. 2385–2404, 2000.

[74] G. Dai and Y. Qian, “Kernel generalized nonlinear discriminant analysis algorithm for pattern recognition,” in *International Conference on Image Processing*, vol. 4, oct. 2004, pp. 2697 – 2700.

[75] D. You, O. C. Hamsici, and A. M. Martinez, “Kernel optimization in discriminant analysis,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 3, pp. 631–638, 2011.

[76] C. Lee and D. Landgrebe, “Feature extraction based on decision boundaries,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 4, pp. 388–400, 1993.

[77] L. Huang, Y. Ma, Y. Ijiri, S. Lao, M. Kawade, and Y. Zhao, “An adaptive nonparametric discriminant analysis method and its application to face recognition,” in *Asian Conference on Computer Vision*, 2007, pp. 680 – 689.

[78] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik, “Feature selection for SVMs,” *Advances in Neural Information Processing Systems 13*, pp. 668 –674, 2000.

[79] P. Shivaswamy and T. Jebara, “Maximum relative margin and data-dependent regularization,” *Journal of Machine Learning Research*, vol. 11, pp. 747–788, 2010.

[80] ——, “Elliposoidal kernel machines,” in *Proceedings of the Artificial Intelligence and Statistics*, 2007.

[81] S. Keerthi, “Efficient tuning of SVM hyperparameters using radius/margin bound and iterative algorithms,” *IEEE Transactions on Neural Networks*, vol. 13, no. 5, pp. 1225–1229, Sep 2002.

[82] B. Zhang, X. Chen, S. Shan, and W. Gao, “Nonlinear face recognition based on maximum average margin criterion,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 554 – 559.

[83] K. Crammer, M. Dredze, and F. Pereira, “Exact convex confidence-weighted learning,” *Advances in Neural Information Processing Systems 21*, 2009.

[84] L. Rosasco, E. D. Vito, A. Caponnetto, M. Piana, and A. Verri, “Are loss functions all the same?” *Neural Computation*, vol. 16, no. 5, pp. 1063–1076, 2004.

[85] B. Scholkopf and A. Smola, *Learning With Kernels-Support Vector Machines, Regularization, Optimization and Beyond.* Cambridge: MA: MIT Press, 2001.

[86] R. Duda, P. E. Hart, and D. Stork, *Pattern Classification (2nd Edition).* Wiley-Interscience, 2000.

[87] K. Fukunaga and L. Hostetler, “The estimation of the gradient of a density function, with applications in pattern recognition,” *IEEE Transactions on Information Theory*, vol. 21, no. 1, pp. 32–40, Jan 1975.

[88] M. E. Tipping, “Sparse Bayesian learning and the relevance vector machine,” *Journal of Machine Learning Research*, vol. 1, pp. 211–244, 2001.

[89] R. Horn and R. Charles, *Matrix Analysis.* Cambridge University Press, 1990.

[90] H. Kuhn and A. Tucker, “Nonlinear programming,” *In Proceedings of the 2nd Berkeley Symposium*, pp. 481–492, 1950.

[91] MATLAB Bioinformatics Toolbox, “The MathWorks$^{\text{TM}}$,” 2014.

[92] G. Golub and C. V. Loan, *Matrix Computations (third edition).* The John Hopkins University Press, 1996.

[93] T. Coleman and Y. Li, “A reflective newton method for minimizing a quadratic function subject to bounds on some of the variables,” *SIAM Journal on Optimization*, vol. 6, no. 4, pp. 1040–1058, 1996.

[94] I. Rish, “An empirical study of the naive Bayes classifier,” in *IJCAI Workshop on Empirical Methods in AI*, 2001.

[95] A. D. Belegundu and T. R. Chandrupatla, *Optimization Concepts and Applications in Engineering*. Prentice Hall, 1999.

[96] G. Ratsch, T. Onoda, and K. Muller, “Soft margins for adaboost,” *Machine Learning*, vol. 42, no. 3, pp. 287–320, 2000.

[97] A. Asuncion and D. Newman, “UCI machine learning repository,” 2007, university of California, Irvine, School of Information and Computer Sciences.

[98] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes: The Art of Scientific Computing, third ed.* Cambridge University Press, 2007.

[99] P. Belhumeur, J. Hespanha, and D. Kriegman, “Eigenfaces vs. Fisherfaces: recognition using class specific linear projection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 711–720, 1997.

[100] C. Elkan, “Naive bayes learning,” Department of Computer Science and Engineering, University of California, San Diego, California, Tech. Rep. CS97-557, September 1997.

[101] P. Juszczak, “Learning to recognise. a study on one-class classiifcation and active learning,” Ph.D. dissertation, Delft University of Technology, Delft, The Netherlands, 2006.

[102] C. Varun, B. Arindam, , and K. Vipin, “Anomaly detection: A survey,” *ACM Computing Surveys*, vol. 41, pp. 1–58, 2009.

[103] D. Tax and R. Duin, “Support vector data description,” *Machine Learning*, vol. 54, pp. 45–66, 2004.

[104] P. Hayton, B. Schölkopf, L. Tarassenko, and P. Anuzis, “Support vector novelty detection applied to jet engine vibration spectra,” *Advances in Neural Information Processing Systems*, vol. 13, pp. 946–952, 2000.

[105] L. Guo, “Tumor detection in MR images using one-class immune feature weighted SVMs,” *IEEE Transactions on Magnetics*, vol. 16, no. 10, pp. 3849–3852, 2011.

[106] S. Sharma, C. Bellinger, and N. Japkowicz, “Clustering based one-class classification for compliance verification of the comprehensive nuclear-test-ban treaty,” in *Canadian Conference on Artificial Intelligence*. IEEE Press, 2012, pp. 181–193.

[107] Y. Hwanjo, “PEBL: Web page classification without negative examples,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, pp. 70–81, 2004.

[108] Y.-S. Chen and Y.-M. Chen, “Combining incremental hidden Markov model and adaboost algorithm for anomaly intrusion detection,” in *ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics*. ACM, 2009, pp. 3–9.

[109] D. Tax, “One-class classification,” Ph.D. dissertation, Delft University of Technology, Delft, The Netherlands, 2001.

[110] J. A. Hartigan and M. A. Wong, “A K-means clustering algorithm,” *Applied Statistics*, vol. 28, pp. 100–108, 1979.

[111] V. Barnett and T. Lewis, *Outliers in Statistical Data 2nd Ed.* Cambridge, UK: John Wiley and Sons Ltd, 1978.

[112] C. Bishop, “Novelty detection and neural network validation,” *IEEE Proceedings on Vision, Image and Signal Processing. Special Issue on Applications of Neural Networks*, vol. 141, no. 4, pp. 217–222, 1994.

[113] L. Tarassenko, P. Hayton, and M. Brady, “Novelty detection for the identification of masses in mammograms,” in *IEEE International Conference on Artificial Neural Networks*. IEEE Press, Oct. 1995, pp. 442–447.

[114] L. Parra, G. Deco, and S. Miesbach, “Statistical independence and novelty detection with information preserving nonlinear maps,” *Neural Computation*, vol. 8, pp. 260–269, 1996.

[115] D. Tax and K.-R. Mller, “Feature extraction for one-class classification,” in *Artificial Neural Networks and Neural Information Processing*. IEEE Press, 2003, pp. 342–349.

[116] S. Geman, E. Bienenstock, , and R. Doursat, “Neural networks and the bias/variance dilemma,” *Neural Computation*, vol. 4, pp. 1–58, 1992.

[117] A. Rakotomamonjy, Y. Grandvalet, S. Canu, and V. Guigue, “Svm and kernel methods toolbox,” 2007, http://mloss.org/software/view/33/.

[118] S. Saitoh, *Theory of Reproducing Kernels and Its Applications*. Harlow, England: Longman Scientific and Technical, 1998.

[119] C. Michelli, “Interpolation of scattered data: Distance matrices and conditionally positive definite functions,” *Constructive Approximation*, vol. 2, pp. 11–22, 1986.

[120] R. Horn and R. Charles, *Matrix Analysis*. Cambridge University Press, 1990.

[121] C.-C. Chang and C.-J. Lin, “LIBSVM: A library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 1–27, 2011.

[122] G. Cabral and A. I. de Oliveira, “A novel one-class classification method based on feature analysis and prototype reduction,” in *IEEE International Conference on System, Man and Cybernetics*. IEEE Press, Oct. 2011, pp. 983–988.

[123] T. Fawcett, “ROC graphs: notes and practical considerations for data mining researchers,” HP Laboratories, Tech. Rep. HPL-20034, 01 2003.

[124] J. Hanley and B. McNeil, “A method of comparing the areas under receiver operating characteristic curves derived from the same cases,” *Radiology*, vol. 148, no. 3, pp. 839–843, 1983.

[125] E. Parzen, “On estimation of a probability density function and mode,” *Annals of Mathematical Statistics*, vol. 33, pp. 1065–1076, 1962.

[126] R. Duin, “On the choice of the smoothing parameters for parzen estimators of probability density functions,” *IEEE Transactions on Computers*, vol. C-25, no. 11, pp. 1175–1179, 1976.

[127] S. Harmeling, G. Dornhege, D. Tax, F. Meinecke, , and K.-R. Mueller, “From outliers to prototypes: Ordering data,” *Neurocomputing*, vol. 69, no. 13-15, pp. 1608–1618, 2006.

[128] I. W. Tsang, J. T. Kwok, and S. Li, "Learning the kernel in Mahalanobis one-class support vector machines," in *International Joint Conference on Neural Networks*. INNS, 2006, pp. 1169–1175.

[129] D. Tax, "DDtools, the data description toolbox for Matlab," May 2014, version 1.9.1.

[130] Y. Jiang and Z.-H. Zhou, "Editing training data for k-NN classifiers with neural network ensemble," in *International Symposium on Neural Networks*. IEEE Press, 2004, pp. 356–361.

[131] Y. Muto and Y. Hamamoto, "Improvement of the Parzen classifier in small training sample size situations," *Intelligent Data Analysis*, vol. 5, no. 6, pp. 477–490, 2001.

[132] R. Duin, P. Juszczak, P. Paclik, E. Pekalska, D. de Ridder, D. Tax, and S. Verzakov, "PR-Tools 4.1, a matlab toolbox for pattern recognition," 2007, http://prtools.org.

[133] T. Zhou, D. Tao, and X. Wu, "NESVM: A fast gradient method for support vector machines," in *International Conference on Data Mining*. IEEE Press, 2010, pp. 679–688.

[134] E. Hazan, T. Koren, and N. Srebro, "Beating SGD: Learning SVMs in sublinear time," *Advances in Neural Information Processing Systems*, vol. 24, pp. 1233–1241, 2011.

[135] C. Studholme, D. Hill, and D. Hawkes, "An overlap invariant entropy measure for 3D medical image alignment," *Pattern Recognition*, vol. 32, pp. 71–86, 1999.

[136] J. Venugopalan, C. Cheng, T. Stokes, and M. Wang, "Kinect-based rehabilitation system for patients with traumatic brain injury," in *Annual International Conference of the IEEE EMBS*. IEEE Press, 2013, pp. 4625–4628.

[137] P. Tormene, T. Giorgino, S. Quaglini, and M. Stefanelli, "Matching incomplete time series with dynamic time warping: an algorithm and an application to post-stroke rehabilitation," *Artificial Intelligence in Medicine*, vol. 45, pp. 11–34, 2009.

[138] A. Yurtman and B. Barshan, "Detection and evaluation of physical therapy exercises by dynamic time warping using wearable motion sensor units," *Information Science and Systems*, vol. 264, pp. 305–314, 2013.

[139] D. Alexiadis, P. Kelly, T. Boubakeur, and M. Moussa, "Evaluating a dancer's performance using Kinect-based skeleton tracking," in *Proceedings of the ACM International Conference on Multimedia*. ACM, 2011, pp. 659–662.

[140] V. Venkataraman, P. Turaga, N. Lehrer, M. Baran, T. Rikakis, and S. Wolf, "Attractor-shape for dynamical analysis of human movement: Applications in stroke rehabilitation and action recognition," in *IEEEE Conference on Computer Vision and Pattern Recognition Workshops*. IEEE Press, 2013, pp. 514–520.

[141] P. Daponte, L. Vito, and C. Sementa, "A wireless-based home rehabilitation system for monitoring 3D movements," in *IEEEE International Symposium on Medical Measurements and Applications*. IEEE Press, 2013, pp. 282–287.

[142] "Nintendo Wii," 2014, Retrieved February 2014. [Online]. Available: {https://www.nintendo.com/wii}

[143] "Microsoft Kinect," 2014, Retrieved February 2014. [Online]. Available: {http://www.xbox.com/en-CA/Kinect}

[144] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *IEEE Conference on Computer Vision and Pattern Recognition*. Washington, DC, USA: IEEE Computer Society, 2011, pp. 1297–1304.

[145] R. Clark, Y.-H. Pua, K. Fortin, C. Ritchie, K. Webster, L. Denehy, and A. Bryant, "Validity of the Microsoft Kinect for assessment of postural control," *Gait and Posture*, vol. 36, pp. 372–377, 2012.

[146] D. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *AAAI Workshop on Knowledge Discovery in Databases*. ACM, 1994, pp. 359–370.

[147] T. Arici, S. Celebi, A. Aydin, and T. Temiz, "Robust gesture recognition using feature pre-processing and weighted dynamic time warping," *Multimedia Tools and Applications*, July 2013.

[148] A. Lee, Y. Zhang, and J. Glass, "Mispronunciation detection via dynamic time warping on deep belief network-based posteriorgrams," in *Proceedings of the IEEE International*

Conference on Acoustics, Speech and Signal Processing. IEEE Press, 2013, pp. 8227–8231.

[149] Y. Zhu, Y.-C. Kim, M. Proctor, S. Narayanan, and K. Nayak, "Dynamic 3-D visualization of vocal tract shaping during speech," *IEEE Transactions on Medical Imaging*, vol. 32, no. 5, pp. 838–848, 2013.

[150] M. Muller, D. Ellis, A. Klapuri, and G. Richard, "Signal processing for music analysis," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 6, pp. 1088–1110, 2011.

[151] S. Dixon, "An online time-warping algorithm for tracking musical performances," in *International Joint Conference on Artificial Intelligence*. IEEE Press, 2005, pp. 1727–1728.

[152] C. Metcalf, R. Robinson, A. Malpass, T. Bogle, T. Dell, C. Harris, and S. Demain, "Markerless motion capture and measurement of hand kinematics: Validation and application to home-based upper limb rehabilitation," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 8, pp. 2184–2192, August 2013.

[153] C.-Y. Chang, B. Lange, M. Zhang, and S. Koenig, "Towards pervasive physical rehabilitation using Microsoft Kinect," in *International Conference on pervasive Computing Technologies for Healthcare*. IEEE Press, 2012, pp. 159–162.

[154] I. Pastor, H. Hayes, and S. Bamberg, "A feasibility study of an upper limb rehabilitation system using kinect and computer games," in *Annual International Conference of the IEEE EMBS*. IEEE Press, 2012, pp. 1286–1289.

[155] M. Raptis, D. Kirovski, and H. Hoppe, "Real-time classification of dance gestures from skeleton animation," in *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Computer Animation*. ACM, 2011, pp. 147–156.

[156] W. Shen, K. Deng, T. Leyvand, B. Guo, and Z. Tu, "Exemplar-based human action pose correction," *IEEE Transactions on Cybernetics*, vol. PP, no. 99, 2013.

[157] W. Shen, K. Deng, X. Bai, T. Leyvand, B. Guo, and Z. Tu, "Exemplar-based human action pose correction and tagging," in *IEEEE Conference on Computer Vision and Pattern Recognition*. IEEE Press, 2012, pp. 1784–1791.

[158] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, “Searching and mining trillions of time series subsequences under dynamic time warping,” in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2012, pp. 262–270.

[159] H. Shako and S. Chiba, “Dynamic programming algorithm optimization for spoken word recognition,” *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[160] L. Rabiner and B.-H. Juang, Eds., *Fundamentals of speech recognition*. New York, NY, USA: Prentice Hall Inc., 1993.

[161] Y. Sakurai, C. Faloutsos, and M. Yamamuro, “Stream monitoring under the time warping distance,” in *IEEE International Conference on Data Engineering*. IEEE Press, 2007, pp. 1046–1055.

[162] L. Rabiner, A. Rosenberg, and S. Levinson, “Considerations in dynamic time warping algorithm for discrene world recognition,” *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 26, no. 6, pp. 575–582, 1978.

[163] E. Keogh, “Exact indexing by dynamic time warping,” in *International Conference on Very Large Databases*. ACM, 2002, pp. 406–417.

[164] S. Salvador and P. Chan, “Toward accurate dynamic time warping in linear time and space,” *Intelligent Data Analysis*, vol. 11, no. 5, pp. 561–580, 2007.

[165] G. Al-Naymat, S. Chawla, and J. Taheri, “SparseDTW: a novel approach to speed up dynamic time warping,” in *Australian Data Mining Conference*. ACM, 2009, pp. 117–127.

[166] M. Chen and H. Hagen, “Guest editors’ introduction: Knowledge-assisted visualization,” *IEEE Computer Graphics and Applications*, vol. 30, no. 1, pp. 15–16, 2010.