The Dissertation Committee for Youngin Shin

certifies that this is the approved version of the following dissertation:

Parametric Kernels for Structured Data Analysis

Committee:

Donald S. Fussell, Supervisor

Okan Arikan

Joydeep Ghosh

Kristine L. Grauman

Peter Stone

Parametric Kernels for Structured Data Analysis

by

Youngin Shin, B.E.; M.S.C.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

May 2008

To my loving wife Hyun-young Kim and

my coolest friend and transporter, Segway HT

Acknowledgments

I would like to thank numerous people for their support over the past years during my graduate study without which this thesis would not have been possible. I am fortunate to have had valuable experience to collaborate with them.

My utmost gratitude goes to my supervisor Dr. Donald Fussell for his support and advice throughout the years. Dr. Fussell has been not only a great research adviser but an excellent mentor of my life full of humor and inspiration. He provided me with a great degree of freedom to explore the research areas, find the topic, and pursue my ideas. Throughout this, his sharp insight kept me from going to the dead-ends. It has been always great pleasure working with him.

I am deeply indebted to Dr. Peter Stone for providing me with encouragement and insightful comments on my work. Days I spent with Dr. Stone and the colleagues of his UT Austin Villa robot research lab has been my precious moments. Especially, participating the Robocup at Home competition 2007 under his guidance was one of the most exciting and challenging experience in my graduate school career. I cannot thank enough for what he has given me. I also would like to thank Dr. Kristen Grauman for sharing her extensive knowledge and deep

understanding in machine learning, artificial intelligence, and computer vision.

Large portion of the work in this thesis would not have been possible without the help of my friends and the members of UT Austin Villa robot research lab. Hyuk Cho helped me so much with valuable comments on my research and answering to my tedious questions on machine learning and data mining; Juhyun Lee, Tekin Mericli, Brad Knox, and many others worked hard to gather valuable data for face recognition.

I also would like to thank my former supervisor Dr. Chandrajit Bajaj for providing me with a valuable opportunity to start my academic career at this university as well as support and advice during my earlier years. My gratitude goes also to Dr. Zijian Zheng at Microsoft Data Mining for Search Quality team for being unusually supportive to me to finish this thesis.

Last, but not the least, I am grateful to my dad Dr. Tae-gon Shin and my mom Hyang-seon Shim for their love and support. Also, my heartfelt gratitude goes to my loving wife Hyun-young Kim for being my dearest friend and supporter. She has been and will be the everlasting source of energy, pleasure, and happiness in my life. Finally, I thank my coolest friend and transporter, Segway HT.

YOUNGIN SHIN

The University of Texas at Austin

May 2008

Parametric Kernels for Structured Data Analysis

Publication No. _____

Youngin Shin, Ph.D.

The University of Texas at Austin, 2008

Supervisor: Donald S. Fussell

Structured representation of input physical patterns as a set of local features has been useful for a veriety of robotics and human computer interaction (HCI) applications. It enables a stable understanding of the variable inputs. However, this representation does not fit the conventional machine learning algorithms and distance metrics because they assume vector inputs. To learn from input patterns with variable structure is thus challenging. To address this problem, I propose a general and systematic method to design distance metrics between structured inputs that can be used in conventional learning algorithms. Based on the observation of the stability in the geometric distributions of local features over the physical patterns across similar inputs, this is done combining the local similarities and the conformity of the geometric relationship between local features. The produced distance metrics, called “parametric kernels”, are positive semi-definite and require almost

linear time to compute. To demonstrate the general applicability and the efficacy of this approach, I designed and applied parametric kernels to handwritten character recognition, on-line face recognition, and object detection from laser range finder sensor data. Parametric kernels achieve recognition rates competitive to state-of-the-art approaches in these tasks.

Table of Contents

List of Figures

Chapter 1

Introduction

Humans constantly interact with the physical world for various purposes. Continuous streams of physical patterns from one's surroundings are sensed and analyzed to extract information that is useful for either taking appropriate actions or understanding the world. They approach a destination, avoid obstacles or dangers, recognize faces, group similar objects into categories, localize, or exchange information with peers using language or images.

The main goal of robotics or human-computer interaction (HCI) research is to develop techniques that enable a computer to autonomously reason about and interact with its environment much like humans do. In doing so, we hope to achieve a deeper understanding of our own human abilities, and to learn to build more powerful and intelligent robots or interactive computer systems.

In this research, I focus on the problem of *computing the similarity between physical patterns*. This is a fundamental problem that underlies a variety of robotics

and HCI learning tasks, such as recognition, grouping, estimation, or searching, where the input examples are physical patterns obtained mainly by taking measurements or capturing human interaction. For instance, mobile robots use laser range finders to measure distances to surrounding objects or webcams to capture visual information. They make decisions on its next action to achieve the goal such as navigation or object avoidance. Human authentication systems require biological signatures such as fingerprints, voice, face images, or handwritten characters, as the input.

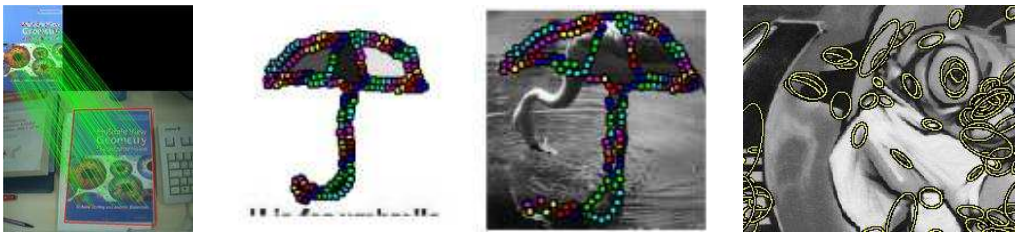
Humans are remarkably accurate and efficient at telling how similar two physical patterns are. This ability is extremely robust against all sorts of change in the physical conditions. The first step to make computers do this is to represent the physical patterns in a form that computers can access. Mostly, this is done by digitally sampling the *physical input space*. For instance, on-line handwritten characters consist of sequences of 2D points that are obtained by capturing pen or mouse points in a 2D plane. Likewise, music and speech consist of sequences of quantized amplitude values that are obtained by recording using microphones, shape contours are represented as sequences of 2D points around the edges of objects in images, and 2D arrays of pixels have been used to represent images. Unfortunately, changes in physical conditions such as pose or illumination against which humans can reliably analyze physical patterns often may result in varying inputs that are extremely difficult for computers to learn.

To cope with this difficulty, people have tried to find features that are robust against such variations from the input patterns. Consider, for instance, contours of

objects that are represented as sets of 2D points. Contours of the same object may vary in size, orientation, or position, or are composed of different numbers of points. But, they are considered contours of the same object by humans because they contain parts that are relatively invariant against those variations. In *shape context*, a histogram of normalized relative distances to other contour points is computed at each contour point [11]. Shape context is invariant to changes in size or position.

Object recognition or category learning in computer vision research requires the ability to find similar parts of similar or identical objects across different images. Scale invariant feature transform (SIFT) [34], geometric blur [12], or the Harris-Affine transform [38] extract an unordered set of local feature vectors from a varying number of characteristic or salient regions in an image, typically identified by the use of an interest operator. Alternatively, features can be extracted from a predefined set of points. For instance, the set of geodesic distances between the fiducial points of human faces, such as nostrils, eyes, or mouse tips, is an effective feature for 3D face recognition [24]. Such feature vectors are shown to be stable against transformations such as pose or illumination variation, or noise such as clutter, occlusions, or partial variation. With such representations, we can compute the similarity between two images by, for instance, *matching* the local features that are closest in the feature space [23].

Usually the number of local features extracted varies across different input patterns, and therefore the representations are not vectors of a fixed dimension. In the machine learning literature, inputs that are not vectors are referred to as *structured*. Sets of local features of varying cardinality are thus structured inputs. An



(a) SIFT [34] (b) Geometric Blur [12] (c) Harris-Affine [38]

Figure 1.1: Examples of Sets of Local Features

interesting and useful property that underlies such representations, specifically for physical patterns, is the *relationship* of local features. This is of particular interest because it can provide a very useful means to compute the similarity between input physical patterns. A major goal of this research is to find effective ways to take advantage of this useful information. Methods for doing this are not well studied in the literature to date.

To illustrate the types of relationship between local features that I deal with, and to motivate our approach, consider the problem of online handwritten character recognition. Each handwritten character consists of a set of 2D points of varying cardinality. By applying existing techniques to compute the similarity between two sets of vectors, we may compute the similarity between the two handwritten characters. However, in doing so, we neglect the order information underlying the set of points. This is of particular importance since it contains the key information that enables the recovery of strokes, which are critical when we compute the similarity between two handwritten characters. Without the notion of strokes, it is harder for computers to distinguish some sets of points, e.g. 'c', 'e', or 'o'. One simple approach to represent this order information is, for instance, to append the

sequence number i to the coordinates of the i-th input point (x_i, y_i) as an additional dimension. This transforms the sequence of points in a handwritten character into an unordered set of local features (x_i, y_i, i). By applying the same technique to compute the similarity between two sets of vectors, we can compute the similarity between the two handwritten characters, without neglecting the order information. Thus the specific type of relationship in this example is the input order of the data points.

As another example, consider two sets of SIFT features computed from various locations within two different input images. For each SIFT feature in one set, we may find the nearest neighbour from the other set and compute the numerical distance between them. By aggregating the distances computed, we can evaluate the similarity bewteen the two input images. However, there is a possibility that two matched nearest neighbours in the feature space are not extracted from nearby positions in the input images. In the context of certain application domains, e.g. face recognition, this may indicate a *bad* match in the sense that it may result in an undesirable increase in the similarity between face images of two different people. One simple approach is to append to each of the SIFT features ϕ_i the coordinates of the location (x_i, y_i) within the image where each SIFT feature is located. This transforms the set of SIFT features computed from an image into a set of local features (ϕ_i, x_i, y_i). When applying techniques to compute the similarity between two sets of vectors, we may additionally rule out or penalize such bad matches. In this manner, we can compute the similarity between the two images, without neglecting the geometric relationship between the local features. The specific type of relation-

ship in this example is the geometric distribution of the SIFT feature positions over the input images.

This research deals with particular types of structured data where the information about such additional relationships between the local features is quite useful or critical. In this thesis, such types of structured data are said to be *geometric* or *geometrically structured*. Also, such a relationship is referred to as the *geometric structure* of local features. I search for a novel technique to represent such relationships between the local features in a systematic manner and combine this representation data with the local features in the form of a similarity metric. It can then be used directly in a conventional machine learning framework.

1.1 Learning Algorithms

Once the representation of features is determined, appropriate learning algorithms must be chosen. Two important criteria must be taken into account in this process. First, a learning algorithm must be able to take the chosen representation as input. Second, the similarity measure between any two such inputs must be efficient to compute.

Learning algorithms for structured inputs can be roughly categorized into two groups, i.e. *generative* and *discriminative*. Generative methods such as hidden Markov models (HMM) provide a principled way of handling data with variable structures and treating missing information. They usually require certain underlying probabilistic models for the process of data generation. The drawback is that

the accuracy of generative models greatly depends on the accuracy of such probabilistic models. However, such models are very difficult to identify in general because we often have no knowledge of the data generation process for complex physical patterns. Discriminative methods such as neural nets (NN) or support vector machines (SVM) instead learn decision boundaries. They can find complex and flexible decision boundaries using kernel functions, and it is not necessary to build any probabilistic models for the data generation. However, many conventional discriminative learning techniques assume inputs are represented as fixed dimensional vectors and do not allow for the direct application of structured data. There have also been hybrid approaches that take advantage of both generative and discriminative algorithms to yield improved solutions over methods based on an individual method, but the drawbacks from both methods can also be inherited [28, 39, 56]. Moreover, these methods require two training steps, one for learning the probabilistic model and the other for learning the decision boundaries.

In general, discriminative learning methods have empirically demonstrated superior classification results to those of generative methods. Among the well known discriminative learning methods, support vector machine (SVM) has recently drawn strong interest as it has shown state of the art performance on a variety of learning tasks [49,57]. The ability to learn non-linear classifiers in a linear framework using kernel functions is one of the powerful features of SVM. In particular, the principle of structural risk minimization of SVM guarantees the minimization of the generalization error. A large volume of research has been devoted to the application of SVM to a broad set of problem domains and data representations to

take advantage of its excellent performance.

Traditional kernel methods such as SVM assume that input data is represented as fixed dimensional feature vectors, which are compared using generic kernel functions, e.g. linear kernels or radial basis functions (RBF). In practice, what kernel functions really evaluate is often the similarity between two input vectors. Most methods that use generic kernel functions for SVM somewhat arbitrarily choose them. Considering that the type of kernel has a direct impact on the classification performance and the learning complexity, determining a suitable kernel function given a problem domain is an open problem that is challenging.

Typical approaches to applying such traditional kernel techniques to structured inputs involve transforming the inputs into feature vectors in a single feature space of some dimension and then classifying these using generic kernel functions. The problem with this approach is that it may result in the loss of useful structural information between local features because fixed dimensional feature vectors are often too restricted a format to represent irregular structures. For instance, in [54], a fixed size feature vector is computed from strokes, and an SVM classifier is used. The dimensions of the features include the mean coordinates and second order statistics such as median, variance, minimum and maximum distances, area, etc. However, information about the shape of the strokes is lost in this process. In [32], Extended R-squared (ER^2) is proposed as the similarity measure for sequences. Though it uses the coordinates of points directly as features, they can only operate on point sequences of a fixed-length so as to compute the proposed similarity metric.

In addition, this may not be general enough to be used in other similar problems. For example, a common first step for the detection of an orange ball from camera images is to segment out orange regions from the background. Segmented orange regions are not vectors but rather variable-sized blocks of connected pixels. To tell whether an orange region is a ball or not, a fixed number of features could be computed to construct the feature vector. Since the orange region of the ball is round, one could fit a bounding circle to compute features such as the fitness of the region contour to the bounding circle. This transforms a set of variable-sized blocks of pixels into a fixed dimensional feature vector, which enables the direct use of generic kernel functions. However, fitting a bounding circle would not work similarly if objects to detect have different shapes, such as T-shirts or bags. Different features customized to the specific class of objects to detect are necessary. Unfortunately, finding such custom features for complex objects is very tedious and may be even possible.

An alternative approach is to find kernel functions for such data types that can effectively combine the structural information with the local features into similarity metrics. For instance, the pyramid math kernel (PMK) function computes the similarity between two unordered sets of feature vectors by partially matching them within a hierarchy of histogramming bins [23]. PMK is directly usable in the traditional kernel algorithms based on convex optimization since it is positive and semi-definite. As another example, spectrum kernel compares two strings by counting how many (contiguous) substrings of length p they share and constructing the histogram of the frequencies of all possible common substrings of all possible

lengths (p-spectra) [25, 33, 49, 61, 62].

The disadvantage of this approach is that it is not straightforward to find such kernel functions from the structure of the inputs. Neither is it straightforward to use or modify existing kernel functions such as PMK or spectrum kernels to handle inputs with other types of structures such as trees or graphs. PMK, for instance, has been effectively used in many computer vision tasks including image retrieval, object detection, and clustering to match unordered sets of local features computed from two images. However, it does not support inputs where local features form a more complex structure than unordered sets, such as trees or graphs. The same is true for spectrum kernels which is specifically tailored for sequential inputs which are common in tasks such as text or document classification. More flexible techniques for designing kernels that can be used across a wide range of data types would be much more powerful, but are currently lacking.

The motivation of this work is at the search for a systematic approach to encode the geometric structure of local features into kernel functions. It will provide a straightforward scheme to synthesize kernel functions for the given structure, which could be used in discriminative learning algorithms without the need for transforming input data into fixed dimensional vectors. The approach I propose in this thesis provides the ability to mechanically *tailor* kernel functions customized specifically for a given application domain by making an explicit encoding of the geometry that is used in conjunction with the local features.

1.2 Dissertation Contribution

I first introduce a method to encode the geometric structure of local features. Then I provide a systematic framework to combine the encoded structure and the local features to construct an effective measure of similarity between the geometrically structured data. The constructed similarity metrics on average require *linear* or *close to linear* time to compute.

Encoding the geometric structure begins by imposing a manifold on it. For instance, we could impose a 1D manifold for a sequence of points in a handwritten character, or a 2D manifold for a set of fiducial points in a 3D face image. The proposed method encodes geometric structure by associating each local feature with a point in this manifold. This implements the structural similarity as a distance metric for the manifold. I combine this distance with the problem-specific similarity between local features in a framework called *parametric kernel functions*, which define the similarity metrics between geometrically structured data in a systematic manner that is applicable to a variety of problem domains. This framework provides *tailorability*, which is defined as *the ability to aggregate atomic kernel functions to construct custom kernel functions for structured inputs via the association of parameters with the local features of structured inputs*. The similarity measure provides information about the distribution of the input examples in a space where the class boundaries are searched for. In addition, to allow for the application of kernel-based learning algorithms such as SVM to structured data in this framework, I show that parametric kernel functions are positive and semi-definite. For certain

kernel-based methods that require Mercer kernel functions, this is a necessary and sufficient condition to guarantee the existence of a unique optimal solution [49, 57].

Kernel functions for geometrically structured inputs would be ideal if they have the following properties. 1) Synthesizing kernels that are customized for the given structure is systematic. 2) The kernel functions are computationally efficient to evaluate to handle large inputs. 3) There is no need to explicitly evaluate probabilistic models to model the generative process of the patterns. 4) The synthesized kernels are positive semi-definite to guarantee a unique solution when used in kernel algorithms based on convex optimization. 5) Tailorable kernel functions for one problem are easily modifiable for other problem domains with similar input structures. While previous approaches fail to satisfy some or all of these, all of the requirements are satisfied by our approach.

I applied the proposed technique to a number of machine learning tasks with a variety of different input formats, characteristic features, and geometric structures relating these features. I demonstrate the effectiveness of our approach by showing that I achieve classification performance that is competitive with state of the art techniques for those tasks using application-specific methods. Yet we achieve this with kernel functions that are synthesized in a common framework. I apply this method to well-studied representative tasks for two important types of geometric structures that appear in a wide range of problem domains.

The first type is *sequences*. Over a wide range of robotics or HCI applications, the inputs are represented as sequences of *raw data vectors*, of possibly variable lengths, when measurements are taken from a system over time. For instance,

inputs for on-line handwritten character recognition are represented as sequences of 2D points which are obtained by the regular time sampling of mouse or input pen cursor position, while inputs for voice recognition or music following are digital audio streams which are obtained by the regular time sampling and the quantization of the amplitude of sound waves captured by the microphone. Examples of sequential inputs that are not necessarily ordered over time include the range information captured by laser sensors. Such devices take measurements of distances to the surrounding objects omni-directionally at certain angular resolutions. Inputs for common robot control tasks such as object detection or avoidance are sequences of distance values which are ordered by the angles that the measurements are taken. As the representative task, we implement and test our method for the task of on-line handwritten character recognition. In addition, we will address the aspects to be considered specifically for handling inputs where local features are just raw data vectors.

The second type is *unordered sets*. Unordered sets of *local feature vectors* have been shown to be an effective representation of objects in many computer vision tasks. For instance, combinations of features computed from images, e.g. sets of local energy maxima points or SIFT keypoints, are used as inputs for image classification or object detection and recognition tasks. Other examples include inputs for molecular docking simulation systems which are sets of electromagnetic fields computed at each of the molecules. As the representative tasks, we implement and test our method for the task of face recognition from video streams. This show that utilizing the geometric structure in conjunction with the local features improves

the classification performance.

Lastly, we present the result of applying our method for object detection from the laser range finder sensor data. No comparison with other competitive approaches is made because it is not a much studied problem with any public benchmark data and reported results along with any test criteria.

The novelty of the proposed technique is at the explicit representation of geometric structures in terms of parameters. Parameter space decomposition scheme greatly reduces the size of search space when matching local features, while retaining quality matches. Parametric kernel framework provides a tool to flexibly construct similarity metrics where the geometric structure is easily combined with the local features. Parametric kernels are efficient to evaluate and could be directly used in the conventional kernel framework.

1.3 Dissertation Organization

The remainder of this dissertation is organized as follows. In Chapter 2, I give a more detailed overview of the problem setting and a discussion of related work. In Chapter 3, I define the core parametric kernel algorithm. Following these presentation of the algorithmic components of the proposed method, I provide experiments and results demonstrating these ideas applied to several HCI and robotics problems. Specifically, in Chapter 4, I present on-line hand written character recognition results with a variety of dataset and demonstrate the efficiency and the ease of learning handwritings without the knowledge of any language. In Chapter 5, I present

face recognition experiments and results and show how the extension to higher dimensional manifolds seamlessly provides the same performance gain. Finally, in Chapter 6, I develop a method for detecting objects from sensor data using the same kernel framework for handwritten character recognition.

Chapter 2

Background and Related Work

In this section, I present the basic concepts and the related work for an in-depth understanding of our approach.

2.1 Research Domain

Our goal is a systematic framework to tailor kernel functions for geometrically structured data types. To meet this goal, I apply our method to a number of machine learning tasks of which inputs have a number of different types of representative geometric structures. I demonstrate the effectiveness of our approach by showing that this approach achieves classification performance that is competitive with the state-of-the-art techniques for those tasks. A strong emphasis is added to the fact that I achieve this with kernel functions that are synthesized in a common framework.

To complete our demonstration with a realistic amount of work, I had to limit

the number of applications that I test our method. But to maintain the generality, I chose to apply our method to well studied representative tasks with two important types of input data structures that could cover a wide range of problem domains.

The first type is *sequences*. Over a wide range of robotics or HCI applications, the inputs are represented as sequences of *raw data vectors* of possibly variable lengths when measurements are taken from a system over time. As the representative task, I implement and test our method for the task of on-line handwritten character recognition. In addition, I will address the aspects to be considered specifically for handling inputs where local features are just raw data vectors.

The second type is *unordered sets*. Unordered sets of *local feature vectors* have been shown to be an effective feature representation in many computer vision tasks. As the representative tasks, I implement and test our method for the task of on-line face recognition from video streams. This demonstrates how inputs with no specific *order* between the local features are handled. But more importantly, I show that utilizing the information about the distribution of the local features over the input images improves the classification performance.

Lastly, I present the result of applying our method for the task of ball recognition from the sensor data captured by a laser range finder. No comparison with other competitive approaches is made because it is not a much studied problem with any public benchmark datasets and reported results along with any test criteria. Rather, the purpose is at the demonstration of the generalizability of our approach to a variety of problem domains.

2.2 Kernel-based Learning Methods

Support vector learning is one of the state of the art machine learning techniques that has been used extensively in this research. It is a widely used technique that has been applied to numerous classes of problems including classification, regression, novelty detection, and clustering, to name just a few. We present an introduction to support vector classification (SVC) below as background for the results we will show later.

2-Class Support Vector Classification Given two classes of training inputs (denoted as $\circ$ and $\times$ in Figure 2.1), the objective of SVC is to find a hyperplane f that separates two classes with the maximal margin h.

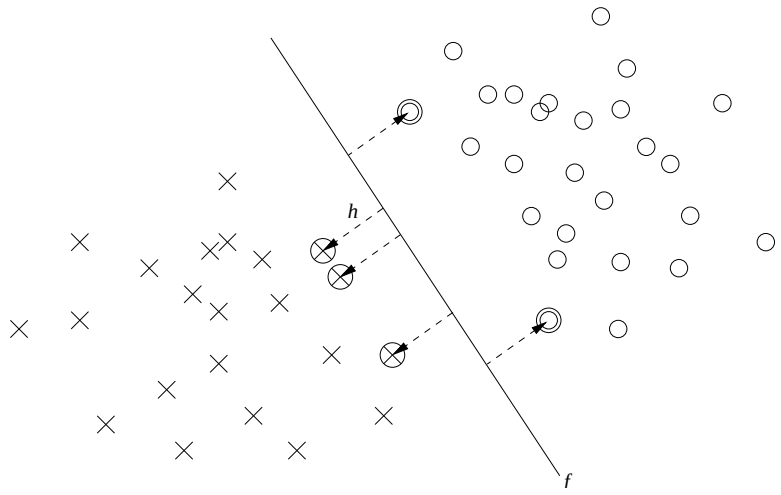


Figure 2.1: An example of support vector learning; 2-class classification

Suppose ℓ training examples $\mathbf{x}_i$ for $i = 1, \cdots, \ell$ are taken from $\mathbb{R}^d$. Each $\mathbf{x}_i$

is labeled as $y_i = 1$ if it is a positive example, or $y_i = -1$ if it is a negative example. Denote the hyperplane as

$$f(\mathbf{x}) = \text{sgn}\big(\langle \mathbf{w} \cdot \mathbf{x} \rangle + b\big), \tag{2.1}$$

where $\mathbf{w} \in \mathbb{R}^d$ is the weight vector and b is the bias. f is found by solving the following quadratic program :

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2 \\ &\text{subject to} \quad y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1, \text{for } i = 1, \cdots, \ell. \end{aligned}$$

The general technique to find the solution is to solve its dual problem. To convert this into the dual form, the Lagrangian of this quadratic program is first differentiated

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{\ell} \alpha_i \big[y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1\big], \tag{2.2}$$

where the non-negative variables α_i for $i = 1, \cdots, \ell$ are called the Lagrangian multipliers, with respect to the primal variables $\mathbf{w}$ and b, and set them to zero for stationarity,

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{\ell} y_i \alpha_i \mathbf{x}_i = \mathbf{0}, \frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial b} = \sum_{i=1}^{\ell} y_i \alpha_i = 0. \tag{2.3}$$

Rewriting the given quadratic problem solely in terms of the dual variables yields the following dual problem

$$\begin{aligned} &\text{maximize} \quad \textstyle\sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle \\ &\text{subject to} \quad \textstyle\sum_{i=1}^{\ell} y_i \alpha_i = 0, \text{ and } \alpha_i \geq 0 \text{ for } i = 1, \cdots, \ell. \end{aligned}$$

Suppose the training set is linearly separable and let α_i^* for $i = 1, \cdots, \ell$ be the solution to the dual problem. Then, $\mathbf{w}^* = \sum_{i=1}^{\ell} y_i \alpha_i^* \mathbf{x}_i$ realizes the maximal margin hyperplane. In this case, the bias is found as

$$b^* = -\frac{\max_{y_i=-1} \big(\langle \mathbf{w}^* \cdot \mathbf{x}_i \rangle\big) + \min_{y_i=1} \big(\langle \mathbf{w}^* \cdot \mathbf{x}_i \rangle\big)}{2}. \tag{2.4}$$

Plugging $\mathbf{w}^*$ into (2.1) yields

$$f(\mathbf{x}) = \text{sgn}\bigg(\sum_{i=1}^{\ell} y_i \alpha_i^* \langle \mathbf{x}_i \cdot \mathbf{x} \rangle + b^* \bigg). \tag{2.5}$$

The Karush-Khun-Tucker conditions, $\alpha_i^* \big[y_i (\langle \mathbf{w}^* \cdot \mathbf{x}_i \rangle + b^*) - 1 \big] = 0$, for $i = 1, \cdots, \ell$, state that the sign of the functional margin $y_i \big(\langle \mathbf{w}^* \cdot \mathbf{x}_i \rangle + b^* \big)$ equals 1, i.e. $\mathbf{x}_i$ is at the margin of f, if and only if $\alpha_i^* > 0$. Otherwise, $\alpha_i^* = 0$, in which case, the corresponding term in equation (2.5) vanishes. It is therefore only the set of examples located at the margin of the hyperplane that actually constitutes the solution. Such training examples are called the *support vectors*. The fewer the support vectors, the sparser the solution becomes, and vice versa.

Kernel Trick The training set may not be linearly separable in the input space. In such cases, mapping to a higher dimensional space may yield a linear solution [17]. Let $\phi : \mathbb{R}^d \rightarrow H$ be a mapping from the input space to a hyperspace H. Consider mapping the training examples into H using ϕ and find f in H. Then (2.5) in H

becomes

$$f(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^{\ell} y_i \alpha_i^* \langle \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}) \rangle + b^* \right). \tag{2.6}$$

The Representer's theorem guarantees that $\mathbf{x}_i$ and $\mathbf{x}$ appear only in the form of an inner product. If there exists a function $\kappa : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ that evaluates the inner product of two vectors $\phi(\mathbf{x})$ and $\phi(\mathbf{z})$ in H, directly using two input vectors $\mathbf{x}$ and $\mathbf{z}$ in $\mathbb{R}^d$, without explicitly evaluating the mapping function ϕ for $\mathbf{x}$ and $\mathbf{z}$ and their inner product in H, then we could rewrite (2.6) in terms of κ as follows,

$$f(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^{\ell} y_i \alpha_i^* \kappa(\mathbf{x}_i, \mathbf{x}) + b^* \right). \tag{2.7}$$

The first advantage of using functions like κ, called *kernel functions*, is to avoid the heavy computation of mapping the ϕ and taking their inner product. Since kernel functions correspond to inner products in hyperspaces, the necessary and sufficient condition that κ must meet to be a valid kernel is that it is positive and semi-definite. That is, for any given dataset, the Gram matrix constructed using κ must have only non-negative real eigenvalues. This guarantees a unique solution to the quadratic program of SVM.

When kernels are used in practice, kernel functions are chosen directly rather than by mapping the ϕ since it may not be possible to find a closed form mapping of ϕ that corresponds to a kernel function. However, a valid kernel guarantees the existence of a hyperspace that ϕ maps the input vectors to. Examples of valid kernel functions include linear kernels, radial basis functions (RBF), or polynomial kernel

functions.

The second advantage of using kernel functions is that, in support vector learning, non-linear decision boundaries could be found in a simple linear framework.

Kernel functions evaluate the inner product in a hyperspace in a technical sense. Meanwhile, the semantics of what is being computed by kernel functions is the similarity between the input vectors. Hence, a general guide to determining the right kernel function for a given problem domain is to choose a kernel function that evaluates to a large (small) value if the inputs are semantically similar (dissimilar). However, most approaches that use generic kernels implement the semantics of similarity by carefully selecting a set of features to construct the input vector. RBF, for instance, computes the distance between any two vectors of the same dimension which exponentially decreases proportional to their geometric distance. Linear kernel computes the inner product of any two vectors. If normalized by the lengths of the two vectors, then linear kernel computes the similarity between two vectors as the cosine of the angles between them. Therefore, the generic kernels in this sense are chosen somewhat arbitrarily. Considering that the type of kernel has a direct impact on the classification performance and the learning complexity, determining a suitable kernel function given a problem domain is an open problem that is challenging.

So far, SVMs have shown excellent performance in quite a number of machine learning problems. In part, this is due to the structural risk minimization (SRM) scheme of support vector learning. The purpose of SRM is the minimiza-

tion of the generalization error of a learning scheme. According to the upper bound of the generalization error of a classifier f that is provided by the VC theory, maximizing the margin h of a given training set minimizes the upper bound of the generalization error.

The immediate disadvantage of using kernel functions is that the determination of kernel functions along with the related parameters is a challenging task. Another more serious problem is that often the implementation requires a large memory to store the kernel matrices. However, algorithms such as sequential minimal optimization (SMO) that do not require a huge memory space for storing the kernel matrices are available as well.

2.3 Learning Structured Data

The two most important components of most applications of machine learning techniques to real world problems are feature extraction and the learning algorithm. Feature extraction is related to the representation the input data, while the learning algorithm is related to finding and modeling the underlying relationship between the features. Feature extraction removes noise and summarizes important characteristics of the input data in a form that can be handled by the learning algorithms to be used. Learning algorithms either use the extracted features in conjunction with generic metrics, e.g. distance or inner product, or define custom metrics to compute the similarity between input examples.

Many learning algorithms assume inputs are in a single feature space of

some fixed dimension, but structured inputs are also quite common. There are two categories of input structures. One is *geometric* and the other is *discrete*. Geometrically structured inputs consist of local features that are contained in a continuous space, e.g. points in 2D plane, while those of discretely structured inputs are elements of a discrete set, e.g. alphabets of a language. In this research, I consider geometrically structured inputs.

Learning Sequence Data Sequence data consist of *ordered* local features. They form a 1-dimensional manifolds. Computing the similarity between two input sequences is challenging because it is not a straightforward task. Two categories of approach are available.

First, one could extract fixed dimensional feature vectors from sequences and use generic similarity measures, though it is difficult to maintain structural information. In [54], a fixed size feature vector is computed from strokes, and an SVM classifier is used. Features used include the mean coordinates and second order statistics such as median, variance, minimum and maximum distances, area, etc. They achieved a high accuracy of over 98%. Such features do not generalize well because they allow only fixed feature vectors for each stroke. In [32], Extended R-squared (ER^2) is proposed as the similarity measure for sequences. Though it uses the coordinates of points directly as features, they can only operate on point sequences of fixed-length so as to compute the proposed similarity metric. In addition, such features are heuristically chosen. Hence, it is very difficult to apply feature vectors that work well for problems of one domain to those of

other domains. Extracting feature vectors of uniform dimension is also possible through histogramming [11, 23, 43]. Histograms are an effective scheme to map varying length sequences into uniform dimensional feature vectors. For instance, Porikli computed histograms of locations, speed, size, aspect ratio, etc. of points in trajectories to extract fixed dimensional feature vectors [43]. Unfortunately, information about the ordering of the points in the trajectories is lost in the process of histogramming.

One can instead extract a set of fixed dimensional feature vectors from sequences. The incline scanning n-tuple classifier (OnSNT) is a very fast and accurate method to learn from sequences. This is based on the standard n-tuple (SNT) classifier proposed by Aleksander and Stoham [7]. OnSNT first extracts both static and dynamic features into chain codes and then a sliding window is scanned across the chain code sampling it into n-tuples that become the features, or "addresses", presented to a probabilistic n-tuple classifier [35]. OnSNT has been applied to both on-line and off-line recognition of handwritten characters and showed excellent performance [36, 45]. Since OnSNT and SNT assume sequences of elements in discrete input space, the input handwritten characters must first be converted into sequences of elements from a discrete space, e.g. alphabets or finite sets of points in $\mathbb{R}^d$. For instance, from a black and white image of handwritten characters, edge transitions are encoded into top-down/left-right/black-white. Similar technique is used to convert handwritten characters represented as 2D points sequences into sequences of elements from a discrete space. But finding such a transformation is in general not a straightforward process.

Alternatively, we could use methods that compare variable length sequences directly, e.g. dynamic time warping (DTW). DTW is an effective method to compute the similarity, or equivalently, distance between two sequences such as speech or handwritten characters [9, 50]. For two sequences of lengths n and m, DTW first computes an $n \times m$ distance matrix $\mathbf{M}$, where $\mathbf{M}_{ij}$ is the distance between i-th element of one sequence and j-th element of the other one. Then, DTW matches elements by walking along a path in $\mathbf{M}$ from $\mathbf{M}_{11}$ to $\mathbf{M}_{nm}$. A diagonal step in this path indicates a match. Total distance is the summation of distances of matched elements. The path that gives minimum total distance is found using dynamic programming. An extensive survey has shown that DTW methods are among the most effective techniques for classifying sequence data [29, 32]. In [9], normalized coordinates and the tangent slope angle are computed at each of the points in a stroke sequence to form a feature vector. The distance is then computed using DTW, which is used as the exponent of a radial basis function (RBF).

DTW is useful because the similarity of variable length sequences can be systematically computed. It does so by non-linearly matching sequences, skipping elements that yield sub-optimal solutions. However, at the same time, information is lost in doing so. Unmatched elements may exhibit useful information for further learning. Also, the path of comparison must start from the top left element of comparison matrix. This means that at least one of the two sequences match from its first element. This will be a limitation if we need to partially match both sequences. In this case, one must start the path from a position other than anywhere in the top row or leftmost column. Similar reasoning applies to the end point as

well. However, DTW does not support any mechanism for the determination of the optimal starting and ending points. A set of subsequence matching techniques has been proposed to match subsequences instead. Another problem with DTW is the heavy computation needed for matching, i.e. $O(nm)$, which makes application to long sequences prohibitive. DTW based kernels can be shown to be symmetric and to satisfy the Cauchy-Schwartz inequality [50]. SVM classifiers may employ such kernels to classify sequential data. However, as the authors of [50] pointed out, kernels based on DTW are not metrics. The triangle inequality is violated in many cases and the resulting kernel matrices are not positive semi-definite. Therefore, they are not admissible for kernel methods such as SVM in that they cannot guarantee the existence of a corresponding feature space and any notion of optimality with respect to such a space. This, however, may not be a problem if one uses learning methods that do not involve convex optimization.

On-line Handwritten Character Recognition On-line handwritten character recognition is aimed at recognizing the movements of the character input devices such as digital pens or mice that are represented as sequences of points into symbols as the characters are written. This is different from the task of recognizing symbols from the images of handwritten characters. Previous work on on-line handwritten character recognition can be roughly grouped into two categories of feature extraction and direct matching, depending on how two handwritten characters are compared. In feature extraction, a fixed number of features is computed from the strokes which are compared using a generic similarity metrics e.g. inner product of

RBF [8,32,45,54]. The advantage of this type of approach is that many conventional learning algorithms are directly applicable that assume vector inputs and the major drawback is the difficulty in representing the information about the ordering of the points as a fixed dimensional vector. In comparison, approaches in direct matching matches points in the strokes directly using techniques such as DTW [9, 50]. The advantage of this approach is that it is easy and straightforward to compare sequences of variable lengths. The drawback is that sequences of variable lengths do not fit traditional learning algorithms and distance metrics that assume vector inputs.

Learning Unordered Sets Many representations used in computer vision consist of unordered sets of features. An example is SIFT keypoints. SIFT computes affine-invariant points in an image based on an analysis using multi-resolution convolutions. To compute the similarity between the two sets of SIFT keypoints of two different images, Lowe suggests to match keypoints based on the distances between descriptors [34]. If more than a certain number of keypoints match between two images, then the images may be considered to contain the same object. SIFT descriptors can be computed efficiently, but the suggested similarity metric between two sets of SIFT descriptors is not a Mercer kernel. We can easily see this by recognizing that it is not symmetric.

Grauman proposed pyramid match kernel (PMK) which computes the similarity of two unordered *sets* of local features [23]. It can handle inputs that consist of unordered sets of features or parts with varying cardinality, where the correspon-

dence between the features across each set is often unknown. It is a valid Mercer kernel based on pyramid or multi-resolution histograms, that guarantees convergence to a unique optimum when used with methods that require positive semi-definite kernels. Kernel computation is extremely efficient $O(dm \log D)$, where d and m are the sizes of two input sets of features and D is the diameter of the smallest sphere that bounds the features.

Local features of this type of data are scattered around in a space that is often different from the space in which the local features are contained. For instance, SIFT keypoints are computed at various *locations* within the image or the electromagnetic fields of molecules or fiducial points in 3D face images are located at various positions in $\mathbb{R}^3$. The locations at which the local feature vectors are computed form a geometric structure by themselves, though there is no notion of order. Unfortunately, this information has been neglected in many approaches. In part, this is because its use resulted in an adverse effect. Our results show, however, that this structural information, if used right, could improve the classification accuracy dramatically.

Numerous similarity metrics for unordered sets of local features have been categorized into voting, bags of prototypical features, and correspondence-based approaches [23]. PMK overcomes the difficulties of these methods by incorporating co-occurrence feature statistics and cross-bin matchings and doing so with a significantly lower computational demand. However, PKM considers multi-level histogramming only in the feature space. My parametric kernel function is similar to PKM in that it incorporates co-occurrence and can flexibly support cross-bin

matching. In contrast to PKM, my approach takes the holistic geometric structure of local features through the use of meta information.

Face Recognition from Video Streams Given an arbitrary image, the goal of face *detection* is to determine whether or not there are any faces in the image and, if present, return the image location and extent of each face. Face *recognition* is the task of comparing an input image against a database of faces of different people and finding a match to a specific person [66]. We apply our tailorable kernel technique to the task of on-line face recognition in real-time on mobile robots. In general, face recognition is a harder problem to solve than face detection because human faces are distinct from other types of objects but have small intra-class variations. It is one of the most studied problems in image analysis and computer vision, with various successful results obtained. Recently, it has drawn significant interest due to the wide range of commercial and law enforcement applications, and the availability of feasible algorithms and hardware systems after 30 years of research [67]. Though there have been many promising face recognition methods, face recognition robust against significant pose and illumination variations is still a very difficult task [67].

Most earlier work in face recognition is single image based. Face regions are first detected from images by separating faces from background area. From the detected face images, features are extracted for further recognition. Feature extraction methods can be categorized into two groups: holistic or component-based. In holistic approaches, a single feature vector is used to represent the face. For instance, Karhunen-Loeve (KL) expansion is used to represent features as low dimensional

vectors of coefficients of orthogonal components known as eigenfaces computed from principal component analysis (PCA) [55]. Other techniques include Fisher's discriminant analysis [10], neural networks [20] and non-negative matrix factorization (NMF) [31]. Holistic approaches, however, fail to achieve our goal because they are highly sensitive to pose variations [26]. Such descriptors require that the face images being compared are registered, i.e. well aligned, to a reasonably high precision [53]. This is also what I have observed from my implementations of face recognizers using PCA and NMF.

Component-based methods locate and extract facial components such as eyes, mouth, and nose and construct features from them [26]. For instance, Gabor wavelets can be used to detect scale-invariant facial components [65]. The advantage of component-based methods is that they do not require accurate alignment of face images. However, facial component extraction may be too slow (e.g. 4 frames per second [64]) for real-time applications. Combined with face detection, overall processing speed drops below 2 Hz. Or facial component extraction may be very difficult to use [22]. Also, recognition performance degradation is observed if facial component extraction is not accurate enough [67].

Certain face recognition methods require special hardware, e.g. a Smart Camera [19], which may be prohibitively expensive. Many other methods are demonstrated by training and testing image from databases such as FERET [65], the CMU PIE database [66], or the Yale Face B dataset. However, such image databases are often constructed with care in a controlled environment, i.e. faces are relatively well aligned and illumination varies in a predictable manner. In com-

parison, in uncontrolled environments as in our problem, the variation in pose and illumination is too extreme, making it almost practically impossible to build such an image database.

Recently, research has focused more on face recognition from video sequences [15]. However, face recognition from video streams captured by surveillance cameras or webcams is still difficult because the image and color quality of the video is low. Tracking head and facial components may enhance face recognition by correctly registering face images. If the camera position is stationary as in surveillance systems, this is achievable because the lighting conditions remain reasonably stationary as well. However, those techniques become mostly infeasible if both the camera and the subject are moving because pose and illumination vary significantly.

SIFT extracts scale and rotation invariant features from images. SIFT features are also partially invariant to changing viewpoints and illumination [34]. It has been used in tasks such as matching different views of an object or scene e.g. stereo vision, object recognition, and robot localization. SIFT has recently been applied to face recognition and promising results are reported [13, 53]. However, the representational ability of SIFT features for face recognition applications has rarely been investigated systematically [37]. Other similar feature extraction methods that find affine-invariant interest point descriptors include [38]. But most are computationally expensive for real-time purposes [14]. Based on our experiments, SIFT recognized faces very well if illumination remains stationary, but slight changes in the direction of lighting result in a significant degradation of recognition perfor-

mance. Nevertheless, I have chosen SIFT as the feature representation because of its advantages including robustness against pose variations, no need for expensive face image registration, and computational efficiency.

Perhaps the most closely related work to ours is that of Tangelder and Schouten [53]. They used an image descriptor called bi-cubic interpolated and histogram equalized gray-scale image (BHG) descriptor to represent detected face images. In the preprocessing phase, the IDIAP frontal face detector [46] extracts near-front face regions of size $n \times n$ pixels, $n \geq 24$. BHG is computed from a detected face image by resizing it into $s \times s$ images ($s = 8, 16, 24, \cdots$) using bi-cubic interpolation and applying histogram equalization for illumination invariance. Tangelder and Schouten compare BHG against other feature representations including SIFT in the framework of face recognition using still images contained in video stream recorded in unconstrained environments [53]. A sparse representation of the most discriminant descriptors learned by a greedy search method is used as the training dataset. Their analysis claims that their method achieves a recognition rate of 94% with a sparse representation containing 10% of all available data, at a false acceptance rate of 4%.

The framework of this method is similar to ours in that a set of image descriptors is constructed during training but differs in a number of aspects. First, test images are captured in a much more controlled environment than ours. The video clips of the ITT-NRC facial video database that are used to test the proposed method are shot under approximately the same illumination conditions (no sunlight, only ceiling light evenly distributed over the room), the same setup and almost the

same background, for all people in the database [21]. This was to test the recognition performance with respect to factors inherent in video-based face recognition such as low resolution, motion blur, out-of-focus, facial expression and orientation variation, and occlusion. In comparison, the camera is not stationary in our system as it is mounted on a mobile robot. People can walk freely in a space where lights are unevenly distributed. In addition, the lighting in video clips of ITT-NRC is brighter than that in our environment. Second, the greedy descriptor selection is not an on-line learning method, thus not directly applicable in our framework.

2.4 Summary

In this chapter, I presented a set of approaches related to the work in this dissertation. Structured inputs do not fit traditional learning methods and distance metrics that assume vector inputs. Neither is it easy to represent the irregular structure into uniform length vectors. Fortunately, we can provide a solution to this problem by defining kernels for structured inputs.

Kernels for sequences and unordered sets of local feature vectors have been proposed. They have been applied to a wide range of applications including on-line handwritten character recognition, trajectory classification, object recognition, or image category learning and retrieval, etc. It is difficult to represent the information about the ordering of the points in strokes in uniform length feature vectors. DTW does not yield a distance metric that cannot be used in kernel-based learning algorithms and is in general not applicable to structures other than sequences. Kernels

such as PMK has successfully computed the distance between two unordered sets of local features. However, the usefulness of the information about the distribution of the local features over the input images has not been addressed enough.

Our review motivates the need of a more systematic approach to build kernels that effectively model the structure among the local features.

Chapter 3

Parametric Kernels

This chapter introduces the fundamental concept of parameterization and parametric kernel functions that provides the ability to tailor kernel functions to adapt to the structures of local features in the inputs. As mentioned in Chapter 1, the underlying intuition is to capture the structural information in the notion of *manifolds* of some dimension. For convenience, the presentation is based on but not limited to sequence structures.

3.1 Parameterization

Consider an input $\mathbf{x}$ represented as a sequence of n elements $x_i \in X$, i.e. $\mathbf{x} = [x_1, \cdots, x_n]$. We could choose to associate each element x_i with parameter $\boldsymbol{\tau}(x_i)$

in parameter space T as follows

$$\boldsymbol{\tau}(x_i) = \begin{cases} \sum_{k=2}^{i} \|x_k - x_{k-1}\| & \text{if } i > 1, \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

The associated parameters form a non-decreasing sequence of non-negative real numbers, starting from zero. This is equivalent to arc length computed from piece-wise linear interpolation of the x_i. For any two such sequences, a parametric kernel at each iteration picks one element from each of the sequences and computes the similarities between the elements and their associated parameters separately. These are multiplied to return an overall similarity between the elements. This product of the similarities of the elements and their parameters implies that two similar elements may contribute significantly to the overall sequence similarity only when their associated parameters are similar as well. This step is repeated for all pairs of elements from the two sequences, and the results are summed to return the overall sequence similarity.

Naïve application of this approach has the potential to be swamped by the computational expense of performing many comparisons between elements with widely divergent parameters which contribute little or nothing to the final result. Intuitively, we could handle this by limiting the comparisons only to subsets of elements that are *close* in parameter space. This closeness in parameter space is easily specified by the decomposition of parameter space into ranges, so that close elements are defined to be those whose parameters fall into the same range. For instance, we could decompose T into non-overlapping intervals of equal length Δ,

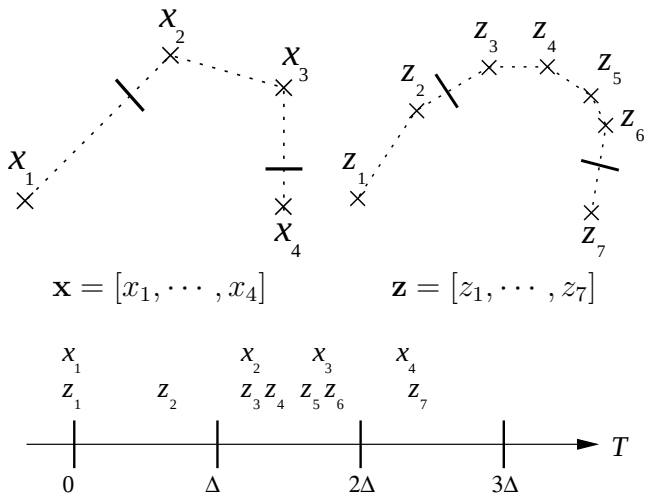


Figure 3.1: Mapping Sequence to Parameter Space

and elements from the two sequences are close if their associated parameters fall into the same interval. See Figure 3.1, where elements in $\mathbf{x}$ and $\mathbf{z}$ are grouped into three ranges based on the aforementioned decomposition scheme. x_1, for instance, will be compared against only z_1 and z_2, both in X and T. Any *sequence* data types fall into this category, e.g. handwritten characters, laser sensor readings, digital signals, and so on.

The underlying intuition in our work is to associate a *parameter* with each of the elements so that enforcing parametric similarity is equivalent to the similarity in the structure of elements in the input patterns. For example, handwritten characters are sequences of 2D points, while images are often converted into unordered sets of d dimensional local features. The *structure* of handwritten characters is a 1D manifold in $\mathbb{R}^2$ and that of unordered sets of d dimensional local feature vectors

computed from images is a 2D manifold in $\mathbb{R}^d$. A parameter of an element is then a point in this manifold that corresponds to the element. If parameters of any two elements are close, then they are structurally close. This parametrization is part of our kernel design scheme. Though we propose methods of designing kernels for variable length sequences of local feature vectors, mathematical extension of our formulation to structures of higher dimensional manifolds is straightforward.

3.2 Parametric Kernel

Our input pattern $\mathbf{x}$ is a sequence of $|\mathbf{x}|$ elements, where each element x_i is a d-dimensional vector, i.e. $\mathbf{x} = [x_1, \cdots, x_{|\mathbf{x}|}]$ and $x_i \in \mathbb{R}^d$. We associate each element with a parameter in parameter space T via a function $\boldsymbol{\tau} : \mathbb{R}^d \rightarrow T$. Consider a decomposition of T into N non-overlapping ranges

$$T = \bigcup_{t=0}^{N-1} T_t. \tag{3.2}$$

For instance, recall the earlier sequence example, where T was the set of non-negative real numbers and $\boldsymbol{\tau}$ is defined as (3.1). T was decomposed as shown in Figure 3.2.

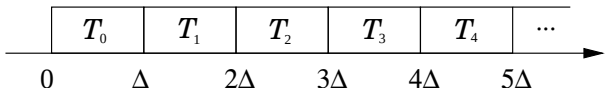


Figure 3.2: Parameter space is decomposed into non-overlapping ranges of length Δ.

For the derivation of parameter kernel functions, we first define the *decomposed element set* for T_t as $\mathcal{I}_t(\mathbf{x}) = \{x_i | \boldsymbol{\tau}(x_i) \in T_t\}$, which is the set of elements of $\mathbf{x}$ whose associated parameters are in T_t. In our previous example shown in Figure 3.1, for instance, $\mathcal{I}_1(\mathbf{x}) = \{x_2, x_3\}$ and $\mathcal{I}_1(\mathbf{z}) = \{z_3, z_4, z_5, z_6\}$. We then compute a similarity for each range by taking a weighted sum of the similarities of every pair of elements of the sets being compared whose parameters fall within the range. For T_1, we will compare x_2 with z_3, x_2 with $z_4, \cdots$, and x_3 with z_6. The similarity for a given pair of elements is obtained by taking the product of the similarity $\kappa_{\boldsymbol{\tau}} : T \times T \rightarrow \mathbb{R}$ between those elements' parameters and a similarity $\kappa_x : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ defined directly on the elements themselves, each of which is a Mercer kernel function. Then, the feature extraction function $\boldsymbol{\phi}$ of a parametric kernel is defined as

$$\boldsymbol{\phi}(\mathbf{x}) = [\phi_0(\mathbf{x}), \phi_1(\mathbf{x}), \cdots \phi_{N-1}(\mathbf{x})], \tag{3.3}$$

where

$$\phi_t(\mathbf{x}) = \sum_{x_i \in \mathcal{I}_t(\mathbf{x})} w_{x_i} \kappa_x(x_i, \cdot) \kappa_{\boldsymbol{\tau}}(\boldsymbol{\tau}(x_i), \boldsymbol{\tau}(\cdot)), \tag{3.4}$$

and w_{x_i} is a non-negative weighting factor. Given two sequences $\mathbf{x} = [x_1, \cdots, x_{|\mathbf{x}|}]$ and $\mathbf{z} = [z_1, \cdots, z_{|\mathbf{z}|}]$, the parametric kernel function *before normalization* is then defined as the sum of the similarities for all ranges :

$$\kappa(\mathbf{x}, \mathbf{z}) = \langle \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{z}) \rangle = \sum_{t=0}^{N-1} \langle \phi_t(\mathbf{x}) \cdot \phi_t(\mathbf{z}) \rangle, \tag{3.5}$$

where

$$\langle \phi_t(\mathbf{x}) \cdot \phi_t(\mathbf{z}) \rangle = \sum_{\substack{x_i \in \mathcal{I}_t(\mathbf{x}) \\ z_j \in \mathcal{I}_t(\mathbf{z})}} w_{x_i} w_{z_j} \kappa_x(x_i, z_j) \kappa_{\boldsymbol{\tau}}(\boldsymbol{\tau}(x_i), \boldsymbol{\tau}(z_j)). \tag{3.6}$$

Note that the product of κ_x and $\kappa_{\boldsymbol{\tau}}$ is taken in (3.4). This means that both must score high to have significance in (3.5). It is worthwhile to compare this form of embedding parametric similarity against other possibilities. A simpler way, for example, is to use combined feature vectors $x'_i = (x_i, \boldsymbol{\tau}(x_i))$ and $z'_j = (z_j, \boldsymbol{\tau}(z_j))$ instead and use only κ_x. In this case, parametric similarity may non-linearly contribute to overall similarity depending on the kernel chosen, resulting in behavior that is difficult to predict. Also note that no comparison is made between elements that are not from a common range. If, instead, we have to compare all elements from one input with all from the other input, we will face a number of undesirable consequences. For instance, in Figure 3.1, we will compare x_1 with $\{z_1, \cdots, z_7\}$, rather than $\{z_1, z_2\}$. This will result in a higher similarity value, which may be helpful for certain cases. But, at the same time, we are more likely to be confused by inputs where there are too many *far* elements, in which case we are swamped by bad comparisons. Of course, we may need dramatically more time to compute (3.5) as the number of kernel evaluations is significantly increased since all elements are compared.

Parameter space decomposition solves such problems. However, such a decomposition scheme can introduce quantization errors. To overcome this problem, we allow the ranges to overlap. For instance, ranges in Figure 3.2 may overlap by $\Delta/2$, as shown in Figure 3.3. To suppress over-contribution of elements that fall

into the intersections of ranges, we introduce weighting factors in (3.4).

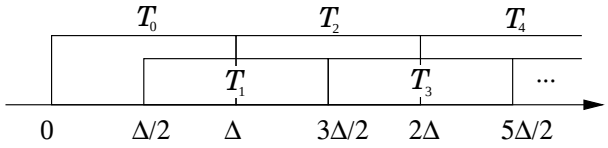


Figure 3.3: Ranges overlap by $\Delta/2$.

The simplest weighting scheme is to take the average of the similarity at the overlapped regions. In this scheme, the default value for w_{x_i} is $1/|\mathcal{T}_{x_i}|$, where $\mathcal{T}_{x_i} = \{T_t | \boldsymbol{\tau}(x_i) \in T_t\}$. When no ranges overlap, we have $|\mathcal{T}_{x_i}| = 1$ and therefore, $w_{x_i} = 1$. Otherwise, overlapped ranges may yield $w_{x_i} < 1$. For instance, with the decomposition scheme in Figure 3.3, at intersection $[\Delta/2, \Delta)$, we will set $w_{x_i} = 1/2$, since $|\mathcal{T}_{x_i}| = 2$. Note that this scheme will not result in $|\mathcal{T}_{x_i}| = 0$, i.e. $w_{x_i} \to \infty$, since (3.4) will be evaluated only when $\mathcal{I}_t(\mathbf{x}) \neq \varnothing$. If $\mathcal{I}_t(\mathbf{x}) = \varnothing$, then $\phi_t(\mathbf{x}) \equiv 0$ and the term is just ignored. Further discussion of different decomposition schemes is given in 3.3.

Finally, to avoid favoring large inputs, we normalize (3.5) by dividing it by the product of the norms of $\mathbf{x}$ and $\mathbf{z}$,

$$\boldsymbol{\kappa}(\mathbf{x}, \mathbf{z}) = \frac{\kappa(\mathbf{x}, \mathbf{z})}{\sqrt{\kappa(\mathbf{x}, \mathbf{x}) \times \kappa(\mathbf{z}, \mathbf{z})}}. \tag{3.7}$$

This is equivalent to computing the cosine of the angle between two feature vectors in a vector space model.

3.3 Parameter Space Decomposition Scheme

In this section, our decomposition scheme is discussed in general in terms of pros and cons with respect to additional cost of computation and changes in classification performance due to range overlapping and similarity weighting. Then, a number of different decomposition schemes are presented.

As mentioned above, decomposition lets us avoid swamping by bad comparisons and dramatically reduces the computational cost of kernel evaluation but introduces quantization error. This is alleviated by allowing for range overlapping and similarity weighting. However, overlapping must be allowed with care. Increasing the size of range overlaps will require additional computation since it is likely to involve more kernel evaluations as more elements are found in each intersection. The gain of decreasing quantization error may provide little improvement in classification performance if we are swamped by bad comparisons. Thus there is a trade off between quantization error and classification performance.

One issue left is the time to compute the decomposed element sets. The more complicated a decomposition scheme gets, the more difficult the implementation becomes and the more time it takes to run. Fortunately, our experimentation has revealed that classification performance is relatively insensitive to minor changes in the decomposition scheme. Therefore, we can often favor simpler decomposition schemes for ease of implementation and efficient kernel evaluation with the expectation of only minimal losses in performance.

We now present a number of example parameter space decomposition schemes.

Regular Decomposition Parameter ranges all have a common length Δ as in Figure 3.2 or 3.3. Implementation is simple, and it shows good classification performance in general. But we have limited freedom to fit the data.

Irregular Decomposition Parameter ranges are of varying lengths. Implementation is complicated and often decomposed element sets take longer to compute. But we can freely decompose the parameter space to better fit the data.

Multi-scale Decomposition Parameter ranges form a hierarchical structure at different resolutions. For instance, we may consider a decomposition where ranges form a pyramid as shown in Figure 3.4. Elements from non-adjacent ranges could be compared at coarser resolutions. Along with a proper weighting scheme, this may improve the performance. But implementation is more complex and kernel evaluation may take longer.

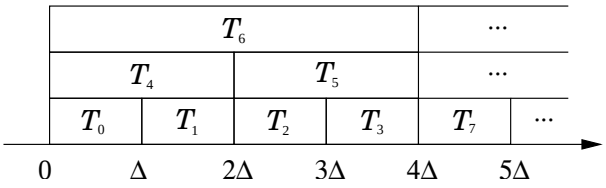


Figure 3.4: Pyramidal Parameter Space Decomposition

3.4 Mercer Condition

According to Mercer's theorem, a kernel function corresponds to an inner product in some feature space if and only if it is positive and semi-definite. A unique optimal solution is guaranteed by kernel methods only when kernels are positive and semi-definite. To see that our proposed method produces positive and semi-definite kernels, we first note that (3.4) is positive and semi-definite. Such kernels are called *summation kernels* and proven to be positive and semi-definite [25]. It is not difficult to see that (3.5) is just a sum of summation kernels. Since we can synthesize a new Mercer kernel by adding Mercer kernels [49], (3.5) is a Mercer kernel.

3.5 Efficiency

The time complexity to compute parametric kernels depends greatly on the particular decomposition scheme used. Here we provide a brief analysis only for regular decomposition schemes.

Assume that constant time is needed to evaluate κ_x and $\kappa_{\boldsymbol{\tau}}$ and that we are using a regular decomposition scheme as in Figure 3.2 or 3.3. Then the time complexity of evaluating (3.5) for sequences $\mathbf{x}$ and $\mathbf{z}$ composed of $|\mathbf{x}|$ and $|\mathbf{z}|$ elements, respectively, is, on average, $O(|\mathbf{x}||\mathbf{z}|\Delta/L)$, where $L = \max(\boldsymbol{\tau}(x_{|\mathbf{x}|}), \boldsymbol{\tau}(z_{|\mathbf{z}|}))$. In the worst case without decomposition, $\Delta = L$, so the complexity is $O(|\mathbf{x}||\mathbf{z}|)$. In general, we expect decomposition to produce $\Delta \ll L$ since we would like to have only a reasonably small subset of elements in each range.

The storage complexity is $O(1)$ since we only keep the sum of kernel evaluations in memory. The time complexity to decompose the sequences into their respective element sets is $O(|\mathbf{x}| + |\mathbf{z}|)$, if constant time is taken for each element.

3.6 Manifolds in Higher Dimensions

For convenience, the presentation in this chapter so far assumed that the structure of the input patterns is sequences. However, the idea is not at all limited to sequences since the parametric space T may be arbitrarily chosen. The importance is rather at the interpretation of the manifold of structures in higher dimensions. Just as with sequences, the natural encoding of local features scatter over the surfaces or the volume should be a 2D or 3D manifold, respectively. Depending on the problems at hand, the manifold of the structure may be even higher dimensional.

Chapter 4

On-line Handwritten Character Recognition

In this chapter, I apply the parametric kernel framework to the task of recognizing on-line handwritten characters. This is a different task from the optical character recognition (OCR) which is aimed at the translation of character images into corresponding characters. The goal of the on-line handwritten character recognition, or equivalently, the handwriting recognition is an automatic conversion of text as it is written using pointers such as a digitizing pen or a mouse, where a sensor picks up the pointer position. I adopt a common representation of on-line characters, where an on-line handwritten character is represented as a sequence of strokes and a stroke is represented as a sequence of 2D coordinate of the sampled pointer position $p(t) = (x(t), y(t))$ over time t. A stroke is separated from another by a pen up or mouse release event.

Taking $p(t)$ as the element and the accumulated piece-wise linear distance as the parameter, it is straightforward to construct a parametric kernel function applicable to on-line handwritten character recognition. As the preliminary work, I applied this kernel to handwritten digits recognition and object detection from the laser range finder sensor data. Though I achieved excellent results of about 1% of average false negative rate [51], it still required an objective performance evaluation including the comparison against existing techniques because the results are based on experiments using datasets that are not public. Hence, a public dataset has been used in the main work presented in this chapter.

I experimented with the UNIPEN dataset, which has been widely used in a large volume of research over a decade. This dataset is very difficult to classify since the underlying data sources are highly variable in terms of (1) tablets, (2) drivers and (3) the signal type (e.g. equidistant in time, equidistant in space, non-equidistant). Also, there are labeling and segmentation problems [5]. In conjunction with SVMs, I obtained competitive results in the task of recognizing digits and upper and lower case alphabets.

In section 4.1, I describe the setup of my experimentation, followed by the results in section 4.2. I conclude this chapter with the discussion in section 4.3.

4.1 Experimentation Setup

In this section, I describe the data sets and normalization used in my experiments on the on-line handwritten character recognition, followed by the definitions of

the and the parametric kernel functions for handwritten characters and the learning algorithm.

4.1.1 Handwritten Character Datasets

Handwritten character recognition is often categorized into two classes, i.e. multi-writer and omni-writer recognition. The goal of the multi-writer recognition is to recognize characters written by a set of predefined writers, while that of the omni-writer is to recognize characters written by any writers. Though the omni-writer recognition is desirable, it is in general a more difficult task than the multi-writer recognition as the handwriting variation is significant between different individuals. The dataset used in the preliminary work is limited in terms of the size of the data and the number of writers to support learning multi- or omni-writer recognizers. Therefore, I experimented with the UNIPEN Train_R01/V07 dataset, which has been widely used in a large volume of research over a decade.

The UNIPEN Train_R01/V07 dataset is composed of total 6 categories of isolated characters and 5 categories of isolated words. The examples are voluntarily generated and submitted by hundreds of writers internally. Among the 11 categories, I experimented with 1a (isolated digits), 1b (isolated upper case alphabets), and 1c (isolated lower case alphabets). This is because my goal is to demonstrate the efficacy of the parametric kernel using 1D manifold parameterization, rather than a full scale handwriting recognition. Also, since most research has used only the three categories, it is feasible to make comparison against them over the same

set of data.

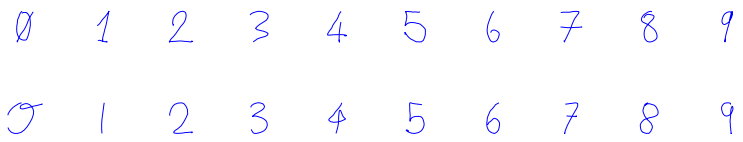
This dataset is, however, very difficult to classify since the underlying data sources are highly variable in terms of (1) tablets, (2) drivers and (3) the signal type (e.g. equidistant in time, equidistant in space, non-equidistant). Also, some examples are corrupt and there are labeling and segmentation errors [5]. Some of the data are lost or unreadable. Especially, I had to deliberately exclude the examples under `tos` directory because the data are corrupt in that either some strokes are missing or noisy strokes are added. See the bad examples in Figure 4.3. The total number of examples and the actual number of examples used in my experiments after removing unreadable or corrupt data from categories 1a, 1b, and 1c of Train_R01/V07 are summarized in Figure 4.1, followed by example characters shown in Figure 4.2.

Category	# of Total Examples	# of Examples Used	Loss Ratio
1a	15953	15404	3.44%
1b	28069	26341	6.16%
1c	61351	59893	2.38%
Total	105373	101638	3.54%

Figure 4.1: Total number of examples are shown in the first column. The number of actual examples used in my experiments after removing corrupt examples under `tos` directory and those with unreadable strokes are shown in the second column. About 3.54% of the overall data in categories 1a, 1b, and 1c are lost. Removing small amount of corrupt, mislabeled, or unreadable examples from Train_R01/V07 has been inevitable and reported by many researchers [45, 59].

Another major problem with UNIPEN dataset is that there is only a train set in Train_R01/V07. In the past, some researchers used a separate dataset called DevTest_R02/V02 as the test set. Unfortunately, this dataset is not publicly available. Therefore, other researchers had to arbitrarily split data in Train_R01/V07 into

1a : Isolated digits



1b : Isolated upper case alphabets

1c : Isolated lower case alphabets

Figure 4.2: Examples of the UNIPEN Train_R01/V07 in categories 1a (isolated digits), 1b (isolated upper case alphabets), and 1c (isolated lower case alphabets)

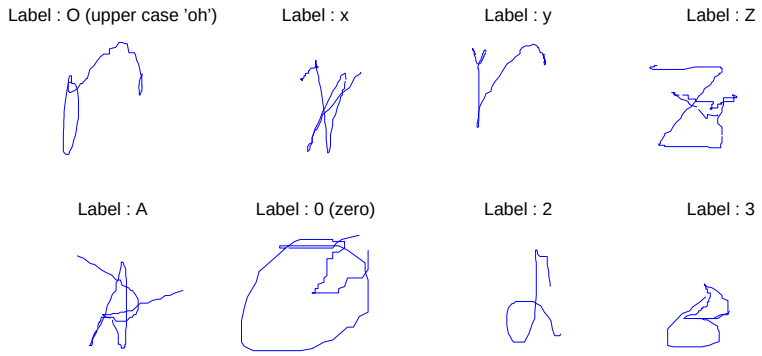


Figure 4.3: Some of the corrupt examples under `tos` directory are shown with their labels. A large portion of the examples are corrupted by added noisy strokes, while the remaining ones are mainly missing strokes, either due to incorrect segment specification or incorrect captures.

train and test sets. Due to the diversity of methods to clean up the data and to split the dataset, it has been very difficult to directly compare recognition rates reported by many other researchers. In this work, I run experiments and compare against existing techniques in a context that is as close to the one provided by Ratzlaff [45].

4.1.2 Character Representation and Normalization

In UNIPEN dataset, a handwritten character $\mathcal{X}$ is represented as a sequence of $|\mathcal{X}|$ strokes, where a stroke X_i is represented as a sequence of $|X_i|$ points $\mathbf{x}_i^k = (x_i^k, y_i^k) \in \mathbb{R}^2$, i.e. $X_i = [\mathbf{x}_i^1, \cdots, \mathbf{x}_i^{|X_i|}]$. Due to the high variability of the data sources in terms of the device and the writer, the characters significantly vary in size and position. To make the system size- and position- invariant, the characters

are normalized by first translating the points by

$$\left(-\frac{\sum_{i=1}^{|\mathcal{X}|} \sum_{k=1}^{|X_i|} x_i^k}{\sum_{i=1}^{|\mathcal{X}|} |X_i|}, -\frac{\sum_{i=1}^{|\mathcal{X}|} \sum_{k=1}^{|X_i|} y_i^k}{\sum_{i=1}^{|\mathcal{X}|} |X_i|} \right) \tag{4.1}$$

so that they are centered at the origin and scaling horizontally and vertically by a uniform scaling factor $h/(\max_{i,k} y_i^k - \min_{i,k} y_i^k)$, so that all characters have equal height h. By default, I used $h = 50$.

4.1.3 Parametric Kernels for Handwritten Characters

In the preliminary work, the parametric kernel function was defined for point sequences [51]. To overcome the aforementioned limitations, I defined parametric kernel functions for segment sequences. The definitions are given below.

Parameterization Let a character $\mathcal{X} = [X_1, \cdots, X_{|\mathcal{X}|}]$ be a sequence of $|\mathcal{X}|$ strokes, where a stroke $X_i = [\mathbf{x}_i^1, \cdots, \mathbf{x}_i^{|X_i|}]$ is a sequence of $|X_i|$ points $\mathbf{x}_i^k \in \mathbb{R}^2$. Then, $\mathbf{x}_i^k$ is associated with a parameter $\boldsymbol{\tau}(\mathbf{x}_i^k)$ in parameter space T as follows

$$\boldsymbol{\tau}(\mathbf{x}_i^k) = \begin{cases} \boldsymbol{\tau}\left(\mathbf{x}_i^1\right) + \sum_{n=2}^{k} \left\|\mathbf{x}_i^n - \mathbf{x}_i^{n-1}\right\| & \text{if } i > 1, k > 1, \\ \sum_{n=2}^{k} \left\|\mathbf{x}_i^n - \mathbf{x}_i^{n-1}\right\| & \text{if } i = 1, k > 1, \\ \boldsymbol{\tau}\left(\mathbf{x}_{i-1}^{|X_{i-1}|}\right) + \left\|\mathbf{x}_i^1 - \mathbf{x}_{i-1}^{|X_{i-1}|}\right\| & \text{if } i > 1, k = 1, \\ 0 & \text{if } i = 1, k = 1, \end{cases} \tag{4.2}$$

which is the piece-wise linear accumulated distance, considering that all strokes are concatenated into a single sequence of points in their order. This parameterization is used in the following parametric kernel definitions.

Parametric kernel for handwritten characters The definition of the parametric kernel function for handwritten characters that is provided in this section is stroke-blind, i.e. each of the characters is represented as a single sequence of points. Characters represented as sequences of strokes are straightforwardly converted into this representation by concatenating all strokes into a single sequence of points in their order. Consider a decomposition of T into N ranges

$$T = \bigcup_{t=0}^{N-1} T_t. \tag{4.3}$$

Given two characters $\mathcal{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_{|\mathcal{X}|}]$ and $\mathcal{Z} = [\mathbf{z}_1, \cdots, \mathbf{z}_{|\mathcal{Z}|}]$, the parametric kernel before normalization is defined as

$$\kappa(\mathcal{X}, \mathcal{Z}) = \sum_{t=0}^{N-1} \langle \phi_t(\mathcal{X}) \cdot \phi_t(\mathcal{Z}) \rangle, \tag{4.4}$$

where

$$\langle \phi_t(\mathcal{X}) \cdot \phi_t(\mathcal{Z}) \rangle = \sum_{\substack{\mathbf{x}_i \in \mathcal{I}_t(\mathcal{X}) \\ \mathbf{z}_j \in \mathcal{I}_t(\mathcal{Z})}} w_{x_i} w_{z_j} \kappa_{\mathrm{PT}}(\mathbf{x}_i, \mathbf{z}_j) \kappa_{\boldsymbol{\tau}}(\boldsymbol{\tau}(\mathbf{x}_i), \boldsymbol{\tau}(\mathbf{z}_j)), \tag{4.5}$$

where $\kappa_{\mathrm{PT}} : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ and $\kappa_{\boldsymbol{\tau}} : T \times T \to \mathbb{R}$ are Mercer kernels that evaluates the similarity between two points in the sequence and their parametric similarity, respectively. The definition of the decomposed element set $\mathcal{I}_t$ and the weighting scheme is identical as in Chapter 3.

To suppress favoring large inputs and to penalize the presence of unmatched

points, (4.4) is normalized by the produce of self similarity of $\mathcal{X}$ and $\mathcal{Z}$:

$$\boldsymbol{\kappa}(\mathcal{X}, \mathcal{Z}) = \frac{\kappa(\mathcal{X}, \mathcal{Z})}{\sqrt{\kappa(\mathcal{X}, \mathcal{X}) \times \kappa(\mathcal{Z}, \mathcal{Z})}}. \tag{4.6}$$

Assuming that a regular overlapping decomposition scheme is used with range length Δ and hop length $\Delta/2$ and that constant time is needed to evaluate κ_{PT} and $\kappa_{\boldsymbol{\tau}}$, the average time complexity of κ is $O(|\mathcal{X}||\mathcal{Z}|\Delta/L)$, where $L = \max(\boldsymbol{\tau}(\mathbf{x}_{|\mathcal{X}|}), \boldsymbol{\tau}(\mathbf{z}_{|\mathcal{Z}|}))$.

4.1.4 Learning Algorithm

I take a kernel-based approach to learn classifiers. Let Σ be the set of total symbols in the dataset. For each symbol $\varsigma \in \Sigma$, a set of ℓ training character examples $X_{\varsigma}^{\text{train}} = \{\mathcal{X}_{\varsigma}^{1}, \cdots, \mathcal{X}_{\varsigma}^{\ell}\}$ are provided. The objective of training is to learn a multi-class classifier $f : \mathcal{X} \mapsto \varsigma \in \{\varnothing\} \cup \Sigma$, that maps an input character $\mathcal{X}$ to the correct label of the character, if it is a symbol in Σ, or a null symbol $\varnothing$, otherwise. In my learning framework, I implement f as a set of $|\Sigma|$ one-versus-all classifiers, $f_{\varsigma} : \mathcal{X} \mapsto \text{sgn}(\vartheta_{\varsigma} - \theta_{\varsigma})$ for all $\varsigma \in \Sigma$, which computes a certainty $\vartheta_{\varsigma} \in \mathbb{R}$ that indicates how certain f_{ς} is that the true label of $\mathcal{X}$ is ς and maps to 1 if $\vartheta_{\varsigma} > \theta_{\varsigma}$, and -1, otherwise, for a threshold $\theta_{\varsigma} \in \mathbb{R}$. The final decision is made such that, for $\Sigma_{+} = \{\varsigma \mid f_{\varsigma}(\mathcal{X}) > 0 \text{ for } \varsigma \in \Sigma\}$,

$$f(\mathcal{X}) = \begin{cases} \underset{\varsigma \in \Sigma_{+}}{\arg\max}\, \vartheta_{\varsigma} & \text{if } |\Sigma_{+}| > 0, \\ \varnothing & \text{otherwise.} \end{cases} \tag{4.7}$$

During testing, f classifies a set of test character examples $\mathcal{X}_n$ for $n = 1, 2, \cdots$, each of which is labeled with the correct symbol $\varsigma_n \in \Sigma$. The classification of f is said to be correct when $f(\mathcal{X}_n) = \varsigma_n$. If either $f(\mathcal{X}_n) \neq \varsigma_n$ or $f(\mathcal{X}_n) = \varnothing$, then the classification of f is said to be incorrect. I use the support vector novelty detection algorithm to learn f_ς.

The main shortcoming of the multi-class classifier in (4.7), which is sometimes called *winner-takes-all* approach, is that it is somewhat heuristic. Each f_ς is trained on different unsupervised learning problem, the certainty values may not be on comparable scales. When more than one f_ς classifies an example as positive, i.e. $|\Sigma_+| > 0$, then their certainty values must be somehow compared to choose one class as the decision. For this, there has been some effort to convert the certainty values into probabilities [48, 52], such as relevance vector machines [41]. Other common approach is to train a binary classifier for every possible pair of classes [30]. This results in $|\Sigma|(|\Sigma| - 1)/2$ binary classifiers. Due to the quadratic increase in the number binary classifiers to train and evaluate during testing, it is often prohibitive for practical purposes. For instance, for a set of upper case alphabet, total $26(26 - 1)/2 = 325$ binary classifiers must be evaluated to classify a single example. An alternative approach is to train and test simultaneously by having a multi-dimensional labels [63]. Unfortunately, optimization is difficult since it has to deal with all support vectors at the same time. Overall, it is fair to say that there is probably no multi-class approach that generally outperforms the others [47]. Therefore, I chose a one-versus-all approach, which allows for fast training and classification with reasonably acceptable results.

Support vector novelty detection Support vector novelty detection (SVND) is an unsupervised learning algorithm for the estimation of novelty of an example. This problem can be described as follows. Given a set of unlabeled examples drawn from an underlying probability distribution P, we estimate a subset $\mathcal{S}$ of the input space such that the probability that a test point drawn from P lies outside of $\mathcal{S}$ equals some a priori specified value between 0 and 1 [47]. This problem is solved by finding the boundary function f which is positive in $\mathcal{S}$ and negative on the complement in the support vector framework as follows.

Suppose we are given a set of unlabeled, i.e. *normal* training examples drawn from an input space $\mathcal{X}$

$$X = \{\mathbf{x}_1, \cdots, \mathbf{x}_\ell\} \subset \mathcal{X}, \tag{4.8}$$

where ℓ is the training set size. For simplicity, suppose $\mathcal{X}$ is a compact subset of $\mathbb{R}^d$ for some dimension d. Consider the following boundary function

$$f(\mathbf{x}) = \text{sgn}\big(\langle \mathbf{w} \cdot \mathbf{x} \rangle - \rho\big), \tag{4.9}$$

where $\mathbf{w} \in \mathbb{R}^d$ is the weight vector and $\rho \in \mathbb{R}$ is the bias. We find $\mathbf{w}$ and ρ by solving the following quadratic problem :

$$\begin{aligned} &\text{minimize } \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\nu\ell}\sum \xi_i - \rho \\ &\text{subject to } \langle \mathbf{w} \cdot \mathbf{x}_i \rangle \geq \rho - \xi_i, \\ &\qquad\qquad \xi_i \geq 0, \text{ for } i = 1, \cdots, \ell, \end{aligned} \tag{4.10}$$

where ξ_i are the slack variables and $\nu \in (0, 1]$ is a control parameter.

The dual of this problem is

$$\begin{aligned} &\text{maximize } \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j\langle \mathbf{x}_i \cdot \mathbf{x}_j\rangle \\ &\text{subject to } 0 \leq \alpha \leq \frac{1}{\nu\ell}, \text{ for } i = 1, \cdots, \ell, \\ &\qquad\qquad \sum_{i=1}^{\ell}\alpha_i = 1, \end{aligned} \tag{4.11}$$

where α_i are the Lagrangian multipliers. If α_i^* solve the dual problem, then the primal variables can be computed as $\mathbf{w} = \sum_{i=1}^{\ell}\alpha_i^*\mathbf{x}_i$ and $\rho = \sum_j \alpha_j^*\langle \mathbf{x}_j \cdot \mathbf{x}_i\rangle$, for any support vector, i.e. $0 < \alpha_i^* < 1/(\nu\ell)$. A non-linear solution could be found by substituting $\langle \mathbf{x}_i \cdot \mathbf{x}_j\rangle$ in (5.17) with a non-linear Mercer kernel. The following statements hold for ν if $\rho \neq 0$.

- ν is an upper bound on the fraction of outliers.
- ν is a lower bound on the fraction of support vectors.

4.2 Results

I begin with presenting the result of the preliminary work in 4.2.1 on the multi-writer digit recognition. I experimented with a handwritten character dataset of digits of which is constructed by a predefined number of writers in a similar manner as the UNIPEN dataset. The results of this preliminary work was very promising [51]. In 4.2.2, I present the results of the main work on the UNIPEN dataset

which is widely used by many researchers for over a decade. I followed the context suggested by Ratzlaff to make comparisons against other existing techniques on the same dataset [45].

4.2.1 Preliminary Work on Digit Recognition

As a proof of concept of the parametric kernels, I designed a parametric kernel for sequences of elements and implemented a handwritten digit recognizer. I used a handwritten character dataset of digits constructed as follows. The train dataset is composed of 200 labeled examples created by two writers, each of whom wrote numeric characters from '0' to '9' ten times. The test dataset is composed of 500 labeled examples created by other authors, 50 for each character. Figure 4.4 shows some training examples for characters '0' to '9' with the number of points in each of them shown below. Characters are normalized to fit the a bounding box of size 300×300 centered at the origin.

Figure 4.4: Examples of handwritten digits

I used a regular overlapping parameter space decomposition scheme shown

in Figure 3.3, with range length $\Delta = 60$ and hop length $\Delta/2 = 30$. I chose κ_{PT} as a radial basis function

$$\kappa_{\mathrm{PT}}(\mathbf{x}, \mathbf{z}) = e^{-\frac{\|\mathbf{x}-\mathbf{z}\|^2}{\gamma\sigma_{\mathrm{PT}}^2}}, \tag{4.12}$$

where $\sigma_{\mathrm{PT}} \in \mathbb{R}$ is the width, $\gamma \in \mathbb{R}$ is the width control parameter, and $\kappa_{\boldsymbol{\tau}}$ as a radial basis function

$$\kappa_{\boldsymbol{\tau}}(\boldsymbol{\tau}(\mathbf{x}), \boldsymbol{\tau}(\mathbf{z})) = e^{-\frac{\|\boldsymbol{\tau}(\mathbf{x})-\boldsymbol{\tau}(\mathbf{z})\|^2}{\gamma_{\boldsymbol{\tau}}\sigma_{\boldsymbol{\tau}}^2}}, \tag{4.13}$$

where $\sigma_{\boldsymbol{\tau}} \in \mathbb{R}$ is the width and $\gamma_{\boldsymbol{\tau}} \in \mathbb{R}$ is the width control parameter. The widths are set to $\sigma_{\mathrm{PT}} = 30$, $\sigma_{\boldsymbol{\tau}} = 30$, the width control parameters are set to $\gamma = \gamma_{\boldsymbol{\tau}} = 1.0$, and the SVND parameter ν is set to $\nu = 0.8$.

The result is shown in Figure 4.5. The classification error is measured as the ratio of incorrectly classified examples to the total examples in the test set. The average error rate is about 1%.

Class	Error	Class	Error
’1’	0.98 %	’6’	1.15 %
’2’	1.12 %	’7’	0.41 %
’3’	0.10 %	’8’	0.23 %
’4’	0.89 %	’9’	0.09 %
’5’	1.01 %	’0’	0.01 %

Figure 4.5: Results of handwritten digit recognition

Though the result is promising to be a proof of concept, it is difficult to objectively compare against other approaches because the results are not based on a

publicly available dataset. However, this work identified a number of important components of the idea including data normalization and parameter space decomposition.

4.2.2 Multi-Writer Character Recognition

Many researchers reported the difficulty in directly comparing the recognition rates of existing techniques using the UNIPEN Train_R01/V07 dataset. Also, There is only a train set in Train_R01/V07. In the past, some researchers used a separate dataset called DevTest_R02/V02 as the test set. Unfortunately, this dataset is not publicly available. Therefore, other researchers had to arbitrarily split data in Train_R01/V07 into train and test sets. Due to the diversity of methods to clean up the data and to split the dataset, it has been very difficult to directly compare recognition rates reported by many other researchers. Here, I run experiments and compare against existing techniques in a context that is as close to the one provided by Ratzlaff [45].

For each category, e.g. 1a, 1b, or 1c, a subset of 10%, 20%, 33%, 50%, 66%, and 90% of the Train_R01/V07 dataset as the train data and the rest as the test data. Following the scheme suggested by Ratzlaff, the characters for each category are drawn in character-by-character, file-by-file order as given in the file list and distributed in sequence into different buckets taking the modulus $N = 10, 5, 3, 2$. For $10\% \sim 50\%$, the subset in the first bucket are used as the train set and the remainder as the test set, while for 66% and 90%, I used $N = 3$ and $N = 10$

and chose the first $N-1$ buckets as the test data. I use the same split scheme in my experiments. The classification error is measured as the ratio of the number of incorrectly classified test characters to total number of characters in the test subset. I used the same learning algorithm and the evaluation metric as in the preliminary work.

The parameter values are determined as follows. Let the *parametric length* $\boldsymbol{\tau}_{\mathcal{X}}$ of a character $\mathcal{X}$ be the parameter of the last point in the last stroke. The average parametric length of characters in a train set $\mathcal{S}_\varsigma = \{\mathcal{X}_\varsigma^1, \cdots, \mathcal{X}_\varsigma^\ell\}$ for a symbol ς is

$$\boldsymbol{\tau}_\varsigma = \frac{1}{\ell} \sum_{n=1}^{\ell} \boldsymbol{\tau}_{\mathcal{X}_\varsigma^n}. \tag{4.14}$$

I set Δ as $\beta\boldsymbol{\tau}_\varsigma$, where $\beta \in \mathbb{R}$ is a control parameter. The default value of β is 0.25. Also, I set the hop length as $\Delta/2$. For two characters $\mathcal{X} = [X_1, \cdots, X_{|\mathcal{X}|}]$ and $\mathcal{X}' = [X'_1, \cdots, X'_{|\mathcal{X}'|}]$, let the average point distance be

$$\sigma^{\mathcal{X}\mathcal{X}'} = \frac{\sum_{t=0}^{N-1} \sum_{\substack{\mathbf{x}\in\mathcal{I}_t(\mathcal{X}) \\ \mathbf{x}'\in\mathcal{I}_t(\mathcal{X}')}} \|\mathbf{x}-\mathbf{x}'\|}{\sum_{t=0}^{N-1} |\mathcal{I}_t(\mathcal{X})||\mathcal{I}_t(\mathcal{X}')|}. \tag{4.15}$$

For the train set $\mathcal{S}_\varsigma$, the width of κ_{PT} for a symbol ς is

$$\sigma_{\mathrm{PT}} = \frac{\sum_{n=1}^{\ell} \sum_{m\neq n}^{\ell} \sigma^{\mathcal{X}_\varsigma^n \mathcal{X}_\varsigma^m}}{\ell(\ell-1)}. \tag{4.16}$$

Similarly, let the average parametric distance be

$$\sigma_{\boldsymbol{\tau}}^{\mathcal{X}\mathcal{X}'} = \frac{\sum_{t=0}^{N-1} \sum_{\substack{\mathbf{x}\in\mathcal{I}_t(\mathcal{X}) \\ \mathbf{x}'\in\mathcal{I}_t(\mathcal{X}')}} \|\boldsymbol{\tau}(\mathbf{x})-\boldsymbol{\tau}(\mathbf{x}')\|}{\sum_{t=0}^{N-1} |\mathcal{I}_t(\mathcal{X})||\mathcal{I}_t(\mathcal{X}')|}. \tag{4.17}$$

For the train set $\mathcal{S}_\varsigma$, the width of $\kappa_{\boldsymbol{\tau}}$ for a symbol ς is

$$\sigma_{\boldsymbol{\tau}} = \frac{\sum_{n=1}^{\ell} \sum_{m \neq n}^{\ell} \sigma_{\boldsymbol{\tau}}^{\mathcal{X}_\varsigma^n \mathcal{X}_\varsigma^m}}{\ell(\ell-1)}. \tag{4.18}$$

The width control parameters are set to $\gamma = \gamma_{\boldsymbol{\tau}} = 0.5$. For SVND, I used $\nu = 0.5$ based on the Schölkopf's observation that a reasonably large ν results in classifiers that do not overfit the data but, at the same time, cover isolated examples in the feature space [47].

For any given portion of the training data used, I achieve the lowest and the highest error rates in category 1a (isolated digits) and 1c (isolated lower case alphabets), respectively. This is not only because 1a has less number of classes but the class boundaries between many of the lower case alphabets are fuzzy, for instance, the cursive writings of 'e' vs. 'l' or 'o' vs. 'c' vs. 'e', or 'f' vs. 'h'. For each category, the error gets lower as more number of examples are used. At about 50% or higher, the error rates get stabilized around the minimum value. The classification errors of multi-writer recognition from tests with varying proportion of the train set used for the three categories 1a, 1b, and 1c are shown in Figure 4.6, 4.7, and 4.8.

For category 1a, the parametric kernels achieve error rates of 3.9% and 3.4%, using 20% and 33% of the training data, respectively. Compared to this, Bahlmann's SVM / GDTW method achieved 4% and 3.8% using 20% and 40% of the training data that are randomly drawn [9]. Using HMM / SDTW [8], Bahlmann achieved 4.5% and 3.2% of the error rates, from 20% and 40% of training data ratio. After removing about 4% of "bad characters", Hu et el. achieved 3.2% of error rate

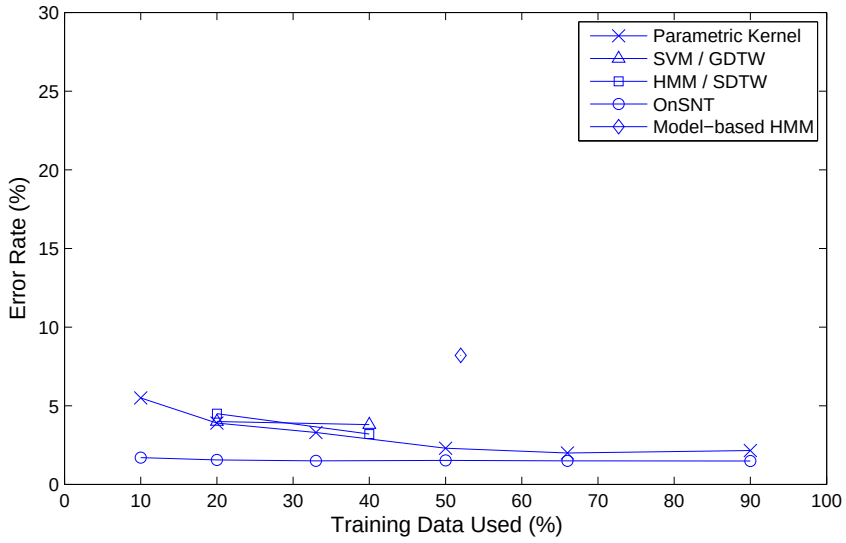


Figure 4.6: Multi-writer recognition error rates for categories 1a (isolated digits)

using HMM [27]. Using the on-line scanning n-tuple (OnSNT) classifier [35, 44], Ratzlaff achieved the best error rates of about 1.5% or less. OnSNT is known as a fast and accurate method for classifying sequences [36]. OnSNT first extracts both static and dynamic features into chain codes and then a sliding window is scanned across the chain code sampling it into n-tuples that become the features (or "addresses") presented to a probabilistic n-tuple classifier. Also, the HMM model based approach of Li et. el achieved 8.2% of error rate, using about 50% of the data. The error rates of my approach is about $2 \sim 2.5\%$ higher than those of OnSNT, if less than half of the training data are used, while it is superior to other approaches. If 50% or more data are used, then the error rate drops to about 2%, which is only about 0.5% more than the best known results.

For category 1b, the parametric kernels achieve error rates of 7.2% and 6.3%,

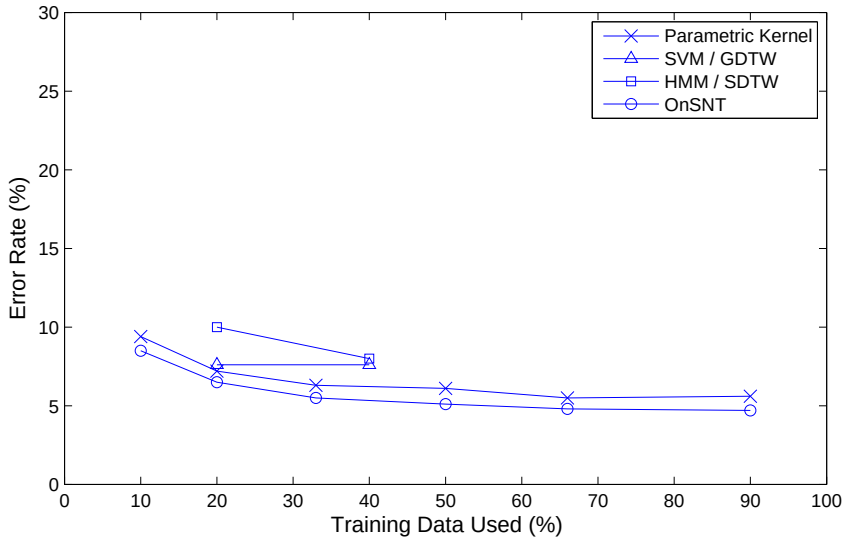


Figure 4.7: Multi-writer recognition error rates for categories 1b (isolated upper case alphabets)

using 20% and 33% of the training data, respectively. Compared to this, SVM / GDTW method achieved 7.6%, using both 20% and 40% of the training data [9]. HMM / SDTW method achieved 10% and 8%, using 20% and 40% of the data, respectively. Vuurpijl's two-stage classification method using hierarchical clustering and applying SVC for misclassified examples showed 6% of error rate [60]. OnSNT achieved the best error rates of about 6.7% and 5.5%, using 20% and 33% of the training data. Also, the HMM model based approach of Li et el. achieved an error rate of 6.4% after removing about 4% of "bad characters". The error rates of my approach is on average about $0.8 \sim 1\%$ higher than the those of OnSNT, for all of the partition ratios used. If more than 50% of the data are used, then the error rate drops to about $6 \sim 7\%$.

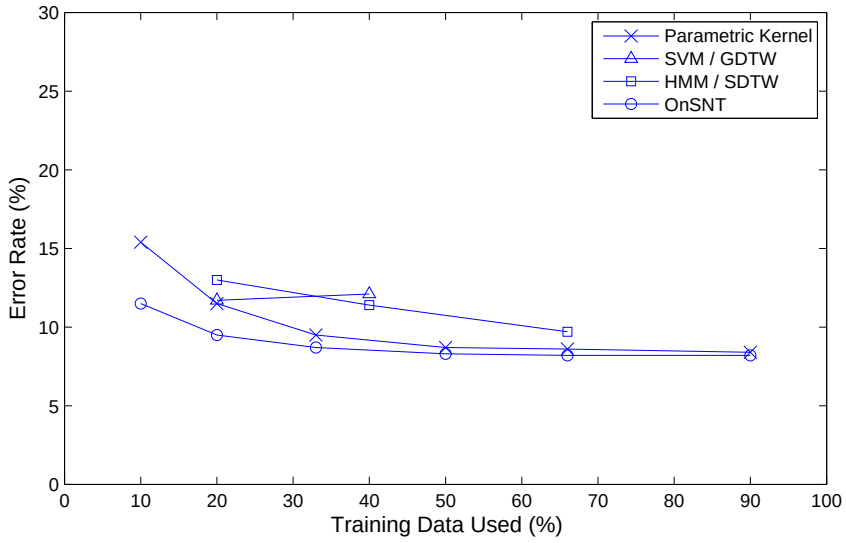


Figure 4.8: Multi-writer recognition error rates for categories 1c (isolated lower case alphabets)

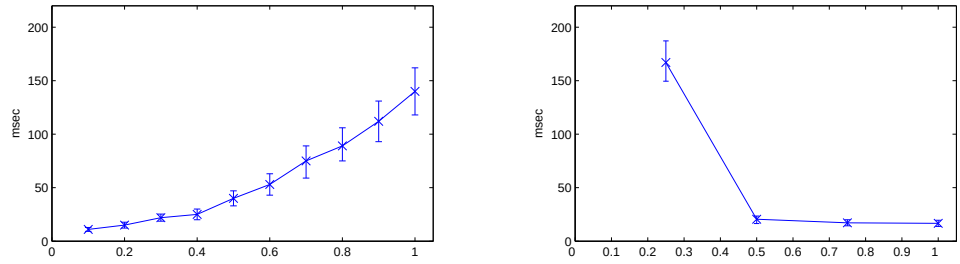
For category 1c, the parametric kernels achieve error rates is $12.2\%, 10.7\%$, and 9.5%, using $20\%, 33\%$, and 50% of the training data, respectively. Compared to this, SVM / GDTW method achieved 11.7% and 12.1% using 20% and 40% of the training data [9]. HMM / SDTW method achieved $13\%, 11.4\%$, and 9.7% of error rates using $10\%, 20\%$, and 66% of the training data, respectively. After removing about 4% of "bad characters", the HMM model based approach of Li et el. achieved 14.1% of error rate. OnSNT achieved the best error rates of about 8.6% using 50% of the training data or more. With only $10 \sim 20\%$ of the training data, the error rates of my approach is about $2 \sim 4\%$ higher than that of OnSNT. However, with 30% of more of the training data, my approach is higher than that of OnSNT only by 1% or less. My apporach shows superior performance to approaches other than

OnSNT, irrespective of the partition ratio.

4.2.3 Computational Complexity

For kernel-based learning techniques, the speed with which classifiers can be learned and used depends greatly on the computational cost to evaluate the kernel functions. This becomes a critical bottleneck when we need to construct kernel matrices for huge datasets. In practice, the time cost therefore has a strong influence on making the decision whether or not to use a new kernel or kernel-based learning technique. Since the time for training and testing classifiers may vary significantly according to the specific algorithm used as well as many other factors, I present an analysis only on the speed of evaluating the parametric kernels for sequences and then compare this against GDTW. For this, I randomly sample 10 examples from each of the ten categories of the UNIPEN category 1a dataset (isolated digits) and measure the time to compute the parametric kernel functions for all of the 100×100 pairs of characters. I measure the average and the variance of the time to compute the similarity metrics implemented in C++ on Linux. I set the range width Δ as $\beta\boldsymbol{\tau}_{\varsigma}$ which is given in (4.14), while setting the hop length as $\alpha\Delta$. I varied β from 0.1 to 1 and α from 0.25 to 1 to show how the kernel computation time varies according to the range size and overlap. A single Intel Core 2 Duo 2.13 GHz with 4MB L-2 cache is used. For comparison, I also implement and compute the time to evaluate GDTW kernel with RBF on the same set of randomly sampled examples.

I also measure the average and the variance of the time to evaluate the



(a) Varying range size (b) Varying hop size

Figure 4.9: In (a), the range size varies according to β, while α is fixed to 0.5. The computation time of the parametric kernel function on sequences shows a roughly quadratic increase as Δ increases according to β. In (b), the hop length varies according to α, while β is fixed to 0.25. Since a larger number of ranges overlap as α gets smaller, i.e. the hop length gets smaller, the computation time increases dramatically.

GDTW kernel with RBF. Specifically, in DTW, the distance matrix is computed first by evaluating RBF at each cell, and the optimal path is searched for using a recursion based on dynamic programming. To profile the cost more accurately, I measured the time taken in matrix computation and recursive path searching separately. On evarage, it took 98.3 ± 5.78 msec to construct the kernel matrix and 174 ± 32.84 msec to recursively search for the optimal path in this matrix.

With parameters set to $\alpha = 0.5$ and $\beta = 0.25$, which are used in the experiments on my online handwritten recognition task, the results show that it is about ten times faster to evaluate parametric kernels than GDTW kernels. The source of computational cost is mainly in the evaluation of element-wise and parameter-wise kernel functions for parametric kernels. In comparison, recursion in GDTW is a significant bottleneck due to context switching. Also, building the distance matrix requires a quadratic order number of kernel evaluations, which may significantly

slow down for long sequences. Though the time complexity is still quadratic, parametric kernels alleviate this via parametric space decomposition. Thus, in practice, parametric kernels are much more efficient to compute.

4.3 Discussion

The parametric kernel framework is applied to the on-line handwritten character recognition problem. Since my primary goal is to demonstrate the efficacy of the parametric kernel using 1D manifold parameterization, rather than to build a full-scale handwritten character recognizer, only the three categories 1a (isolated digits), 1b (isolated upper case alphabets), and 1c (isolated lower case alphabets) of UNIPEN dataset have been used in the experiments. However, with an appropriate scheme to break cursive handwritings in other categories of the UNIPEN dataset, the proposed technique is equally applicable. Since DevTest_R02/V02 test set is not publicly available, I took the scheme of splitting Train_R01/V07 dataset into the train and the test subsets, as suggested by Ratzlaff [45].

For the data three categories, I achieved competitive results compared against the state-of-the-art methods. My approach shows superior performance over most of the approaches compared except for the work using OnSNT [45]. Though OnSNT has shown excellent results for on-line and off-line handwritten character recognition, it suffers from the lack of generality. It is not easy to apply OnSNT to sequences of other types of local vectors than 2D points for handwritten characters since the input sequences must be first converted into sequences of elements

from a discrete space. Their scheme worked well for handwritten characters but it is not straightforward for sequences of other types of elements. My approach is straightforwardly applicable to sequences of any type of elements from a continuous space. Furthermore, the same technique is applicable to inputs represented in any complex structure as long as the manifold of the structure is "parameterizable". My approach is not dependent on any language- or alphabet- specific information, e.g. mathematical symbols, English alphabets, or Tamil scripts, etc., since I treat the characters as just a set of strokes. This technique is thus equally applicable to any type of sequences of ordered elements such as music, voice recordings, or motion capture sequences.

Chapter 5

Online Face Recognition from Video Streams

5.1 Introduction

The previous chapter on online handwritten character recognition focused on learning sequentially structured data using 1D manifold parametrization. In this chapter, I build parametric kernels for online face recognition from video streams based on the parameterization in higher manifolds. I show that the parametric kernel framework is easily applicable to complex structures other than sequences and that the kernel tailoring technique applied to face images represented as unordered sets of local feature vector using their relative positions in the image as the parameters greatly improves the performance over other existing techniques that use the same feature representations but do not adopt the geometric structure between the local

feature vectors.

In many computer vision tasks, an unordered set of local feature vectors has shown to be an effective representation since local feature are more invariant against overall noise or changes in the pose or lighting condition. Usually, such a set comprises descriptors of regions of interest in an input image. In addition, a variety of meta information is also computed, such as the location in pixel coordinates of a region, its size, and its orientation. To learn classifiers from sets of local features, numerous similarity metrics based on *matching* features have been proposed [23]. In general, it is expected that a greater number of local features match between similar images than between dissimilar ones.

This representation has been successful mainly because local features have strong invariance to visual distortions such as pose and illumination variation. At the same time, local approaches have the drawback that two features may be indistinguishable if they are identified within locally similar but unrelated parts of the image. In such cases, the meta information within the image provides a useful guide to alleviate confusion if it contains information about the holistic geometric relationship between local features within the image. We can apply the parametric kernel framework to the task of learning from sets of local feature vectors by taking the meta information as parameters to encode the geometric structure of the local features.

Face recognition is one of the most important computer vision tasks. In the past, much research has focused on recognizing faces from image databases that are constructed in strictly controlled pose and illumination conditions. Many re-

searchers have recently worked on recognizing faces from video streams. Online face recognition from video streams has a broad spectrum of applications ranging from video surveillance camera systems to building robots that can identify and follow humans [4]. This task is often much more challenging than recognizing faces from still shot images because the video streams that they deal with are typically captured in uncontrolled environments with various types of noise. Recognizing faces from video streams is usually performed in the following steps: face detection → feature extraction → training → recognition. The most common factors that make traditional face recognition methods fail in this problem are errors in face detection, strong noise due to varying pose and illumination. In spite of the numerous techniques that have been proposed to overcome these problems, most are either not accurate enough or computationally too demanding for real-time performance.

When sets of local features are used for recognizing faces from video streams, the recognition performance degrades due to the bad matching of local features. Specifically, there are two different cases of bad matching. First, almost no features match between two images of the same person under quite dissimilar illumination conditions due to the qualitative limitation of the features. Second, a large number of locally similar but structurally dissimilar features are matched between images of different people because features are computed locally. I apply the parameter kernel framework to define the similarity metric between two face images using the meta information as the parameter, which encodes the structure explicitly. This approach shows improved performance mainly because a large portion of matched features between face images of different people are locally similar but structurally

dissimilar.

In section 5.2, I describe the setup of my experiments, followed by the results in section 5.3. I conclude this chapter with the discussion in section 5.4.

5.2 Experimental Setup

In this section, I describe the data sets and feature extraction used in my experiments on online face recognition from video streams, followed by the definitions of the similarity metrics and online learning algorithm used. A preliminary version of the face recognizer presented in this chapter has been used during the Robocup 2007 US Open [4]. To demonstrate the efficacy of the proposed method for online face recognition by using meta information as the parameters, I make use of the existing feature extraction technique that is already used for the chosen datasets and show the performance gain when the parametric kernel framework is used.

5.2.1 Facial Video Databases

Face recognition under dynamic pose and lighting condition is still a largely unsolved problem [67]. To address this challenge, I have run online face recognition experiments on two databases of video streams; NRC-IIT [1, 21] and UT Austin Villa [3] facial video databases.

NRC-IIT is a public dataset composed of video streams of 11 individuals captured using a commodity webcam. For each individual, there are two video streams of about $10 \sim 20$ seconds; one for training and the other one for testing.

The video streams total 3,023 and 3,679 images for training and testing, respectively. The resolution is 160×120. The camera position is fixed and the lighting conditions and the background remain unchanged throughout the entire video sequence. The NRC-IIT database is thus most suited for testing the recognition performance with respect to such factors inherent to video-based recognition as low resolution, motion blur, focus, facial expression variation, facial orientation variation, and occlusion [1]. Some of the video frames are shown as samples in Figure 5.6.

To address face recognition under dynamic illumination, I constructed an UT Austin Villa database. This database is composed of video streams of 9 individuals. For each individual, there are 200 training and 400 test images, respectively. To incorporate changes in illumination, the video streams are captured by a webcam that is mounted on top of a mobile robot following the human. Figure 5.1 shows the mobile robot that consists of a Segway RMP, a webcam, a URG-04LX laser rangefinder, and a laptop [4]. The training video streams are captured inside a lab where the light is bright, while the test video streams are captured starting inside the lab and moving to the corridor of a hall where the light is much darker. It is often shot from the back of the human as well. I constructed this database for this research due to the difficulty in finding previously used video streams from the public domain that are suitable for testing recognition performance under dynamic illumination conditions. Some of the video frames are shown as samples in Figure 5.7.

To extract facial features, I first detect the face regions from the video frames

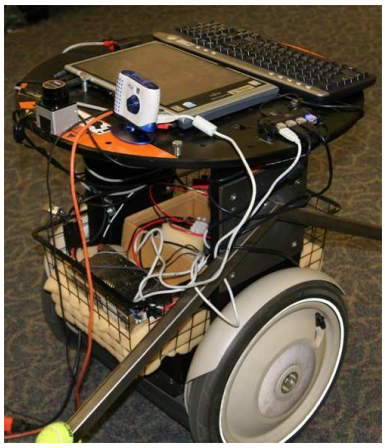


Figure 5.1: The UT Austin Villa facial video database is constructed using the mobile robot shown in this figure. It consists of a Segway RMP, a webcam, a URG-04LX laser rangefinder, and a laptop. Analyzing the input images from the webcam and the distance information from the URG-04LX laser rangefinder, appropriate motion commands are sent to the Segway RMP to turn and to move forward and backwards.

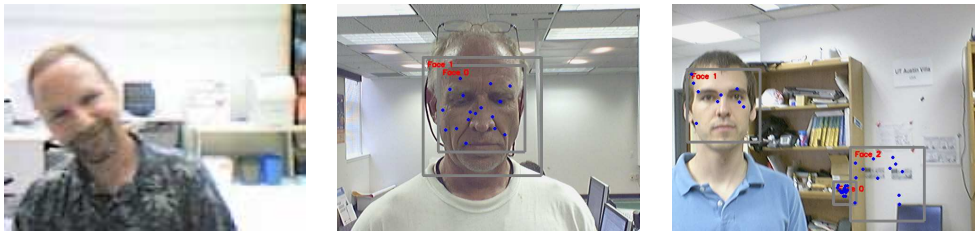
using the default OpenCV implementation of the Haar-like face detector of Viola and Jones [58]. I then use Velaldi's C++ implementation [2] of the Scale-Invariant Feature Transform (SIFT) algorithm originally proposed by Lowe [34]. SIFT locates scale- and rotation- invariant features in an image and computes 128 dimensional descriptor vectors for each feature along with a variety of other information. The result of running the Haar-like face detector and SIFT feature extraction on the set of images from Figure 5.7 is shown in Figure 5.8. The gray rectangles indicate the detected face regions, while the location of the SIFT keypoints are denoted as the blue dots. More detailed screen shots are shown in Figure 5.3.

The Haar-like face detector is in general quite accurate in detecting faces from front. But the accuracy drops rapidly in a number of different cases. First,

Figure 5.2: Examples of face detection using the Haar-like face detector and SIFT feature extraction. The gray rectangles indicate the detected face regions, while the locations of SIFT keypoint are denoted as blue dots. The number and the positions of SIFT keypoints may vary across images of the same person.

it cannot detect faces when the subject turns or tilts the head to the side more than about 45° or so (false negative examples). Second, it finds multiple overlapping regions on a single face. Third, it often returns false positive examples. The proportion of false negatives returned by the Haar-like face detector is relatively small with respect to other sources of error. That is, unless the pose or facial expression vary significantly or a large portion of the face is occluded, it will be detected. Also, detecting a face multiple times should not be a problem if we just take the one with the tightest boundary. But having false positives is fatal because it results in incorrectly trained classifiers, thus making the obtained results less accurate and credible.

I have decided to use the Haar-like face detector for a number of reasons. First, the Haar-like face detector is one of the most efficient algorithms that runs in near real time at about 15 frames per second at about 160×120 resolution. Second, the error is not too significant to affect the face recognition results. This



(a) False Negatives (b) Multiple Detection (c) False Positives

Figure 5.3: Examples of the three major types of error of the Haar-like face detector are shown. False negatives are observed relatively less frequently than the other sources of error. Multiple detections of a single face should not be a problem if we just take the one with the tightest boundary. However, having false positives is fatal because it results in incorrectly trained classifiers, thus making the obtained results less accurate and credible. In (c), the false positive face was reported because the Haar-like face detector confused the two darker colored regions on the wall as eyes.

was also reported in previous research using the same face detector [21, 58] and other part-based face detectors [42,53]. Since this work focuses on the performance measurement of face recognition, this should not be a big problem. Thus, I have manually removed false positives and duplicates from both databases and did not add faces undetected as false negatives. The following table provides the number of detected faces and the number of frames actually contained in each of the video frames.

Though the number of detected faces is significantly smaller than the number of frames contained in some of the video streams, this is not solely due to errors in the face detector used. For instance, there were only 84 faces detected from the NRC-IIT training video for Face 1. This is largely due to other factors such as a face going out of view or going too far away from the camera. Figure 5.5 shows a number of cases where the detector actually failed.

Database	NRC-IIT		UT Austin Villa	
Face	Train	Test	Train	Test
	Detected/Total	Detected/Total	Detected/Total	Detected/Total
0	187/228	177/249	196/200	389/400
1	84/237	103/329	199/200	359/400
2	222/257	231/339	189/200	377/400
3	338/448	300/438	197/200	391/400
4	189/353	281/404	198/200	314/400
5	191/198	208/248	200/200	400/400
6	256/324	256/353	189/200	400/400
7	192/258	208/328	197/200	386/400
8	252/346	386/426	164/200	360/400
9	303/318	257/388	N/A	N/A
10	260/338	281/378	N/A	N/A
fps	20		8	

Figure 5.4: The number of detected faces and the number of frames for each of the video streams used in my experiment.

(a) Subject out of frame (b) Head tilted sideways (c) Head tilted front

Figure 5.5: Examples of video frames where the Haar-like face detector failed to detect the face. Failure in case (a) is not a false negative example, while those in (b) and (c) are.

Summarizing, the Haar-like face detector robustly detects front and upright faces in the scene. Not using the undetected faces may make it difficult to make a fair comparison against other face recognizers using more accurate face detectors on the same data set. At the same time, it is quite difficult to find a common ground for a fair comparison of existing approaches. For face recognition using still shot image databases, a number of protocols such as FERET [18, 40] have been proposed to address this issue. However, no such proposal exists yet for face recognition using video streams. Therefore claims made from performance comparisons against other approaches must be quite conservative in this type of work.

5.2.2 Facial Feature Extraction

I use SIFT as the basis for feature representation of the detected faces. Though SIFT was originally proposed as a method to register images for tasks such as object recognition or stereo image matching, many researcher have recently evaluated it for face recognition [13, 53]. SIFT extracts keypoints at local extrema in the difference of Gaussian scale-space which is produced by applying the cascade filtering over the image with varying scales and taking the difference between neighboring scale images. For a detected keypoint at pixel (i, j) at scale s, the histogram of the gradient over a window of size $n \times n$ around (i, j) is computed. The gradient is computed at each point in the window against 8 different surrounding neighbour directions. The default value for n is 4. The layout of this histogram is concatenated into a $4 \times 4 \times 8 = 128$ dimensional descriptor vector $\mathbf{x}$. Also, the orientation θ

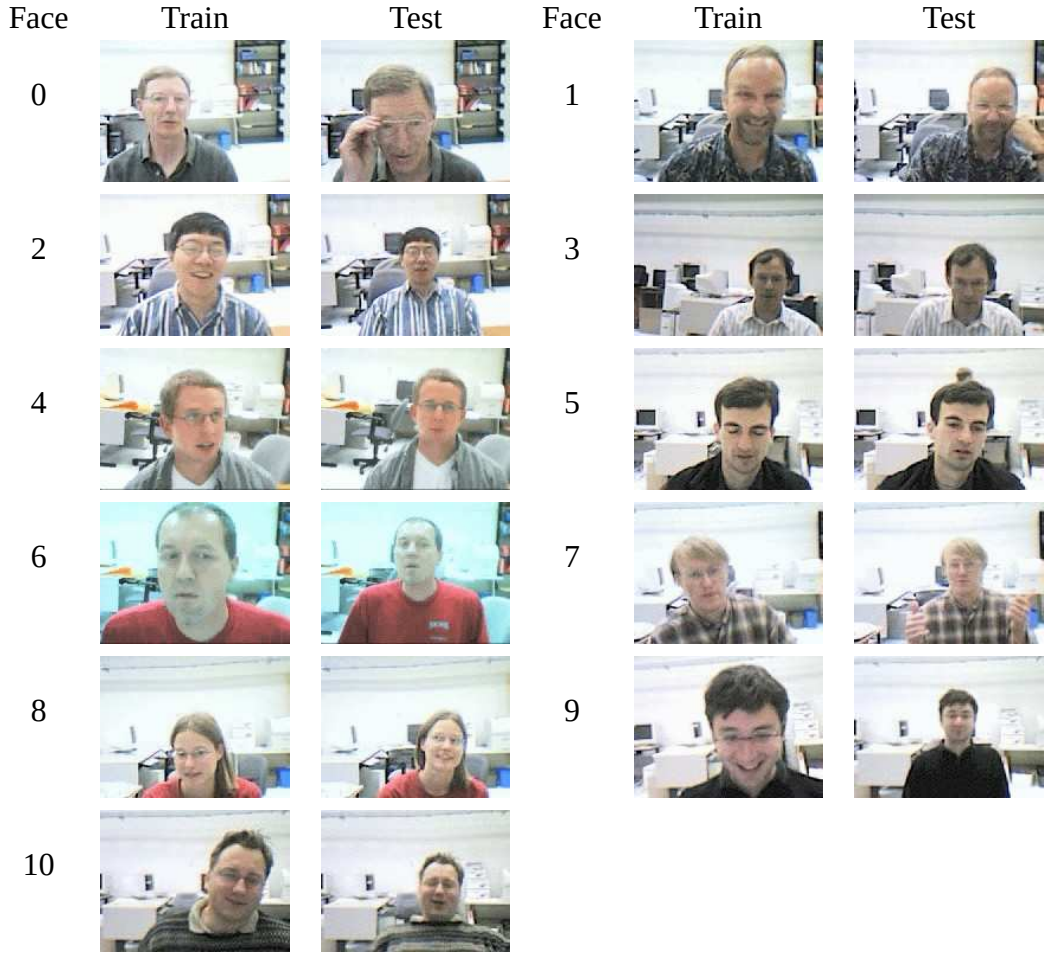


Figure 5.6: Sample images from the NRC-IIT facial video database. Both the training and the test videos for face 0 have been used as queries to the recognizer for *unknown* identity. For face 1 to 10, the training videos are used to learn the initial classifier, while the test videos are used for testing and online learning. The download web page incorrectly showed the face images and the number of frames for Face 3. I used the correct images and numbers of frames here.

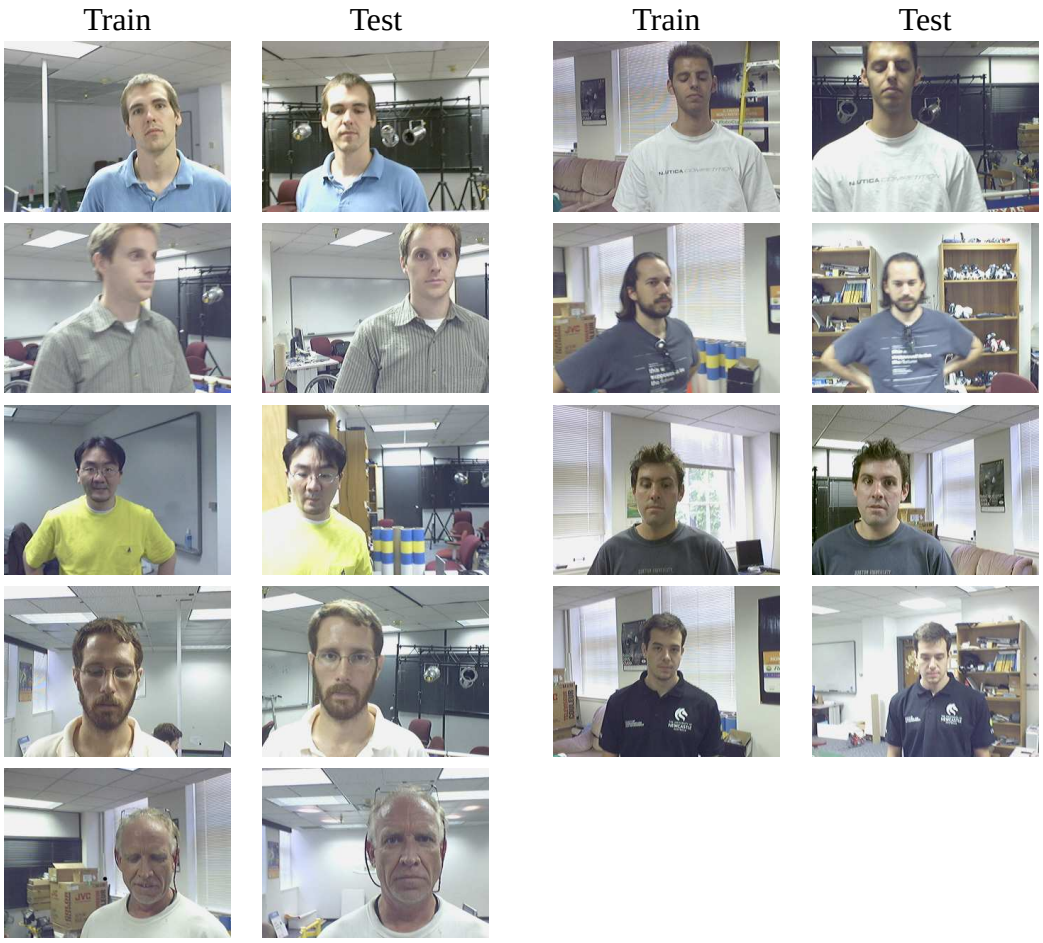


Figure 5.7: Sample images of the UT Austin Villa facial video database. The video streams are captured by a webcam mounted on top of a mobile robot following the human. This results in a strong varation in the illumination conditions. Moreover, the lighting conditions dramatically change in the test video streams as the human walks from inside of the lab to the hallway.

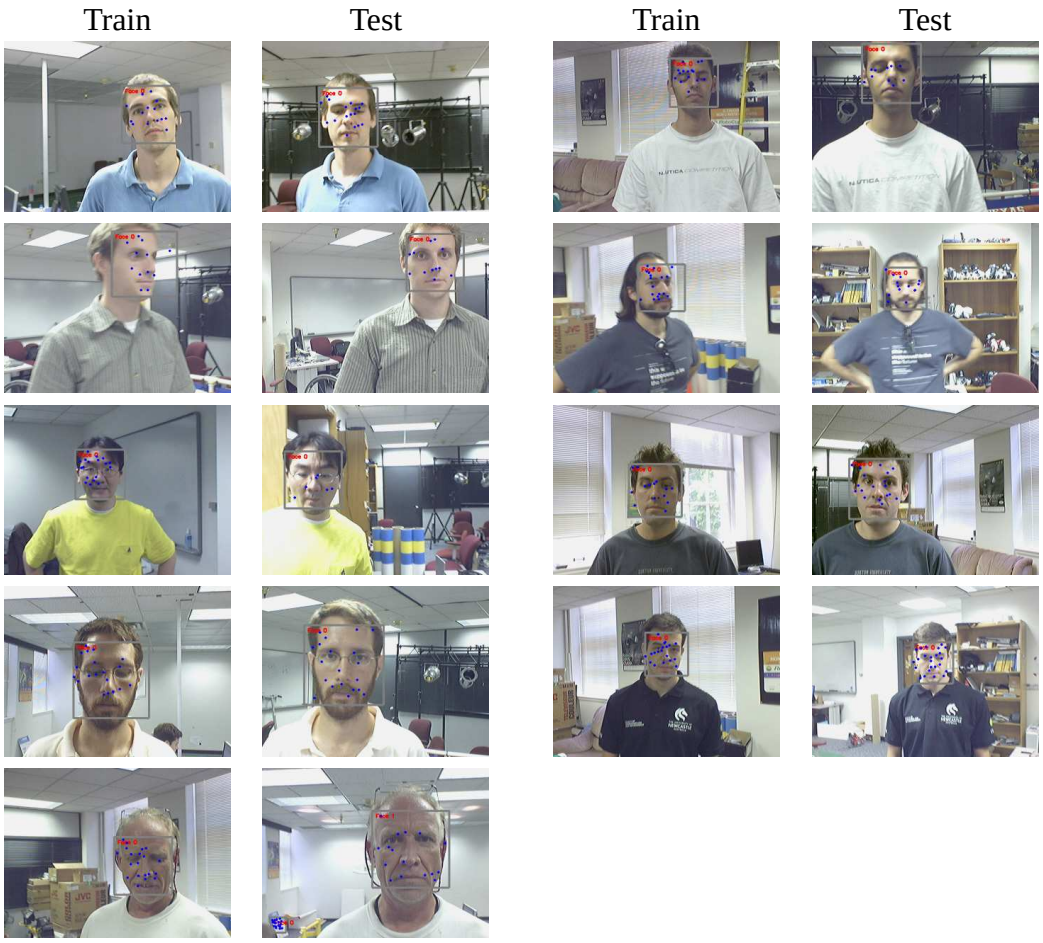


Figure 5.8: Sample images of the UT Austin Villa facial video database showing the detected face regions as gray rectangles and SIFT keypoints as blue dots. The Haar-like face detector scans for face regions in each frame. Each of the detected face regions is converted into grayscale and resized to 24×24 to extract SIFT keypoints. On average, each face contains about $10 \sim 20$ SIFT keypoints.

of the keypoint at (i, j) is obtained as the predominant orientation of the gradient within the window.

To compute the SIFT features for the detected faces, I preprocess the face image as follows. First, I convert the part of the image corresponding to the detected face region into grayscale and scale it to a fixed size of $w \times h$. To reduce the effect of illumination variation, I also apply histogram equalization. Then, I run SIFT feature extraction on the resulting image. Thus, the input face image is transformed into an unordered set of $\mathbf{x}$, each of which is associated with meta information $(i, j), s$, and θ. Figure 5.9 shows examples of the extracted SIFT features on the preprocessed face images.

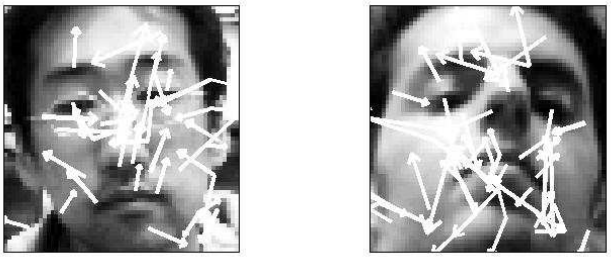


Figure 5.9: SIFT features are extracted from the preprocessed face images. Each of the arrows starts at (x, y) and its length and direction correspond to s and θ of the corresponding SIFT feature.

Once the feature representation is determined, the next step is to define the distance or similarity metric between them. I use four different metrics for matching sets of SIFT descriptors and compare their performance. The first is matching by distance ratio as originally proposed and used by Lowe [34]. The second is a Mercer kernel applied only to the feature space, while the third is the same Mercer

kernel applied to the combined feature vector of SIFT descriptors and the meta information. The last one is the parametric kernel that takes $\mathbf{x}$ as the local element and (i, j) as the parameter. According to the categorization by Grauman [23], the first metric is an example of matching by voting and the second and the third are similar to the pyramid match kernel. The definitions of these metrics are given below.

Similarity based on the distance ratio Let $\mathcal{X} = \{\mathbf{x}_i \mid i = 1, \cdots, n\}$ and $\mathcal{Z} = \{\mathbf{z}_j \mid j = 1, \cdots, m\}$ be the sets of SIFT feature descriptors of two face images. The similarity *from* $\mathcal{X}$ *to* $\mathcal{Z}$ is defined as follows. Considering SIFT descriptors as points in $\mathbb{R}^{128}$, we find the nearest neighbor $\mathbf{z}$ and the second nearest neighbor $\mathbf{z}'$ in $\mathcal{Z}$ to each $\mathbf{x} \in \mathcal{X}$. Then, $\mathbf{x}$ matches $\mathbf{z}$ if

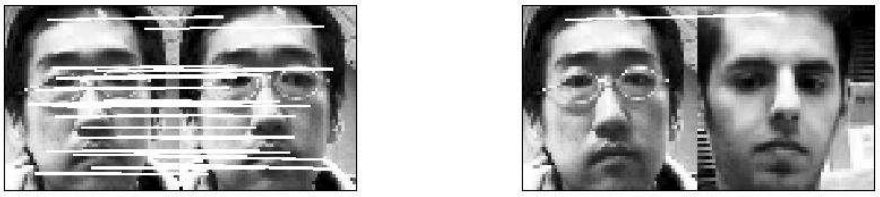
$$\frac{\|\mathbf{x} - \mathbf{z}\|}{\|\mathbf{x} - \mathbf{z}'\|} < \alpha, \tag{5.1}$$

for a predefined ratio threshold α. I use $\alpha = 0.6$ as suggested in [34]. Let $S_{\mathcal{X}\to\mathcal{Z}}$ be the set of matched feature pair $(\mathbf{x}, \mathbf{z})$. The similarity from $\mathcal{Z}$ to $\mathcal{X}$ is defined similarly. The similarity between $\mathcal{X}$ and $\mathcal{Z}$ is defined as the number of matched pairs of SIFT features for both directions,

$$d(\mathcal{X}, \mathcal{Z}) = \big|\{(\mathbf{x}, \mathbf{z}) \mid (\mathbf{x}, \mathbf{z}) \in S_{\mathcal{X}\to\mathcal{Z}} \text{ or } (\mathbf{z}, \mathbf{x}) \in S_{\mathcal{Z}\to\mathcal{X}}\}\big|. \tag{5.2}$$

To handle exceptional cases such as divide-by-zero, it is assumed that $n, m > 1$ and $\mathbf{x}_i \neq \mathbf{x}_j$ for any $i \neq j$. In practice, these assumptions are almost always true. Note that (5.2) is not a valid Mercer kernel because it does not correspond to a norm for an inner product space.

Figure 5.10 shows examples of matching SIFT features between images of the same person in (a) and different people in (b). The locations of matched SIFT features are connected with lines. They clearly show that more SIFT features in general match between images of same person than those of different people.



(a) same person : $d(\mathcal{X}, \mathcal{Z}) = 55$ (b) different peopled : $d(\mathcal{X}, \mathcal{Z}) = 1$

Figure 5.10: The locations of matched SIFT features are connected with a line. More SIFT features match matched between images of same person in (a) than those of different people in (b).

There are two important properties of matching SIFT features descriptors as evidenced by examples in Figure 5.10. First, locations of matched SIFT feature descriptors between images of the same person correspond to similar parts of the face, while those of different people differ significantly. For instance, in Figure 5.10 (b), the descriptor of a SIFT feature located at the right forehead of the left person matched the descriptor of one located at the center of the right person's forehead. This is an instance of a bad match. To exclude such "bad" matches between images of different people, one may apply a geometric verification technique such as the regular grid partitioning scheme introduced in [13]. Second, the locations of SIFT features do not necessarily correspond to facial components such as eyes, nose, and mouth. Nevertheless, SIFT features can robustly characterize faces because they are consistent for faces of the same person. In addition, this lets us avoid expensive

computation for registering face components.

Kernel for sets of SIFT descriptors The second similarity metric is a kernel defined *only* on the sets of SIFT descriptors. For two sets of SIFT descriptors, $\mathcal{X}$ and $\mathcal{Z}$ with $|\mathcal{X}| \leq |\mathcal{Z}|$, the kernel for sets of SIFT descriptors before normalization is defined as

$$k(\mathcal{X}, \mathcal{Z}) = \sum_{i=1}^{|\mathcal{X}|} \kappa_{\text{SIFT}}(\mathbf{x}_i, \mathbf{z}_{\mathbf{x}_i}), \tag{5.3}$$

where $\kappa_{\text{SIFT}} : \mathbb{R}^{128} \times \mathbb{R}^{128} \rightarrow \mathbb{R}$ is a Mercer kernel and $\mathbf{z}_{\mathbf{x}_i} = \underset{\mathbf{z}_j \in \mathcal{Z}}{\arg\min} \|\mathbf{x}_i - \mathbf{z}_j\|$. I chose κ_{SIFT} as a radial basis function

$$\kappa_{\text{SIFT}}(\mathbf{x}, \mathbf{z}) = e^{-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{\gamma \sigma^2}}, \tag{5.4}$$

where $\sigma \in \mathbb{R}$ and $\gamma \in \mathbb{R}$ are the width and the width control parameter, respectively. (5.3) matches SIFT descriptors to their nearest neighbors within the whole set.

To suppress favoring large inputs and to penalize the presence of unmatched SIFT descriptors, (5.3) is normalized by the product of self-similarity of $\mathcal{X}$ and $\mathcal{Z}$:

$$\mathbf{k}(\mathcal{X}, \mathcal{Z}) = \frac{k(\mathcal{X}, \mathcal{Z})}{\sqrt{k(\mathcal{X}, \mathcal{X}) \times k(\mathcal{Z}, \mathcal{Z})}}. \tag{5.5}$$

The complexity of computing (5.3) is $O(|\mathcal{X}||\mathcal{Z}|)$. This is computationally more demanding than PKM, which takes linear time using regular pyramids or sub-linear time using vocabulary-guided pyramids [23]. I did not use PKM here because the computational cost of computing the histogram pyramids is more demanding

than optimal matching based on exhaustive search at the scale of data used in this work.

Kernel for SIFT descriptors and meta information The third similarity metric $\mathbf{k}_{\boldsymbol{\tau}}$ is almost identical to $\mathbf{k}$ except that it is defined on the sets of feature vectors $\mathbf{x}'$, which are constructed by combining a SIFT descriptor $\mathbf{x}$ and the associated meta information $\boldsymbol{\tau}(\mathbf{x}) \in T$. I use (i, j), the location of the SIFT descriptors, as the meta information. Since the only difference between $\mathbf{k}_{\boldsymbol{\tau}}$ and $\mathbf{k}$ is the dimension of the feature vector $(\mathbf{x}, i, j) \in \mathbb{R}^{130}$, the definition of $\mathbf{k}_{\boldsymbol{\tau}}$ is omitted here.

Parametric kernel for SIFT descriptors and meta information The forth similarity metric is a parametric kernel defined for both the SIFT descriptors and their associated meta information. To apply the parameteric kernel framework, I use $\mathbf{x}$ as the local feature and the meta information $\boldsymbol{\tau}(\mathbf{x}) = (i, j)$ as the associated parameter. Recall that the face images are resized to a fixed width w and height h during the preprocessing. The parametric space is thus $T = [1, w] \times [1, h]$. T is decomposed into ranges of size $w/4 \times h/2$ that are horizonally and vertically overlapped by $w/8$ and $h/4$, respectively. This decomposition is inspired by the observations made by Bicego [13] on face recognition with SIFT on the FERET and BANCA face databases. Since T is finite, there are a total of 23 overlapping ranges, as shown in Figure 5.11. For notational consistency with respect to the definition of parametric kernels in Chapter 3, let T_i be the i-th range, for $i = 0, \cdots, 22$, with no favor to any specific ordering.

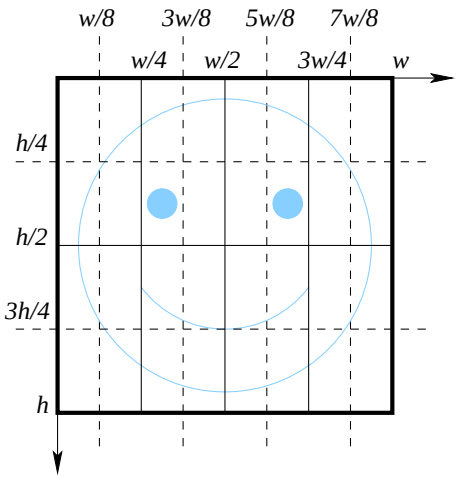


Figure 5.11: Parameter space $T = [1, w] \times [1, h]$ is shown as the thick rectangle that spans the face region. T is decomposed into 23 ranges of size $w/4 \times h/2$ overlapped horizontally and vertically by $w/8$ and $h/4$. For visibility, the borders of the ranges are shown as alternating patterns of dashed and solid lines.

For two sets of SIFT descriptors $\mathcal{X}$ and $\mathcal{Z}$, the parametric kernel before normalization is

$$\kappa(\mathcal{X}, \mathcal{Z}) = \langle \boldsymbol{\phi}(\mathcal{X}) \cdot \boldsymbol{\phi}(\mathcal{Z}) \rangle = \sum_{t=0}^{22} \langle \phi_t(\mathcal{X}) \cdot \phi_t(\mathcal{Z}) \rangle, \tag{5.6}$$

where

$$\langle \phi_t(\mathcal{X}) \cdot \phi_t(\mathcal{Z}) \rangle = \sum_{\substack{\mathbf{x}_i \in \mathcal{I}_t(\mathcal{X}) \\ \mathbf{z}_j \in \mathcal{I}_t(\mathcal{Z})}} w_{\mathbf{x}_i} w_{\mathbf{z}_j} \kappa_{\text{SIFT}}(\mathbf{x}_i, \mathbf{z}_j) \kappa_{\boldsymbol{\tau}}(\boldsymbol{\tau}(\mathbf{x}_i), \boldsymbol{\tau}(\mathbf{z}_j)), \tag{5.7}$$

where κ_{SIFT} is defined in (5.4) and $\kappa_{\boldsymbol{\tau}} : T \times T \rightarrow \mathbb{R}$ is a Mercer kernel that encodes the parametric similarity between two SIFT descriptors. Similar to the parametric kernel definition for sequences, the decomposed element set of $\mathcal{X}$ for T_t is defined

as $\mathcal{I}_t(\mathcal{X}) = \{\mathbf{x}_i | \boldsymbol{\tau}(\mathbf{x}_i) \in T_t\}$. The default value for the weighting factor $w_{\mathbf{x}_i}$ is $1/|\mathcal{T}_{\mathbf{x}_i}|$, where $\mathcal{T}_{\mathbf{x}_i} = \{T_t | \boldsymbol{\tau}(\mathbf{x}_i) \in T_t\}$. I chose $\kappa_{\boldsymbol{\tau}}$ as a radial basis function

$$\kappa_{\boldsymbol{\tau}}(\boldsymbol{\tau}(\mathbf{x}), \boldsymbol{\tau}(\mathbf{z})) = e^{-\frac{\|\boldsymbol{\tau}(\mathbf{x}) - \boldsymbol{\tau}(\mathbf{z})\|^2}{\gamma_{\boldsymbol{\tau}} \sigma_{\boldsymbol{\tau}}^2}}, \tag{5.8}$$

where $\sigma_{\boldsymbol{\tau}} \in \mathbb{R}$ and $\gamma_{\boldsymbol{\tau}} \in \mathbb{R}$ are the width and the width control parameter.

In (5.7), SIFT descriptors match if their parameters fall into the same parameter range.

To suppress favoring large inputs and to penalize the presence of unmatched SIFT descriptors, (5.6) is normalized by the product of self-similarity of $\mathcal{X}$ and $\mathcal{Z}$:

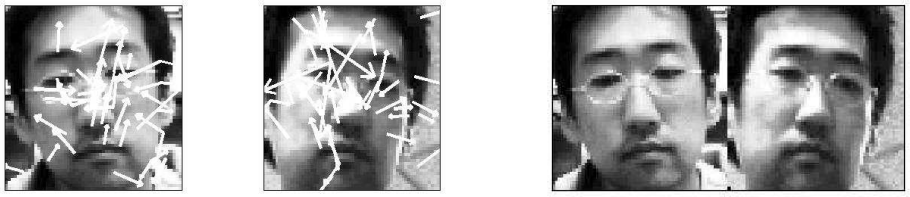
$$\boldsymbol{\kappa}(\mathcal{X}, \mathcal{Z}) = \frac{\kappa(\mathcal{X}, \mathcal{Z})}{\sqrt{\kappa(\mathcal{X}, \mathcal{X}) \times \kappa(\mathcal{Z}, \mathcal{Z})}}. \tag{5.9}$$

5.2.3 Online Learning

In this section, I describe the online learning algorithm used in this experiment. First of all, I investigate the behavior of SIFT features in more detail under illumination changes to show their limitations and motivate the online learning algorithm used in this work.

It is not surprising that the similarity between sets of SIFT features for images of the *same* person may be small if the poses are quite different. However, as shown in Figure 5.12, it may be so even if the pose does not change that much.

Such mismatches are due to the change of illumination when the pose changes rather than merely the change of pose. This is mainly because the changes in illu-

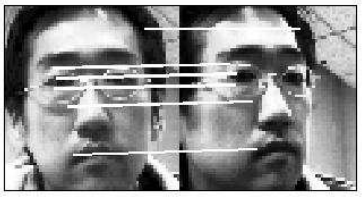


(a) sitting (b) standing (c) match result

Figure 5.12: The subject was sitting in (a) and standing in (b) when the face images were captured. Though the orientations of the face relative to the camera are not siginificantly different, there are no SIFT features matched between (a) and (b) as shown in (c). This is mostly due to the change in the direction of the light.

mination result in significant changes in the facial texture such as shadows or highlights. Figure 5.13 shows examples that manifest this observation. The pose change in Figure 5.13 (a) is not much different from that in Figure 5.12 (c), while there is much less of a pose change in Figure 5.13 (b). However, there are 7 matched SIFT feature descriptors in Figure 5.13 (a), while there are none in Figure 5.13 (b). This type of low similarity between instances of the same person is observed when the light is cast from different sides of the faces, such as in Figure 5.12 (c) and 5.13 (b). SIFT is relatively stable against pose changes but very sensitive to illumination changes. This is inevitable because SIFT is based solely on the brightness of the pixels, which is sensitive to the lighting conditions, and there is no abstraction of the object of interest.

Without complete knowledge of the pose and illumination during training and testing, this type of limitation is inevitable with exisiting techniques. Therefore, I take an online learning approach which learns and adaptively updates classifiers from a series of examples supplied sequentially over time.



(a) pose change

(b) illumination change

Figure 5.13: In (a), the pose varies as the subject stares in different directions, but the lighting conditions remain the same, while in (b), the pose remains stationary but the light is cast from right (left) side of the subject in the left (right) face. There were 7 and 0 matched SIFT feature descriptors in (a) and (b), respectively. This clearly shows that SIFT is very sensitive to the illumination conditions.

Let N be the number of known face classes and, without loss of generality, let i be the labels of a face class C_i for $i = 1, \cdots, N$. For each C_i, a set of ℓ training images $I_i^{\text{train}} = \{\mathbf{i}_i^1, \cdots, \mathbf{i}_i^\ell\}$ is provided. The objective of training is to learn a multi-class classifier $f : \mathbf{i} \mapsto \{0, \cdots, N\}$, that maps an input face image $\mathbf{i}$ to the correct label of the face, if it is a known face, or 0, otherwise. In my learning framework, I implement f as a set of N one-versus-all classifiers, $f_i : \mathbf{i} \mapsto \text{sgn}(\vartheta_i - \theta_i)$ for $i = 1, \cdots, N$, which first computes a certainty $\vartheta_i \in \mathbb{R}$ that indicates how certain f_i is about $\mathbf{i} \in C_i$ and maps to 1 if $\vartheta_i > \theta_i$, and -1, otherwise, for a threshold $\theta_i \in \mathbb{R}$. Since it is the set of SIFT descriptors computed from an input face image that the algorithm eventually deals with, I instead learn $f_i : \mathcal{X} \mapsto \text{sgn}(\vartheta_i - \theta_i)$, where $\mathcal{X}$ is the set of SIFT descriptors computed from $\mathbf{i}$. The final decision is made such that,

for $C_+ = \{k \mid f_k(\mathcal{X}) > 0 \text{ for } k = 1, \cdots, N\}$,

$$f(\mathcal{X}) = \begin{cases} \underset{i \in C_+}{\arg\max}\, \vartheta_i & \text{if } |C_+| > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{5.10}$$

During testing, f classifies a series of test examples $\mathcal{X}_n$ for $n = 1, 2, \cdots$, each of which is labelled with the correct identity $I(\mathcal{X}_n)$. The classification of f is said to be correct when $f(\mathcal{X}_n) = I(\mathcal{X}_n)$, for a test example $\mathcal{X}_n$ of known identity $I(\mathcal{X}_n) \in \{1, \cdots, N\}$, or $f(\mathcal{X}_n) = 0$, if $\mathcal{X}_n$ is the face of an unknown identity. The proposed online learning algorithm updates f during the testing phase to adapt the class boundaries as pose and illumination change over time.

To formally define f, we introduce a number of definitions. Let $X_i^{\text{train}} = \{\mathcal{X}_i^1, \cdots, \mathcal{X}_i^\ell\}$ be the training set for C_i, where $\mathcal{X}_i^k$ is the set of SIFT descriptors computed from the k-th training face image of C_i, and $\sqcap(\mathcal{X}, \mathcal{Z})$ be the similarity metric between two sets of SIFT descriptors $\mathcal{X}$ and $\mathcal{Z}$. Then, f_i is defined as

$$f_i(\mathcal{X}) = \text{sgn}\Big(\sum_{k=1}^{\ell} \alpha_i^k \sqcap (\mathcal{X}_i^k, \mathcal{X}) - \theta_i \Big), \tag{5.11}$$

where $\alpha_i^k \in \mathbb{R}$ is a weighting factor that controls the contribution of the similarity between $\mathcal{X}$ and $\mathcal{X}_i^k$ and θ_i is the threshold value. Learning f, or equivalently f_i, is thus finding the values of α_i^k and θ_i, for $k = 1, \cdots, \ell$ and $i = 1, \cdots, N$.

Actual learning first requires the determination of the similarity metric. In this work, I experiment with the four similarity metrics introduced in the previous section, d, $\mathbf{k}$, $\mathbf{k}_{\boldsymbol{\tau}}$ and $\boldsymbol{\kappa}$. Next, we need a learning algorithm. Independent of the

similarity metric, I use an online learning algorithm that consists of two components : learning the classifier from a static training set and adaptively maintaining the training set during the progress of online classification. Note that the learning is unsupervised since f_i in (5.11) is solely represented using the training examples of C_i. For learning the classifier, I use two different approaches. First, I propose a simple learning algorithm called Equal contribution Certainty cutoff (EC), which runs fast enough to support at least near real-time online learning. In this algorithm, all training examples contribute equally and θ_i is the cutoff value in the certainty distribution for a given error margin. Second, I apply support vector novelty detection (SVND) to demonstrate the behavior of the proposed metrics as the conduits to the kernel-based learning paradigm. Note, however, that the distance-ratio-based metric d is not positive and semi-definite. Since it is not guaranteed that kernel-based learning algorithms, such as support vector machines, based on convex optimization will find a unique optimal solution using d, we do not learn a classifier based on SVND with d. For adaptive maintenance of the training set, the proposed algorithm evaluates a simple criterion for the determination of whether to add the input example to the set of training examples or not using the certainty distribution. In summary, I learn a total of seven different classifiers as shown in Figure 5.14.

The proposed learning algorithm runs in two phases, training and testing. During training, the initial classifier is learned from the initial training set, while during testing, the test set is adaptively maintained. When the training set is updated, the classifier is re-learned with the updated training set. I describe the algorithms for EC and SVND, and adaptive training set maintenance below.

Similarity Metric	Learning Algorithm	
	EC	SVND
d	$f_{d\text{EC}}$	N/A
$\mathbf{k}$	$f_{\mathbf{k}\text{EC}}$	$f_{\mathbf{k}\text{SVND}}$
$\mathbf{k}_{\boldsymbol{\tau}}$	$f_{\mathbf{k}_{\boldsymbol{\tau}}\text{EC}}$	$f_{\mathbf{k}_{\boldsymbol{\tau}}\text{SVND}}$
$\boldsymbol{\kappa}$	$f_{\boldsymbol{\kappa}\text{EC}}$	$f_{\boldsymbol{\kappa}\text{SVND}}$

Figure 5.14: I learn classifiers for the combinations of the learning algorithm and the similarity metric, except for d and SVND since d is not a Mercer kernel.

Equal contribution Certainty cutoff (EC) Learning In EC learning, all training examples contribute to the final outcome equally. That is, $\alpha_i^k = 1$, for all i and k. Therefore, learning f_i in EC is just determining the certainty threshold value θ_i. For this, I introduce a user-specified error rate $\varepsilon_\theta \in [0, 1]$. The idea is to set the value of θ_i such that the probability of incorrectly classifying an unseen example is ε_θ. Finding θ_i can be implemented by first computing the probability distribution of a random variable $\boldsymbol{\vartheta}_i \in \mathbb{R}$, which is the certainty value of an unseen example $\mathcal{X}$ with respect to the training set X_i^{train}, and setting the cutoff value according to ε_θ as θ_i. Unfortunately, it is not possible to compute the true distribution of $\boldsymbol{\vartheta}_i$ using the small number of examples in X_i^{train}. Instead, EC approximates it by computing the the discrete distribution of $\boldsymbol{\vartheta}_i$ from the training examples with respect to X_i^{train} as follows.

Let $p_{\boldsymbol{\vartheta}_i}(\vartheta)$ be the probability mass function that evaluates to the probability that $\boldsymbol{\vartheta}_i$ equals ϑ,

$$p_{\boldsymbol{\vartheta}_i}(\vartheta) = \frac{\left|\{\mathcal{X}_i^n \in X_i^{\text{train}} | \sum_{k=1}^{\ell} \sqcap(\mathcal{X}_i^k, \mathcal{X}_i^n) = \vartheta, \text{ for } n = 1, \cdots, \ell\}\right|}{\ell}. \quad (5.12)$$

EC computes $p_{\boldsymbol{\vartheta}_i}(\vartheta)$ by evaluating the certainty ϑ_i^k of the k-th training example with respect to X_i^{train}. Assume that $\vartheta_i^k \leq \vartheta_i^{k+1}$, for $k = 1, \cdots, \ell - 1$, which can be easily satisfied by rearranging the order of the training examples accordingly. Then, the certainty threshold is determined as $\theta_i = \vartheta_i^K$, where

$$K = \arg\min_{n} \left(\sum_{k=1}^{n} p_{\boldsymbol{\vartheta}_i}(\vartheta_i^k) > \varepsilon_\theta \right). \tag{5.13}$$

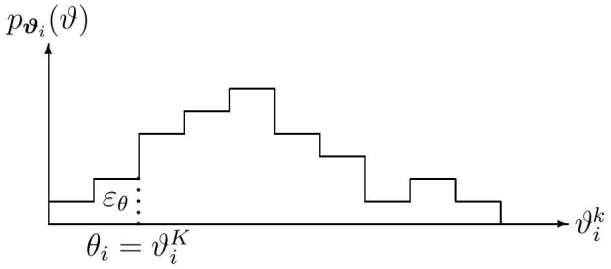


Figure 5.15: The true distribution of the certainty value of an unseen example is approximated by $p_{\boldsymbol{\vartheta}_i}(\vartheta)$. θ_i is determined as the cutoff value ϑ_i^K according to the user-specified error margin ε_θ.

Support Vector Novelty Detection (SVND) Learning Support vector novelty detection (SVND) is an unsupervised learning algorithm for the estimation of the novelty of an example. This problem can be described as follows. Given a set of unlabelled examples drawn from an underlying probability distribution P, we estimate a subset $\mathcal{S}$ of the input space such that the probability that a test point drawn from P lies outside of $\mathcal{S}$ equals some a priori specified value between 0 and 1 [47]. This problem is solved by finding the boundary function f which is positive in $\mathcal{S}$ and negative on the complement in the support vector framework as follows.

Suppose we are given a set of unlabelled, i.e. *normal* training examples drawn from an input space $\mathcal{X}$

$$X = \{\mathbf{x}_1, \cdots, \mathbf{x}_\ell\} \subset \mathcal{X}, \tag{5.14}$$

where ℓ is the training set size. For simplicity, suppose $\mathcal{X}$ is a compact subset of $\mathbb{R}^d$ for some dimension d. Consider the following boundary function

$$f(\mathbf{x}) = \text{sgn}\big(\langle \mathbf{w} \cdot \mathbf{x} \rangle - \rho\big), \tag{5.15}$$

where $\mathbf{w} \in \mathbb{R}^d$ is the weight vector and $\rho \in \mathbb{R}$ is the bias. We find $\mathbf{w}$ and ρ by solving the following quadratic problem :

$$\begin{aligned} &\text{minimize } \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\nu\ell}\sum \xi_i - \rho \\ &\text{subject to } \langle \mathbf{w} \cdot \mathbf{x}_i \rangle \geq \rho - \xi_i, \\ &\qquad\qquad \xi_i \geq 0, \text{ for } i = 1, \cdots, \ell, \end{aligned} \tag{5.16}$$

where ξ_i are the slack variables and $\nu \in (0, 1]$ is a control parameter.

The dual of this problem is

$$\begin{aligned} &\text{maximize } \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle \\ &\text{subject to } 0 \leq \alpha \leq \frac{1}{\nu\ell}, \text{ for } i = 1, \cdots, \ell, \\ &\qquad\qquad \sum_{i=1}^{\ell} \alpha_i = 1, \end{aligned} \tag{5.17}$$

where the α_i are the Lagrangian multipliers. If α_i^* solves the dual problem, then the primal variables can be computed as $\mathbf{w} = \sum_{i=1}^{\ell} \alpha_i^* \mathbf{x}_i$ and $\rho = \sum_j \alpha_j^* \langle \mathbf{x}_j \cdot \mathbf{x}_i \rangle$, for any support vector, i.e. $0 < \alpha_i^* < 1/(\nu\ell)$. A non-linear solution could be found by substituting $\langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle$ in (5.17) with a non-linear Mercer kernel. The following statements hold for ν if $\rho \neq 0$.

- ν is an upper bound on the fraction of outliers.
- ν is a lower bound on the fraction of support vectors.

Since ν controls the upper bound on the fraction of outliers, ν plays a similar role as ε_θ in EC learning. Note that α_i and ρ in (5.17) correspond to α_i^k and θ_i in (5.11). Thus, the solution to (5.17) is actually f_i. Plugging $\mathbf{k}$, $\mathbf{k}_{\boldsymbol{\tau}}$, and $\boldsymbol{\kappa}$ into (5.17) yields $f_{\mathbf{k}\text{SVND}}$, $f_{\mathbf{k}_{\boldsymbol{\tau}}\text{SVND}}$, and $f_{\boldsymbol{\kappa}\text{SVND}}$, respectively.

Online Learning The proposed online algorithm runs in two phases, training and testing. During the training, the initial multi-class classifier f in (5.10) is constructed by learning a one-versus-all classifier f_i from the initial training set for each class C_i. One additional step is needed to finish the training phase, which will be described later. Once the initial learning is finished, the algorithm switches to the testing phase. During the testing phase, the input is a series of test examples provided sequentially over time, $\mathcal{X}_n$, for $n = 1, 2, \cdots$, which are individually classified by f one after another.

Let i_n be the classification result for a test example $\mathcal{X}_n$. In addition to classi-

fying $\mathcal{X}_n$, the algorithm checks whether the certainty ϑ_{i_n} computed by f_{i_n} satisfies

$$\delta_{i_n} < \vartheta_{i_n} < \Delta_{i_n}, \tag{5.18}$$

for threshold values δ_{i_n} and Δ_{i_n}, which are determined according to user-specified error rates ε_δ, $\varepsilon_\Delta \in [0, 1]$ in the same manner as θ_i was determined using $p_{\boldsymbol{\vartheta}_{i_n}}(\vartheta)$. If (5.18) is satisfied, then we first remove a training example from the training set $X_{i_n}^{\text{train}}$ that is the most distant from $\mathcal{X}_n$,

$$\underset{\mathcal{X}_{i_n}^k \in X_{i_n}^{\text{train}}}{\arg \min} \sqcap(\mathcal{X}_{i_n}^k, \mathcal{X}_n) \tag{5.19}$$

and then add $\mathcal{X}_n$ to it. This keeps $X_{i_n}^{\text{train}}$ from growing arbitrarily large. In case of a tie, we remove the oldest one. Finally, we retrain with the updated training set and update δ_{i_n} and Δ_{i_n}. Note that the initial values of δ_i and Δ_i for $i = 1, \cdots, N$, must therefore be computed as the last step of the training phase. The intuition of using the criterion in (5.18) is to add a test example to the training set if f is certain that it is not an outlier but, at the same time, not certain enough to consider it as a trivial example. Hence, in general, the error margins are specified such that that $\varepsilon_\theta < \varepsilon_\delta < \varepsilon_\Delta$. The algorithms for training and testing are described below, where LEARNONEVERSUSALLCLASSIFIER(i) is either EC or SVND for class C_i.

Algorithm 1 Train

for $i = 1$ to N **do**
  $[f_i, \theta_i, \delta_i, \Delta_i] \leftarrow$ LEARNONEVERSUSALLCLASSIFIER(i)
end for

Algorithm 2 Test

for $n = 1, 2, \cdots$ **do**
  $[i, \vartheta] \leftarrow f(\mathcal{X}_n)$
  if $i \neq 0$ **and** $\delta_i < \vartheta < \Delta_i$ **then**
    $\mathcal{X}_{\textbf{min}} \leftarrow \underset{\mathcal{X}_i^k \in X_i^{\textbf{train}}}{\arg\min} \|\mathcal{X}_n - \mathcal{X}_i^k\|$
    $X_i^{\textbf{train}} \leftarrow \{X^{\textbf{train}} - \{\mathcal{X}_{\textbf{min}}\}\} \cup \{\mathcal{X}_n\}$
    $[f_i, \theta_i, \delta_i, \Delta_i] \leftarrow$ LEARNONEVERSUSALLCLASSIFIER(i)
  end if
end for

Summarizing, the online learning algorithm performs an initial training and classifies the stream of test examples iteratively. Certain test examples are added to the training set of the classified class to adapt to the changes of pose and illumination. Also, we remove an example that is the most distant from the newly added example in the hope that it is least similar to the examples that will be observed in the near future. Every time the training set is updated, the corresponding classifier is retrained. In practice, however, we may retrain classifiers much less frequently to save computation if a single update does not result in any significant change in the distribution of the certainty values.

5.3 Results

I begin by presenting the results of the preliminary work in 5.3.1 on the online face recognition developed for the competition at Robocup 2007 [4]. The system was based solely on SIFT features and implemented $f_{d\text{EC}}$ as the face recognizer. The behavioral analysis demonstrates the effectiveness of the system and presents

guidelines for determining the parameter values. At the same time, this also shows its limitations under dynamic illumination.

In the following sections, I present the results of using the parametric kernel framework to overcome this limitation. In 5.3.3, I describe the experiments using a public face database NRC-IIT, which provides an objective comparison with existing techniques. In 5.3.4, I describe the experiments with the UT Austin Villa dataset, which is constructed to exhibit strong variation in the illumination conditions. I show that using the meta information associated with the features in the parametric kernel framework shows the best results among the seven similarity metrics. To clearly demonstrate the advantage of using the parametric kernel method, the experiments are designed to discriminate the contributions by each of the components in my approach.

5.3.1 Face Recognition Under Known Illumination

The goal of the Robocup at Home competition was to build a mobile robot that recognizes faces in real-time. The first task was to learn the faces of a number of people standing in a row during training and then to classify the test faces of a row of a possibly different number and combination of people including unknown faces, standing at the same location. The second task was to learn in the same manner as in the easy task, but the test subjects were standing at locations randomly scattered around the room. At all times the subjects are assumed to face towards the camera.

Towards this goal, we started building the system in the lab environment un-

der two assumptions; 1) the lighting conditions of the lab are similar to those of the real competition field and 2) the flighting conditions in the training phase are similar to those in the test phase, that is, the test lighting conditions are known during the training phase. The lab is equipped with an uneven distribution of light sources, e.g. daylight and light bulbs, where the light cast onto the face creates significantly different patterns of shadow depending on the position and the orientation of the faces of walking subjects.

The system runs in the following stages. First, to construct the initial training set, the robot follows the training subject by tracking the detected face and captures a sequence of ℓ training face images per subject. The initial training set of features is constructed by detecting face images, each of which is scaled to a fixed size, 24×24, and then extracting the SIFT descriptors. Then the system learns the initial classifier. Once the initial classifier is learned, the testing phase begins immediately in a setup identical to that used for training. I used the SIFT descriptor set as the feature set for the detected face images and adopted d and EC as the similarity metric and the learning algorithm, respectively. I implemented tasks 1 and 2 using this face recognizer. The performance was quite satisfactory for the first task. However, I observed a significant drop in the accuracy for the second task. This was mainly because the illumination conditions of the test phase were significantly different from those used in training, thereby violating the second assumption. Nevertheless, our team ended in 2^{nd} place out of 11 teams, since the illumination was fortunately omni-directional in the actual competition. This experience motivated the idea of using meta information in the parametric kernel framework.

I present the result of this system as preliminary work. Though it suffers from the aforementioned limitations, it effectively demonstrates the advantage of the adaptive strategy in EC learning. Also an analysis of its behavior provides useful guidelines for optimizing the parameter values of the learning algorithm.

In the first set of experiments, I give an analysis of the recognition accuracy of EC with respect to the varying parameters. First, I fix the training set size ℓ and measure the true positive and the true negative rates, while varying the error margin ε_θ. There are total four subjects $S_1, \cdots, S_4$; an Asian male, an Asian female, and two Caucasian males. For each subject S_i, a set of $\ell = 100$ training images $\mathcal{S}_i^{\text{train}}$ are collected while the subject moves around the robot at walking speed. After training, $f_{d\text{EC}}$ classifies a set of $n = 300$ face images $\mathcal{S}_i^{\text{test}}$ for each subject in turn and measures the true positive (TP) and the true negative (TN) rates defined as follows :

$$\text{TP} = \frac{1}{4}\sum_{i=1}^{4}\frac{|\{s \in \mathcal{S}_i^{\text{test}} \mid f(s) = i\}|}{n}, \tag{5.20}$$

and

$$\text{TN} = \frac{1}{12}\sum_{i=1}^{4}\sum_{j\neq i}\frac{|\{s \in \mathcal{S}_j^{\text{test}} \mid f(s) \neq i\}|}{n}. \tag{5.21}$$

TP and TN are shown in Figure 5.16. TP starts at 96.37% and drops rapidly as ε_θ gets higher. The true negative rate starts at about 90.12% and soon reaches and remains at almost 100.00% for $\varepsilon_\theta \geq 0.1$. The rapid drop in TP is a direct indication of the fact that the certainties of the majority of test inputs are slightly greater than

the threshold θ_i, which means that the boundary is tight. Also, this means that it is more difficult to correctly classify positive examples than negative ones using our method. This is mainly because EC learning is unsupervised. The most promising value of $\varepsilon_\theta = 0.03$, which is determined such that TP $\doteq$ TN.

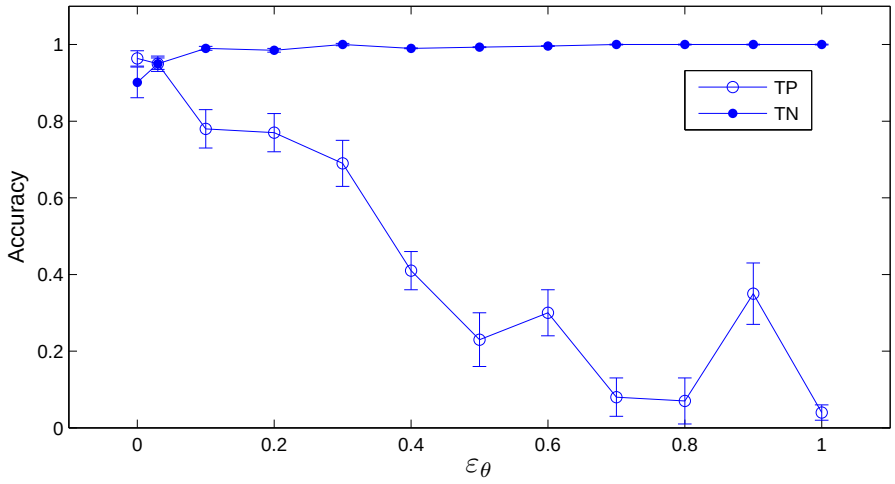


Figure 5.16: True positive and true negative rates for varying ε_θ

In the second experiment, I analyze the speed and the accuracy of the algorithm for varying ℓ. ℓ is used as the control parameter because both the amount of computation to evaluate $f_{d\mathrm{EC}}$ and the accuracy are proportional to it. For this, I measure the average number of frames processed per second and the true positive rate. This time, I train and test with only a single positive subject that moves around while facing the camera. This is to measure the time cost of the classifier that is independent of the number of classes. The result is shown in Figure 5.17. I use $\varepsilon_\rho = 0.03$ and ℓ varies from 10 to 100. We achieve $\geq 95\%$ of true positive rate for $\ell \geq 40$ but the frame rate gradually drops from about 8 Hz to about 5 Hz for

$\ell \geq 70$. The frame rate increases as ℓ gets smaller, but the true positive rate drops below 90% for $\ell \leq 30$. Therefore, we must choose a reasonably but not too large training set. In this experiment, the acceptable range of ℓ is about 40 to 60.

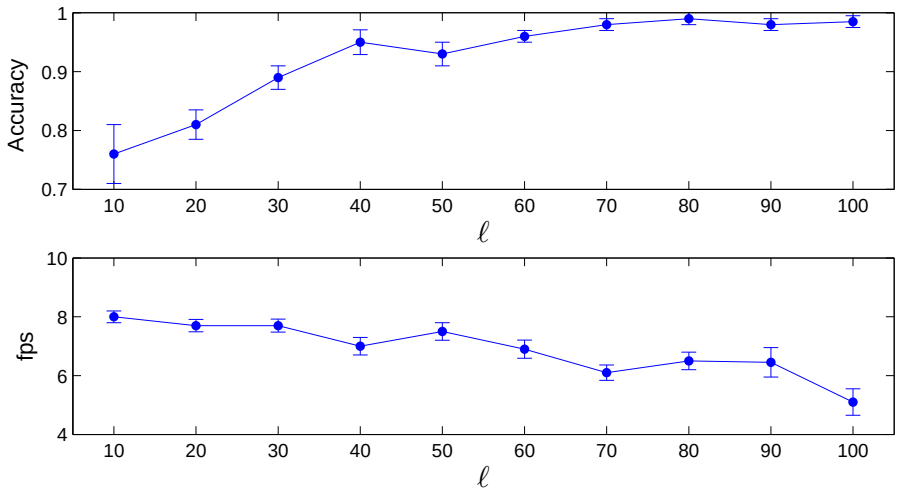


Figure 5.17: Accuracy and speed of the classification for varying ℓ.

In the third experiment, I analyze the effectiveness of the adaptive strategy of EC learning under varying lighting conditions. For this, I measure true positive and true negative rates for the four different cases of turning on(off) the adaptive training set maintenance and moving(fixing) the robot. For each case, I trained with a sequence of $\ell = 80$ training examples. For testing, a sequence of a total of 1000 positive examples followed by 1000 negative examples is presented. Both training and testing examples are captured in the same lighting conditions. Every time a sequence of 50 positive examples is classified, I measure true positive rate. This is repeated 20 times, and the average and the variance of the 20 true positive rates are computed. The following 1000 negative examples are classified in the same

manner, and I compute the average and the variance of the 20 true negative rates. The result is plotted in Figure 5.18. When the robot is moving, the true positive rate increased from $75.2 \pm 7.31\%$ to $97.3 \pm 2.72\%$ and when the robot remained stationary, the increase was from $90.2 \pm 3.32\%$ to $97.0 \pm 1.26\%$. The decrease in true negative rate when the robot was moving (stationary) was $8.5\%(2.0\%)$. These results indicate that the adaptive strategy of EC learning improves the classification accuracy over the cases when it is not used no matter whether the robot is moving or remains still. Meanwhile, we achieve higher true positive and true negative rates when the robot remains stationary than when it moves. This is due to the smaller variation of pose and illumination than in mobile robot platforms.

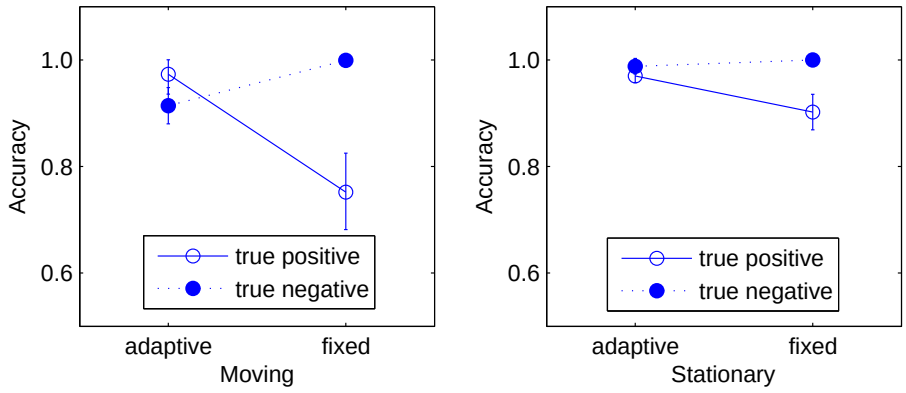


Figure 5.18: The true positive and the true negative rates are measured for the four different cases of turning on(off) the adaptive strategy and moving(fixing) the robot. I used $\varepsilon_\theta = 0.03, \varepsilon_\delta = 0.4, \varepsilon_\Delta = 0.7$, and $\ell = 80$ for all cases.

The results from the first and the second experiments provide useful guidelines to determine the values of ε_θ and ℓ. The adaptive learning strategy shows impressive results if the illumination conditions of the test examples are not signif-

icantly different from those of the training examples. However, it suffers from a rapid drop in accuracy if this condition does not hold.

5.3.2 Evaluation Metrics and Parameter Values

Prior to presenting the main results, I introduce the evaluation metrics used throughout the experiments and the scheme to determine the parameter values. I borrowed the evaluation metrics from Tangelder's work on face recognition using the NRC-IIT facial database [53]. They are discard rate (DR), recognition rate (RR), and false acceptance rate (FAR). If a test face example is classified as unknown, then it is said to be "discarded". Otherwise, it is said to be "recognized" if the test example is of known identity, i.e. $\neq 0$. DR is defined as the ratio of the number of discarded test examples to the total number of test examples of known identity. This definition is slightly different from the original definition by Tangelder in that I define DR over test examples of known identity only, while Tangelder does not clearly differentiate it. Since it is correct (incorrect) to discard test examples of unknown (known) identity, I think my definition makes more sense. For test examples of *known* identity, the recognition rate (RR) is defined as the ratio of correctly classified test examples to the total number of test examples that are not discarded. If a test face example of unknown identity is classified as known, then it is said to be "falsely accepted". The false acceptance rate (FAR) is defined as the ratio of the number of falsely accepted test examples to the total number of test examples of unknown identity. Without loss of generality and following the same experimental setup as Tangelder [53] and

Gorodnichy [21], I use the training and test images of class C_0 as "unknown" test examples.

Since we deal with stored datasets, I do not provide an analysis on the classification speed. But I believe that the preliminary work provides enough understanding of the real-time behavior. At the beginning, I assume that the SIFT features are already precomputed from the detected and preprocessed face regions from the training and the test video streams. The initial training set is constructed using cross validation on randomly selected subsets. During the test phase, examples are provided in the same order as they appear in the original video streams. However, the adaptive strategy of the online learning algorithm depends on the order of the classes of the examples. A test example that is incorrectly classified in the previous step may be added to the training set of an incorrect class. This is unavoidable unless the classification is perfect. To reduce such bias in learning, the classification takes place as follows. For each class, among the test examples that have not yet been classified, pick the earliest one in the order they appear in the test video stream, if any are left. The set of test examples thus collected from the training sets of each class are classified in a random order. Repeat this until no test examples are left.

In my experiments, I use the following scheme to determine the parameter values[1]. ℓ and ε_θ are found in a similar manner as in the preliminary work. ε_δ and ε_Δ are set appropriately to satisfy $\varepsilon_\theta < \varepsilon_\delta < \varepsilon_\Delta$. For SVND, I used $\nu = 0.5$

[1]Parameters can be set freely in a number of different ways including but not limited to the scheme introduced here.

based on Schölkopf's observation that a reasonably large ν results in classifiers that do not overfit the data but, at the same time, cover isolated examples in the feature space [47]. The second set of paramters is the widths in κ_{SIFT} in (5.4) and $\kappa_{\boldsymbol{\tau}}$ in (5.8) and their control parameters. The default values for the width control parameters are $\gamma = 0.5$ and $\gamma_{\boldsymbol{\tau}} = 0.5$. The width is computed differently for each of the classes as follows.

For SIFT descriptor sets $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_{|\mathcal{X}|}\}$ and $\mathcal{X}' = \{\mathbf{x}'_1, \cdots, \mathbf{x}'_{|\mathcal{X}'|}\} \in X_i^{\text{train}}$, with $|\mathcal{X}| \leq |\mathcal{X}'|$, let the average distance between the matched SIFT descriptors using the nearest neighbor matching be

$$\sigma^{\mathcal{X}\mathcal{X}'} = \frac{\sum_{n=1}^{|\mathcal{X}|} \|\mathbf{x}_n - \mathbf{x}'_{\mathbf{x}_n}\|}{|\mathcal{X}|}, \tag{5.22}$$

where $\mathbf{x}'_{\mathbf{x}_n} = \underset{\mathbf{x}'_m \in \mathcal{X}'}{\arg\min} \|\mathbf{x}_n - \mathbf{x}'_m\|$. Then, the width of κ_{SIFT} for class C_i is

$$\sigma_i = \frac{\sum_{n=1}^{\ell} \sum_{m \neq n}^{\ell} \sigma^{\mathcal{X}_n \mathcal{X}_m}}{\ell(\ell - 1)}. \tag{5.23}$$

σ for $\mathbf{k}_{\boldsymbol{\tau}}$ is determined identically, except that the distance is computed between the combined feature vectors $\mathbf{x}' = (\mathbf{x}, i, j) \in \mathbb{R}^{130}$.

σ and $\sigma_{\boldsymbol{\tau}}$ for $\boldsymbol{\kappa}$ are determined as follows. For SIFT descriptor sets $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_{|\mathcal{X}|}\}$ and $\mathcal{X}' = \{\mathbf{x}'_1, \cdots, \mathbf{x}'_{|\mathcal{X}'|}\} \in X_i^{\text{train}}$, with $|\mathcal{X}| \leq |\mathcal{X}'|$, let the average distance between the matched SIFT descriptors be

$$\sigma^{\mathcal{X}\mathcal{X}'} = \frac{\sum_{t=0}^{22} \sum_{\substack{\mathbf{x} \in \mathcal{I}_t(\mathcal{X}) \\ \mathbf{x}' \in \mathcal{I}_t(\mathcal{X}')}} \|\mathbf{x} - \mathbf{x}'\|}{\sum_{t=0}^{22} |\mathcal{I}_t(\mathcal{X})||\mathcal{I}_t(\mathcal{X}')|}. \tag{5.24}$$

Then, the width of κ_{SIFT} for class C_i is

$$\sigma_i = \frac{\sum_{n=1}^{\ell} \sum_{m \neq n}^{\ell} \sigma^{\mathcal{X}_n \mathcal{X}_m}}{\ell(\ell-1)}. \tag{5.25}$$

Similarly, let the average parametric distance between the matched SIFT descriptors be

$$\sigma_{\boldsymbol{\tau}}^{\mathcal{X}\mathcal{X}'} = \frac{\sum_{t=0}^{22} \sum_{\substack{\mathbf{x} \in \mathcal{I}_t(\mathcal{X}) \\ \mathbf{x}' \in \mathcal{I}_t(\mathcal{X}')}} \|\boldsymbol{\tau}(\mathbf{x}) - \boldsymbol{\tau}(\mathbf{x}')\|}{\sum_{t=0}^{22} |\mathcal{I}_t(\mathcal{X})||\mathcal{I}_t(\mathcal{X}')|}. \tag{5.26}$$

Then, the width of $\kappa_{\boldsymbol{\tau}}$ for class C_i is

$$\sigma_{\boldsymbol{\tau}}^{i} = \frac{\sum_{n=1}^{\ell} \sum_{m \neq n}^{\ell} \sigma_{\boldsymbol{\tau}}^{\mathcal{X}_n \mathcal{X}_m}}{\ell(\ell-1)}. \tag{5.27}$$

Although determining the values of these parameters is computationally demanding, this could be performed offline during the preprocessing stage.

5.3.3 Face Recognition Under Steady Illumination

In this section, I present the results of experiments with the NRC-IIT facial database. The illumination remains steady throughout the entire database, while other conditions such as pose, expression, or occlusion vary significantly. A total of seven different similarity metrics in Figure 5.14 are evaluated in terms of the three metrics, RR, DR, and FAR. To isolate the performance gains due to the adaptive strategy of the online learning algorithm from those due to using the parametric kernel, I run two identical classifications for each similarity metric, where the adaptive training

set maintenanace is turned on in one set and off in the other. The parameter values are $\ell = 60, \varepsilon_\theta = 0.03, \varepsilon_\delta = 0.4$, and $\varepsilon_\Delta = 0.7$.

RR, DR, FAR using $f_{d\text{EC}}$, $f_{\mathbf{k}\text{EC}}$, $f_{\mathbf{k}\text{SVND}}$, $f_{\mathbf{k}_\tau\text{EC}}$, $f_{\mathbf{k}_\tau\text{SVND}}$, $f_{\boldsymbol{\kappa}\text{EC}}$, and $f_{\boldsymbol{\kappa}\text{SVND}}$ are showin in Figure 5.19, 5.21, and 5.22, respectively. For each metric, I show blue and red bars that correspond to the peformance metric computed with the adaptive training set maintenance feature of the online learning algorithm turned off (Fixed) and on (Adaptive), respectively.

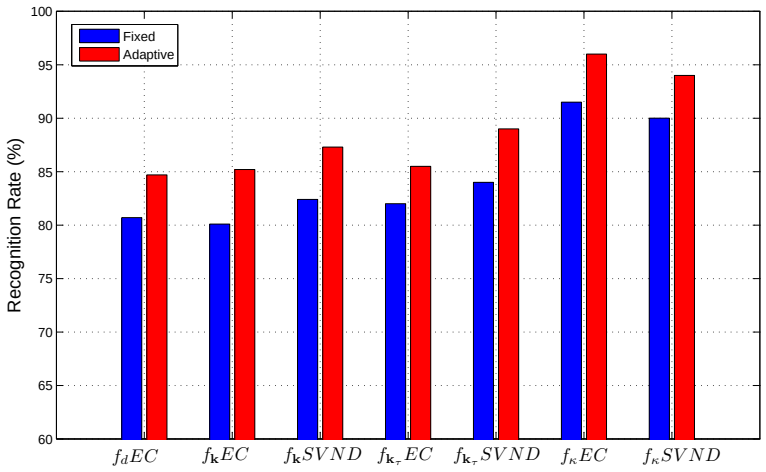


Figure 5.19: The blue and red bars correspond to the peformance metric computed with the adaptive training set maintenance feature of the online learning algorithm turned off (Fixed) and on (Adaptive), respectively. Irrespective of the type of similarity metric used, the adaptive learning strategy increases the recognition rate. Meanwhile, $f_{\boldsymbol{\kappa}\text{EC}}$, along with the adaptive strategy, resulted in the highest RR $= 96.3\%$.

As shown in Figure 5.19, the strategy of adaptively maintaining the training set results in an increase in RR of about $3\% \sim 5\%$, irrespective of the type of similarity metric used. The increase is not as impressive as in the preliminary work.

This is due to the characteristics of the NRC-IIT database, where the lighting conditions are steady throughout the entire set of examples. Using $f_{\kappa\text{EC}}$, I obtain the best results of 96.3% and 91.5% with and without the adative strategy, respectively. This is a competitive result compared to the RR of 94% by Tangelder [53], while RRs for other similarity metrics are about 10% lower. Comparing this against using $f_{\mathbf{k}\text{EC}}$ shows the advantage of using the meta information. In this example, RR increased about 10% on average. It is also intersting to see that $f_{\mathbf{k}_\tau\text{EC}}$ increases RR by at most $3\% \sim 4\%$, which indicates that using meta information in the form of additional dimensions is not a very significant improvement under these condistions. This is because the meta information does not add any significant information if it is used in this manner. See Figure 5.20 which shows the intra- and inter-class histograms of different types of distances between the local feature vectors in the left and the right columns, respectively.

The intra-class histograms show the distances between the matched feature vectors computed from the randomly chosen 150 training and 150 test examples for Face 0, while the inter-class histograms show those computed from randomly chosen 150 training examples of Face 0 and Face 10, respectively. The top row shows the distance distributions between the nearest neighbor SIFT descriptors. The two distributions largely overlap, except that the intra-class distribution has a long left tail. The long left tail corresponds to very close matches in the space of SIFT descriptors between examples of the same class. The second row shows the distance distributions between the nearest neighbor combined feature vector $\mathbf{x}' = (\mathbf{x}, i, j)$. Adding the meta information has the effect of slightly shifting the distributions

to the right hand side. On average, both distributions shift about $+0.1$. The distributions look almost similar to those without the meta information. This partly explains why the increase in RR by $f_{\mathbf{k}_\tau\mathrm{EC}}$ compared to f_{kEC} is small. In addition, combining the local feature vector and the meta information may not be numerically appropriate since they may not be in comparable scale. Without careful numerical adjustment, the meta information will either dominate the effect of local feature vectors or add no extra meaning to them. The paramteric kernel framework solves this problem by computing the normalized similarity between the meta information separately from that between the local feature vectors.

The third row shows the distance distributions between SIFT descriptors matched using the parametric kernel framework. Decomposing the parameter space into smaller overlapping regions and enforcing matching between features that co-occur in the same range prevents features far apart in the feature space from matching. At the same time, the co-occurring features may not be the nearest neighbors. Consequently, a large portion of the matched feature vectors for the historgrams shown in the top two rows disappear, while the newly matched feature vectors are sub-optimal, that is, further apart. There are 44700 matches in the histograms of top two rows, while there are only 21735 and 14621 matches in the left and the right histograms as shown in the third row, respectively. This indicates that the matched local features are likely to be computed from far apart locations within the face, if the match is between images of different people rather than a single person. Also, the sub-optimal matching explains the increase of the mean distances in the third row. It is interesting to see that the long left tail still remains in the intra-class dis-

tance distribution for the parametric kernel. This means that in intra-class matching, many SIFT descriptors that are close in the feature space also co-occur in the same parameter range, while there are almost none in inter-class matching. By favoring these matches, the parametric kernel framework shows superior performance over other similarity metrics.

The DRs are computed with ε_θ for which the the RRs are computed as shown in Figure 5.19. DRs decrease if the adaptive strategy is used, except for $f_{\kappa\text{SVND}}$. Compared to the DRs in Tangelder's work (26%), I achieve much smaller DRs. My classifiers are empirically shown to correctly discard negative examples much better than correctly recognizing positive examples. I believe that this is closely related to the unsupervised nature of learning one-versus-all classifiers in the proposed framework. In general, the lack of knowledge of the distribution of negative examples tends to make the class boundary fuzzy and thus, more examples are classified as positive. This also explains in part why the RRs are mostly lower than 94% in Tangelder's work.

FARs for the seven similarity metrics are shown in Figure 5.22. Compared to Tangelder's result using BHG (4%), the FARs of my approach are about twice as high, but much less than those using SIFT (18%). In addition, the adaptive strategy does not help that much in lowering the FARs. This is mainly because examples of unknown identity are rarely added to any of the training sets, even though they are classified as positive by some of the one-versus-all classifiers.

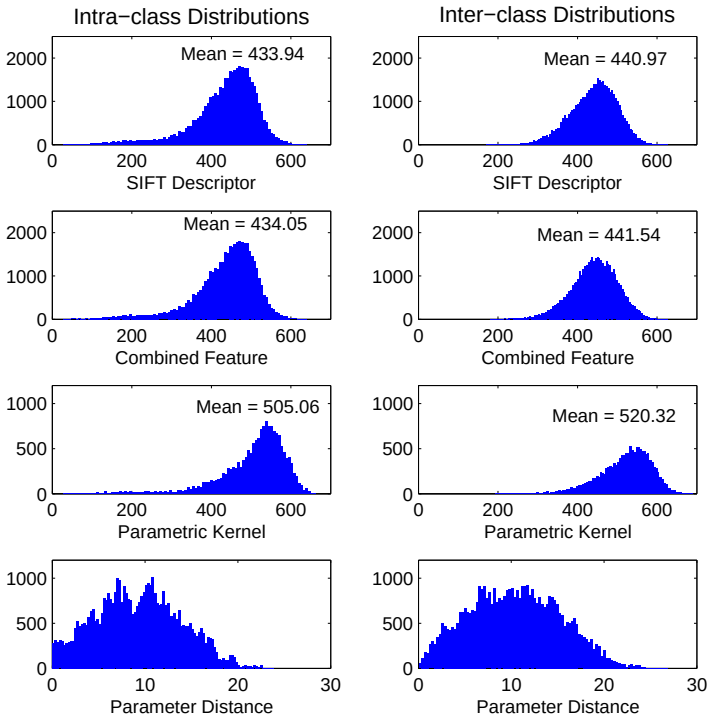


Figure 5.20: The intra- and inter-class distance distributions are shown in the left and the right columns, respectively. The top row shows the distributions of distances between matched SIFT descriptors, while the second row shows that of the combined feature vector $\mathbf{x}' = (\mathbf{x}, i, j)$. The distribution remains almost indentical, which means that using the meta information in the form of extra dimensions does not make much difference. The histograms in the third row show that matching in the parametric kernel framework is sub-optimal in that the modes shift to the right hand side and a much smaller number of features match. Meanwhile, the long left tail still remains in the intra-class distance distribution for the parametric kernel, which is missing in the inter-class distance distribution. This means that, in intra-class matching, many SIFT descriptors that are close in the feature space also co-occur in the same parameter range. By favoring these matches, the parametric kernel framework shows superior performance over other similarity metrics. The last row shows the distribution of parameter distances between the nearest neighbor SIFT descriptors. The intra-class histogram shows strong concentration around $[0, 5]$, which is missing in the inter-class histogram. This implies the importance of using parameter distances.

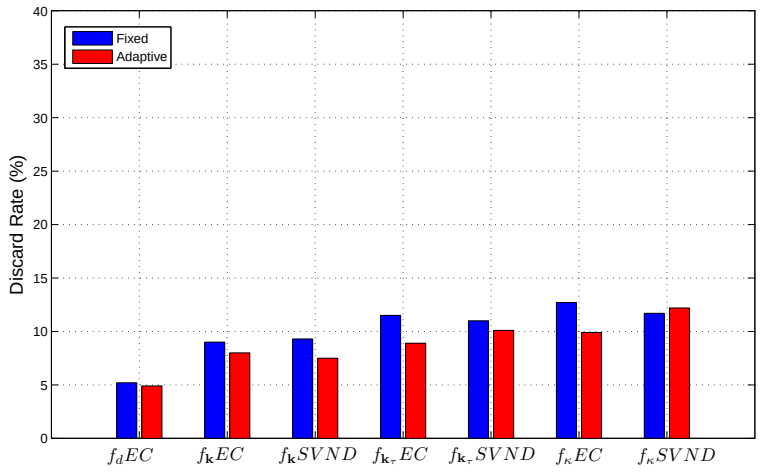


Figure 5.21: ε_θ for which the RRs are computed in 5.19 are used to compute the DRs. Except for $f_{\kappa\text{SVND}}$, DRs decrease if the adaptive strategy is used. The DRs are much smaller than that of Tangelder's approach 26%. However, I believe this is because the boundaries of the one-versus-all classifiers in my approach are loose since no information about the distribution of negative examples is used.

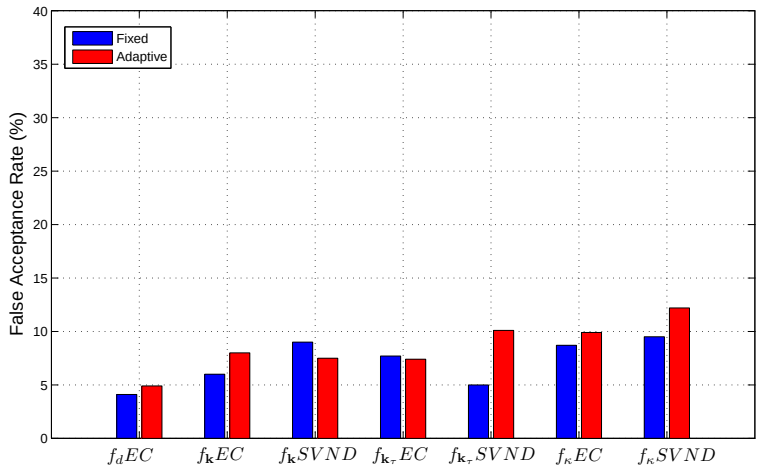


Figure 5.22: FARs are about twice as high as those of Tangelder's approach using BHG (4%) but much less than using SIFT (18%). The adaptive strategy does not help that much in lowering the FARs. This is mainly because examples of unknown identity are rarely added to any of the training sets, even though they are classified as positive by some of the one-versus-all classifiers.

5.3.4 Face Recognition Under Dynamic Illumination

In this section, I present the results of experiments with the UT Austin Villa facial database. The lighting conditions of the test examples are significantly different from those of the training examples and unknown a priori, which makes the problem very difficult to solve. In particular, SIFT is very sensitive to the lighting conditions, as explained in 5.2.3. The experiments are set up much as in the work with the NRC-IIT database. The parameter values are $\ell = 80, \varepsilon_\theta = 0.05, \varepsilon_\delta = 0.3$, and $\varepsilon_\Delta = 0.7$.

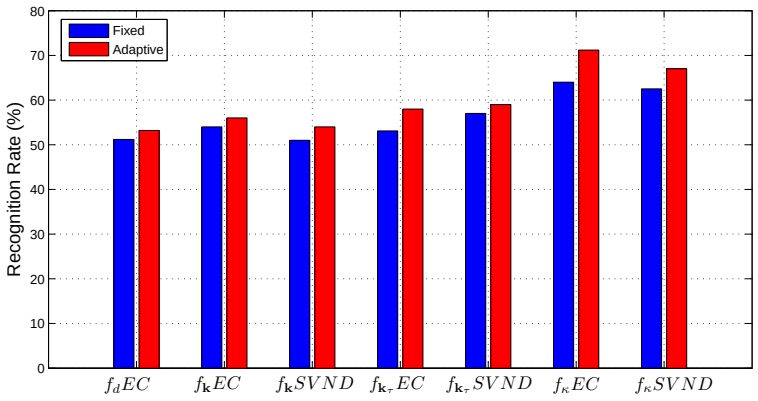


Figure 5.23: $f_{\boldsymbol{\kappa}\text{EC}}$, along with the adaptive strategy, resulted in a maximum RR of 71.2%, while $f_{\boldsymbol{\kappa}\text{SVND}}$ resulted in a maximum RR of 65.2%. Though the adaptive strategy is an important factor in increasing the RRs, we can still see that the parametric kernel framework effectively utilizes the meta information.

As shown in Figure 5.23, adaptive training set maintenance increases the RRs, irrespective of the type of similarity metric used. $f_{\boldsymbol{\kappa}\text{EC}}$, along with the adaptive strategy, resulted in a maximum RR of 71.2%, while $f_{\boldsymbol{\kappa}\text{SVND}}$ resulted in a maximum RR of 65.2% without the adaptive strategy.

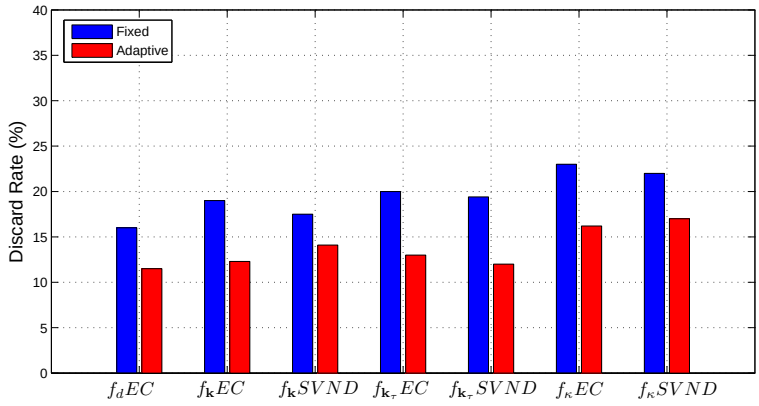


Figure 5.24: ε_θ for which the RRs are computed in 5.19 are used to compute DRs. $f_{\kappa\text{EC}}$ and $f_{\kappa\text{SVND}}$ resulted in maximum DRs of 17.2% and 23%, with and without the adaptive strategy, respectively.

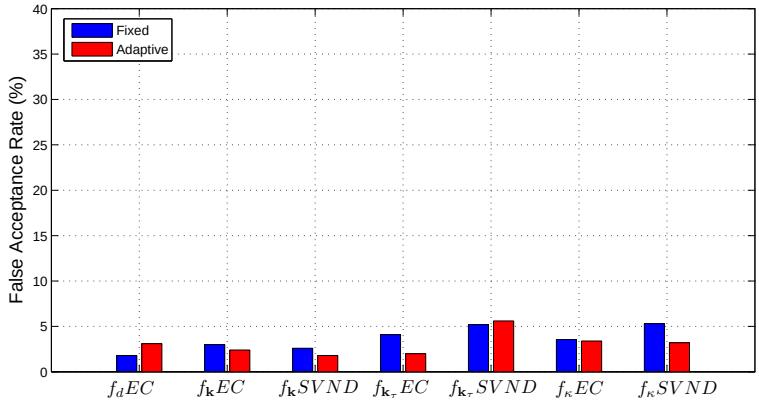


Figure 5.25: $f_{\kappa\text{EC}}$, which showed the highest RR, resulted in a FAR of 3.4%.

Compared to the results for the NRC-IIT data, the best RR for UT Austin Villa is about 25% lower. I think that this is impressive considering the strong sensitivity of SIFT features to illumination variations and the unknown test lighting conditions during the training phase. Though the adaptive strategy consistently increases the RRs, the increase is smaller than for the NRC-IIT data. This is mainly due to the lower certainty values during testing. Lowering ε_δ and ε_Δ may allow more examples to be added to the training set, but this also admits more false positive examples.

5.4 Conclusion

The parametric kernel framework has been applied to the problem of online face recognition from video streams as part of effort for the Robocup at Home 2007 competition. Recognizing faces from online video streams has recently been much studied due to the applicability to a wide spectrum of problem domains. Compared to traditional face recognition based on single-shot face image databases such as FERET or BANCA, recognizing faces from video streams is a much more challenging problem due to strong variance in pose, expression, occlusion, and illumination. Especially changes in illumination make the problem very difficult to solve.

Representing an image as a set of local feature vectors has been effective in many computer vision tasks. This is particularly advantageous if the global representation of a given image changes considerably due to variations in pose, view points, occlusions, or other types of deformations. The similarity between two im-

ages could then be computed by matching their local features. In this work, I used sets of SIFT descriptors as the feature representation for the detected faces. Unfortunately, using local features does not work that well under significant changes in the illumination. This resulted in a large number of matches between features located at quite irrelevant regions within the face images.

Compared to images from other vision applications such as object categorization or image retrieval, the global representations of face images change much less dramatically. I take advantage of this characteristic by adopting the geometric relationship between the locations of local features as the parameter in the parameteric kernel framework to suppress matching features computed from irrelevant regions. However, this strategy reduces bad matches but does not boost good ones. To handle this, I proposed an online strategy that maintains the training set adaptively by adding examples during the test phase, which is used in combination with the parameteric kernels.

I demonstrated the efficacy of this approach through experiments on two datasets. On the NRC-IIT facial database, I achieved a RR of 96.3% at a FAR of 9.9% using the parametric kernel and EC-learning. I compared this to the state-of-the-art results using the same dataset produced by Tangelder [53]. Using BHG as the feature, his approach achieved a RR of 94% at a FAR of 4%. With SIFT, his approach achieved a RR of 95% but the DR and the FAR are 47% and 18%. My approach shows a superior RR at the cost of a higher FAR compared to the result using BHG. However, compared to the results using SIFT, I obtain much smaller DR and FAR. Considering that he computes SIFT features after extracting fiducial

points from the face image, including nose, eyes, and mouth, and that he constructs the training set optimally using a greedy selection scheme, my results look quite impressive.

The second sets of experiments with the UT Austin Villa facial database are aimed at performance analysis under considerable amount of variation in the illumination, since the illumination was steady in the NRC-IIT data. In particular, the lighting conditions during the test phase are quite different from those during the training phase and unknown a priori. I achieved a RR of 71.2% at a FAR of 3.4%. Though I cannot provide any comparison since the UT Austin Villa data is a private dataset, I believe that my results are strong considering the significant amount of variability in illumination.

Summarizing, I have applied parameteric kernels and an adaptive online learning strategy to the face recognition task using the locations of local features as the parameters. The objective was to demonstrate the general applicability of the parametric kernel framework to structures other than sequences. The only change required is extending the parameter space to a higher dimensional manifold and applying an appropriate decomposition scheme within this space. This shows the potential of the parametric kernel framework to provide a systemmatic approach to learning structured data.

Chapter 6

Sensor Data Analysis

In this chapter, I apply the parametric kernel framework using 1D parameterization to the task of object detection from the sensor data captured by laser range finders. The work demonstrates that the parameteric kernel framework for handwritten character recognition is directly usable in detecting objects from sensor data. This is because the input patterns from both problem domains are sequentially structured.

Laser range finders are often used in robot systems to sense the nearby environment. This is done by measuring the distance to the nearest object omnidirectionally in 2D plane at a certain frame rate. Analyzing this information, robots take appropriate actions to achieve the goal such as localization, avoiding obstacles, or approaching the destination. At each frame, the scanned scene returned by laser range finders is represented as an array or a sequence of distance values, where each dimension is associated with a certain angle of the direction that the distance was measured. The first step to analyze a scene is to remove min and max values from

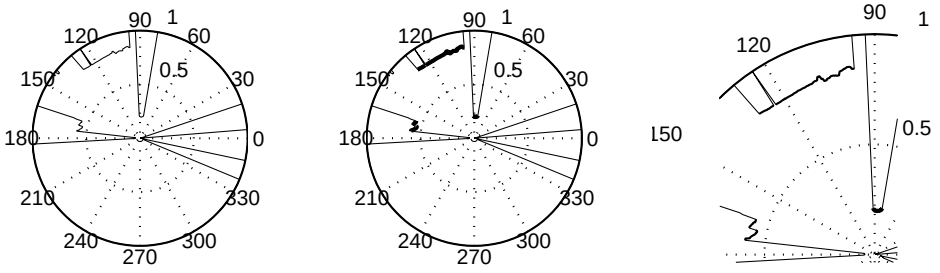
the sequence and segment the remaining values into a set of meaningful subregions called blobs. Object detection in this context is to task of finding a blob that corresponds to the surface of the target object. If the robot or the target object is moving and/or due to the noise and error of the device, the blobs for the same target object look similar but are all different in terms of the sequence lengths and the distance values. The parametric kernel framework with 1D manifold parameterization introduced in Chapter 4 is directly applicable to this task with minor difference in the data normalization scheme and the parameter values. I implemented a soccer ball detector to demonstrate the efficacy and the generality of the proposed approach.

In section 6.1, I describe the setup of my experimentation, followed by the results in section 6.2. I conclude this chapter with the discussion in section 6.2.1.

6.1 Experimental Setup

6.1.1 Data Representation and Normalization

I used Hokuyo URG-04LX laser range finder which scans at 10 frames per second in 2D plane. It is mounted on the front side of the Segway RMP robot that navigates in the lab. The objective is to locate a region in a frame of sensor data that corresponds to a soccer ball. A frame is represented as a sequence of 768 distance values measured from -120° to 120° relative to the front direction. Each distance is measured in millimeters, ranging from 20 to 4095 with maximum 1% error. See Figure 6.1 (a) for a snapshot.



(a) Raw Input (b) Segmented (c) Detected

Figure 6.1: Raw sensor input is shown in (a) and thick curves in (b) are the blobs after segmentation. In (c), the thick round blob at angle about 90° and distance 0.2 is detected as the soccer ball.

Each frame is preprocessed as follows. First, a frame is normalize by dividing it by the maximum distance value and segmenting into several meaningful subregions called blobs. A blob is defined as a subregion of a frame where all distance values in the subregion are in $(\theta, 1)$ and every consecutive values are at most δ apart. In this experiment, I used $\theta = 0.05$ (about 20cm) and $\delta = 0.02$ (about 8cm), which are found after a number of trials. Excluding values less than θ is necessary to remove noise due to parts of the robot that are at the proximity of the sensor. Excluding value of 1 is also necessary to remove the empty space. Blobs are further normalized by scaling so that the min and max distance values in each blob are 0 and 1.

Blobs that corresponds to the soccer ball are roughly semi-circles because Hokuyo URG-04LX scans from only one side of it. However, since this is also the case for all round objects such as human legs or beacons, they will confuse the

classifier. Without any extra information, there is no way to tell one from another. For demonstration purposes, therefore, it is assumed that no such confusing objects exist in the frame. Some of the ball and non-ball examples are shown in Figure 6.2.

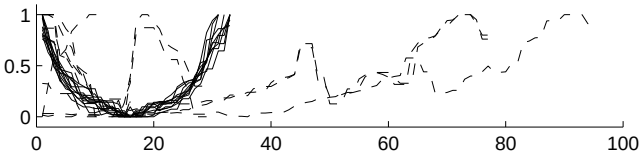


Figure 6.2: Solid (dashed) curves are ball (non ball) examples, where vertical and horizontal axis correspond to the normalized distance and the number of points in blobs. Clearly, ball blobs are semi-circular, while others are irregular.

6.1.2 Parametric Kernels for Blobs

Let $\mathcal{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_{|\mathcal{X}|}]$ be a blob, where $\mathbf{x}_i \in (\theta, 1)$ are the normalized distance values. The parameter for $\mathbf{x}_i$ is defined as

$$\boldsymbol{\tau}(\mathbf{x}_i) = \begin{cases} \sum_{k=2}^{i} \|\mathbf{x}_k - \mathbf{x}_{k-1}\| & \text{if } i > 1, \\ 0 & \text{otherwise.} \end{cases} \tag{6.1}$$

Consider a decomposition of parameter space T into N ranges

$$T = \bigcup_{t=0}^{N-1} T_t. \tag{6.2}$$

Given two blobs $\mathcal{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_{|\mathcal{X}|}]$ and $\mathcal{Z} = [\mathbf{z}_1, \cdots, \mathbf{z}_{|\mathcal{Z}|}]$, the parametric

kernel for blobs before normalization is defined as

$$\kappa(\mathcal{X}, \mathcal{Z}) = \sum_{t=0}^{N-1} \langle \phi(\mathcal{X}) \cdot \phi(\mathcal{Z}) \rangle, \tag{6.3}$$

where

$$\langle \phi(\mathcal{X}) \cdot \phi(\mathcal{Z}) \rangle = \sum_{\substack{\mathbf{x}_i \in \mathcal{I}_t(\mathcal{X}) \\ \mathbf{z}_j \in \mathcal{I}_t(\mathcal{Z})}} w_{x_i} w_{z_j} \kappa_{\text{BD}}(\mathbf{x}_i, \mathbf{z}_j) \kappa_{\boldsymbol{\tau}}(\boldsymbol{\tau}(\mathbf{x}_i), \boldsymbol{\tau}(\mathbf{z}_j)), \tag{6.4}$$

where $\kappa_{\text{BD}} : (\theta, 1) \times (\theta, 1) \rightarrow \mathbb{R}$ and $\kappa_{\boldsymbol{\tau}} : T \times T \rightarrow \mathbb{R}$ are Mercer kernels that evaluates the similarity between two distance values in the blobs and their parametric similarity, respectively. The definition of the decomposed element set $\mathcal{I}_t$ and the weighting scheme is identical as in Chapter 3.

To suppress favoring large inputs and to penalize the presence of unmatched points, (6.3) is normalized by the produce of self similarity of $\mathcal{X}$ and $\mathcal{Z}$:

$$\boldsymbol{\kappa}(\mathcal{X}, \mathcal{Z}) = \frac{\kappa(\mathcal{X}, \mathcal{Z})}{\sqrt{\kappa(\mathcal{X}, \mathcal{X}) \times \kappa(\mathcal{Z}, \mathcal{Z})}}. \tag{6.5}$$

6.2 Results

The laser range finder is mounted at about 20cm from the ground on the front side of the mobile robot. The robot is controlled to slowly navigate around the field with a number of objects including the soccer ball, boxes, and walls. The frames are captured at about fps for about 30 seconds. After preprocessing each of the frames into normalized blobs, each of the blobs are manually labeled as either +1 if

it is a ball, or -1, otherwise. The dataset $\mathcal{S}$ thusly constructed is composed of total 442 ball blobs and 2199 non-ball blobs. For learning, $\mathcal{S}$ is split into train and test subsets. The train subset is constructed by randomly drawing 20 ball blobs and 22 non-ball blobs from $\mathcal{S}$ and the remainder is the test subset. I trained a soft-margin support vector classifier (SVC) with a quadratic loss function and $\boldsymbol{\kappa}$ in (6.5) as the kernel function. The details of SVC has been described in Chapter 2. See [47, 49] for an introduction to SVCs.

I used a regular overlapping parameter space decomposition scheme as in the handwritten character recognition with the range length $\Delta = 0.1$ and the hop length $\Delta/2 = 0.05$. I chose κ_{BD} as a radial basis function

$$\kappa_{\text{BD}}(\mathbf{x}, \mathbf{z}) = e^{-\frac{\|\mathbf{x}-\mathbf{z}\|^2}{\sigma_{\text{BD}}^2}} \tag{6.6}$$

where $\sigma_{\text{BD}} \in \mathbb{R}$ is the width, and $\kappa_{\boldsymbol{\tau}}$ as a radial basis function

$$\kappa_{\boldsymbol{\tau}}(\boldsymbol{\tau}(\mathbf{x}), \boldsymbol{\tau}(\mathbf{z})) = e^{-\frac{\|\boldsymbol{\tau}(\mathbf{x})-\boldsymbol{\tau}(\mathbf{z})\|^2}{\sigma_{\boldsymbol{\tau}}^2}}, \tag{6.7}$$

where $\sigma_{\boldsymbol{\tau}} \in \mathbb{R}$ is the width. The widths are set to $\sigma_{\text{BD}} = \sigma_{\boldsymbol{\tau}} = 0.1$. The SVC parameter C is set to 1000. For a blob $\mathcal{X}$ labeled as $y \in \{-1, 1\}$, the classification of the learned classifier f is correct if $f(\mathcal{X}) = y$, or incorrect, if $f(\mathcal{X}) \neq y$.

The classification error is measured as the ratio of the incorrectly classified test blobs to the total number of the test blobs. On average, the classification error was 0.82% with 78.6% of the training data being the support vectors. Thus, with only about 1.2% of the total data, I achieved more than 99% of error rate. Each

scene on average contained about 10 to 20 blobs, among which only one corresponds to a soccer ball. Our C++ implementation running on a tablet PC with Intel Pentium M 1.2 GHz processor on average required less than 0.1 seconds to classify blobs in each scene.

6.2.1 Discussion

A parametric kernel function is defined for blobs. It has been used in SVC to learn a blob classifier for the task of object detection from laser range finder sensor data. With a few minor modification including the data normalization and other parameter values, the parametric kernel functions for handwritten character recognition using a 1D parameterization was directly applicable to this task without resort to any heuristic feature extraction. This scheme provides a more systematic, flexible, and intuitive way to build effective similarity measures that could be used in conjunction with kernel machines. However, more work needs to be done in terms of data normalization if the shape of the target object is more irregular and complex than just the balls. Since the ball was round, it is scale- and rotation- invariant. A possible approach for affine-invariant data normalization is the shape context proposed by Belongie [11].

Chapter 7

Conclusion and Future Work

Input patterns in a variety of robotics and HCI learning tasks are geometrically structured. Making computers recognize the similarities between physical patterns is an extremely difficult task due to significant amount of noise and change in the physical conditions against which humans can reliably learn physical patterns. To cope with this difficulty, a set of local features are extracted from parts of the input patterns that are locally invariant against such noise or change in the physical conditions.

Unfortunately, such representation does not fit conventional learning algorithms and distance metrics. They assume fixed dimensional vector inputs but the each physical pattern consists of a variable number of local features. It is not easy to represent the irregular structure into uniform length vectors either. Kernel functions provide a flexible solution to this problem. However, neither defining such kernels for a given type of structured inputs nor adopting existing kernels for other problem

domains is straightforward.

The work in this dissertation presents a solution to overcome these limitations, which is generally applicable to a diversity of geometric structures. I explore the concept of *tailoring kernels* to address the problem of defining the distance metrics between sets of geometrically related local features. The geometric structure between the local features is embedded into the notion of *kernel parameters*.

7.1 Contributions

We can summarize the major contributions of this dissertation as follows.

- **Synthesizing Kernels**

  The parametric kernel framework provides a method to systematically synthesize customized kernel functions for structured inputs by aggregating kernel functions for the local features. The fundamental idea is based on the concept of convolution kernels proposed by Haussler, which was one of the first instances of kernel functions on structured data [16, 25, 33]. However, in general, finding the definition of substructures and their "part-of" relationship with the composite objects for a specific problem is quite difficult [6]. The parametric kernel framework provides an important step to overcome this limitation by providing a general framework to group local features into substructures based on their geometric relationship.

- **Structure Embedding via Parameterization**

  Representing complex geometric structure of local feature vectors in a form that could directly be used as a distance metric suited to many conventional learning algorithms is a difficult task. The parametric kernel framework provides the ability to intuitively encode the geometric structure underlying the local feature vectors of physical patterns via *parameterization*. This scheme imposes a manifold to represent the geometric structure and associates each of the local feature vectors to a point in this manifold. As the proof of concept, I synthesized and applied parametric kernels for handwritten character recognition and sensor data analysis by parameterization in a 1D manifold, and face recognition by parameterization in a 2D manifold, respectively. I achieved competitive results on these tasks.

- **General Applicability**

  Traditional approaches to handling structured data extract application-specific feature vectors from the input patterns. Therefore, a feature representation that is effective for inputs with a certain structure is often heuristic and not generally applicable to inputs from other problem domains with similar structures. Since the parametric kernel function is not dependent on any specific context of certain problem domains, it is generally applicable to inputs from other problem domains as long as they have similar geometric structure. Using an identical framework with a minimum of changes limited to the settings for parameterization and values of the kernel parameters, I achieved compet-

itive results on seemingly quite unrelated problem domains.

- **Scalability**

  The parametric kernel function requires on average a near linear time cost to evaluate, depending on the parameter space decomposition scheme. Therefore, it is scalable to inputs that consist of a large number of local feature vectors. Also, since it is a Mercer kernel, the parametric kernels can be applied to any learning algorithm based on convex optimization. In this dissertation, the parametric kernel functions defined for sequences and unordered sets have been successfully used in support vector learning algorithms for classification and novelty detection.

- **Handwritten Character Recognition**

  As the first set of experiments, I applied the parametric kernel framework to the task of recognizing handwritten characters. I designed parametric kernel functions for sequences of points using 1D manifold parameterization. I achieved results that are superior to a number of state-of-the-art techniques that are specifically designed to recognize digits, English alphabets, and some mathematical symbols. With just minor changes in the kernel parameter values, the same kernel function has been applied to the task of recognizing known objects from the sensor data captured by a laser range finder. With an appropriate choice of local feature representations such as the shape context [11], it is straightforward to extend this application to perform critical robotics tasks such as localization or navigation.

- **Face recognition under varying illumination**

  Face recognition under varying illumination is an extremely difficult task to solve. This is all the more so if we are to recognize faces from low quality video streams captured by commodity webcams that are often used in robotics. As part of the work for Robocup at Home competition, I built a real-time system that learns to identify faces from video streams. Using SIFT as the feature representation, I achieved excellent results under steady or known illumination conditions. However, the performance dropped significantly when the test illumination condition is unknown during training. I designed a parametric kernel for face images represented as a set of SIFT descriptors, using the position of each SIFT descriptor as its parameter in 2D manifold. This greatly improved the recognition rate.

7.2 Applications

The series of problems solved in this thesis provides typical examples of using my method. I design the parameter space and construct customized similarity metrics as parametric kernel functions following the proposed scheme. In this section, I discuss the application of the parametric kernel technique to other problem domains that I have not addressed in this work. The primary goal of this discussion is to provide an analysis of the proposed method which may provide a useful insight into issues such as when and how one could use the parametric kernel technique to solve the problem at hand, what must be considered to make this method work

well, and what its limits are.

The parametric kernel technique allows for an easy and straightforward adaptation to sequences. In particular, if one needs to construct similarity metrics for varying length sequences of feature vectors that could be used within a state-of-the-art kernel learning framework such as support vector machines, my approach yields an effective and efficient solution. In comparison to the well known sequence matching technique based on dynamic time warping (DTW), parametric kernels take much less time and memory to evaluate. The parametric kernel technique works well with sequence data where not only the order but also the relative distances between the consecutive feature vectors convey information that defines the data characteristics. For instance, consider handwritten characters for the same letter that consist of varying numbers of points. This may happen in real handwritten character systems due to differences in the speed they were written. If the characters look similar to humans, then my method yields quite stable similarity metrics that are less dependent on how dense or sparse the distribution of the points is. However, my method may be weak when the characters contain noise or the shapes for the same characer vary dramatically. For instance, if a noise point is introduced in the middle of a stroke, then this will shift the parameters of the remaining points in the stroke. In comparison, DTW yields similarity metrics that are much more stable than my method in such cases since such noisy points are skipped (warped) during optimization without interfering with matching the remaining points in the stroke.

We can apply the parametric kernel technique to structured data types with no specific notion of order between the elements as well. Instead, their positioning

in the input physical space provides similar information. For instance, consider the face recognition problem. I used the normalized coordinates of the positions within an image where SIFT features are computed as their parameters. In some sense, I would like to argue that the positioning could be considered as an extended notion of ordering. My reasoning is as follows. Parameters for sequences encode the order and the relative distances in terms of their positions along a 1D parametric axis. Elevating parameters to a higher dimension yields a notion of ordering that supports the kind of parameters used in my work on face recognition. The parametric kernel technique in higher dimensions works well if the relative positioning of feature vectors conveys information that defines the data characteristics. For instance, in my face recognition work, the relative positions of fiducial points in the face such as eyes, nose, and mouth within the face do not vary dramatically, while the SIFT features varied very sensitively due to the changes in the illumination condition. However, my method may be weak if there is a significant amount of occlusion or distortion of the objects. Therefore, my method will not work well with image categorization partial matching for image retrieval.

7.3 Future Work

The work in this dissertation entails some challenging topics for future research.

- **Automated and flexible parameter space decomposition** This dissertation made heavy use of an overlapping regular parameter space decomposition scheme. However, determining the size of each range and the hop length was

somewhat arbitrary and heuristic. If the decomposition is too fine-grained, there will be no local features that match, while, with a composition that is too rough-grained, we will be swamped by bad matches [23]. It will be challenging but very helpful to be able to automatically determine the optimal size of the ranges.

Also, regular decomposition may be less effective than more flexible manual decompositions and not even work for certain problem domains. There has been some similar work to find irregular decomposition of the space of local features for local matching, e.g. vocabulary-guided irregular pyramid match kernels [23]. Finding an optimal decomposition scheme using any other useful pieces of information from the context of a given problem is a challenging and important step to enhance the quality of the technique presented in this dissertation.

- **Computational efficiency**

  Though the evaluation of parametric kernels requires a near linear time cost, it still means that the kernel functions that compute the distances between the local feature vectors and between their parameters must be evaluated that many times. Moreover, overlapping the ranges increases the computational cost even further. Therefore, reducing the computation cost is also a very important enhancement to the current technique. Various computationally efficient approximation techniques may be applicable, or kernel functions that are computationally efficient than those used in this work may be chosen

instead.

- **Application to a broader range of problems**

  This dissertation demonstrates the general applicability of the parametric kernel framework by applying it to three seemingly quite unrelated problem domains. However, the structure that underlies the input patterns of those tasks is limited to just 1D or 2D manifolds. It will be very interesting to see how this technique works in a broader range of problem domains and more complex input structures. A problem domain to which we can immediately apply parametric kernels using 1D manifold parameterization is analyzing audio streams.

Bibliography

[1] *http://synapse.vit.iit.nrc.ca/db/video/faces/cvglab/*.

[2] *http://vision.ucla.edu/~vedaldi/code/sift/sift.html*.

[3] *http://www.cs.utexas.edu/~AustinVilla/*.

[4] *http://www.robocup-us.org/*.

[5] *http://www.unipen.org/*.

[6] *Predicting Structured Data*. Edited by G. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Takar, and S. V. N. Vishwanathan, 2007.

[7] I. Aleksander and T. Stoham. *Guide to Pattern Recognition Using Random-Access Memories*. IEE Proceedings on Computer and Digital Techniques, Vol. 2, pages 22–40, 1979.

[8] C. Bahlmann and H. Burkhardt. *Measuring HMM Similarity with the Bayes probability of error and its applications to on-line handwriting recognition*. Proceedings of the 6th International Conference on Document Analysis and Recognition, 2001.

[9] C. Bahlmann, B. Haasdonk, and H. Burkhardt. *On-line Handwriting Recognition with Support Vector Machines - a Kernel Approach*. Proceedings of the 8th International Workshop on Frontiers in Handwriting Recognition, 2002.

[10] P. Belhumeur, J. Hesphana, and D. Kriegman. *Eigenfaces vs. Fisherfaces: Recognition using Class Specific Linear Projection*. PAMI, Vol. 19, No. 7, pp 711-720, 1997.

[11] S. Belongie, J. Malik, and J. Puzicha. *Shape Matching and Object Recognition Using Shape Contexts*, volume 24. IEEE Transactions on Pattern Analysis and Machine Intelligence, April 2002.

[12] A. Berg and J. Malik. *Geometric blur for template matching*. IEEE Conference on Computer Vision and Pattern Recognition, 2001.

[13] M. Bicego, A. Lagorio, E. Grosso, and M. Tistarelli. *On the Use of SIFT Features for Face Authentication*. Conference on Computer Vision and Pattern Recognition Workshop, 2006.

[14] C. Cabani. *Implementation of An Affine-invariant Feature Detector in Field-Programmable Gate Arrays*. Masters Thesis, University of Toronto, 2006.

[15] T. Choudhury, B. Clarkson, T. Jebara, and A. Pentland. *Multimodal Person Recognition Using Unconstrained Audio and Video*. Proceedings of International Conference on Audio- and Video-based Person Authentication, pp 176-181, 1999.

[16] M. Collins and N. Duffy. *Convolution Kernels for Natural Language*. Advances in Neural Information Processing Systems, Vol. 14, pages 625–32, Cambridge, MA, MIT Press, 2002.

[17] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[18] E. Bailly-Baillire et el. *The BANCA database and evaluation protocol*. International Conference on Audio- and Video- Based Biometric Person Authentication, 2003.

[19] H. Fatemi, R. Kleihorst, H. Corporaal, , and P. Jonker. *Real-time Face Recognition on A Smart Camera*. Proceedings of Advanced Concepts for Intelligent Vision Systems, 2003.

[20] M. Fleming and G. Gottrell. *Categorization of Faces Using Unsupervised Feature Extraction*. Proceedings of the International Joint Conference on Neural Networks, Vol. 2, pp 65-70, 1990.

[21] Dmitry O. Gorodnichy. *Video-based framework for face recognition in video*. Computer and Robot Vision, 2005.

[22] N. Gourier, D. Hall, , and J.L. Crowley. *Facial Features Detection Robust to Pose, Illumination, and Identity*. IEEE International Conference on Systems, Man, and Cybernetics, 2004.

[23] K. L. Grauman. *Matching Sets of Features for Efficient Retrieval and Recognition*. Ph.D. Thesis, MIT, 2006.

[24] S. Gupta, J. K. Aggarwal, M. K. Markey, and A. C. Bovik. *3D Face Recognition Founded on the Structural Diversity of Human Faces*. IEEE Conference on Computer Vision and Pattern Recognition, 2007.

[25] D. Haussler. *Convolution Kernels on Discrete Structures*. UC Santa Cruz Technical Report UCS-CRL-99-10, 1999.

[26] B. Heisele, P. Ho, , and T. Poggio. *Face Recognition with Support Vector Machines: Global versus Component-based Approach*. IEEE International Conference on Computer Vision, 2001.

[27] J. Hu, S. G. Lim, and M. K. Brown. *Writer independent on-line handwriting recognition using an HMM approach.* Pattern Recognition, Vol. 33, pages 133-147, 2000.

[28] T. S. Jaakkola and D. Haussler. *Exploiting generative models in discriminative classifiers*. Advances in Neural Information Processing Systems, Vol. 11, 1998.

[29] E. Keogh and S. Kasetty. *On the need for time series data mining benchmarks: A survey and empirical demonstration.* SIGKDD, 2002.

[30] U. Kreßel. *Pairwise Classification and Support Vector Machines*. Advances in Kernel Methods - Support Vector Learning, 1999.

[31] D. D. Lee and H.S. Seung. *Learning the Parts of Objects by Non-negative Matrix Factorization*. Nature, Vol. 401, No. 6755, pp 788-791, 1999.

[32] H. Lei and V. Govindaraju. *Similarity-driven Sequence Classification Based on Support Vector Machines*. ICDAR, 2005.

[33] H. Lodhi, J. Shawe-Taylor, N. Cristianini, and C. Watkins. *Text Classification Using String Kernels*. Advances in Neural Information Processing Systems, Vol. 13, Cambridge, MA MIT Press, 2001.

[34] David G. Lowe. *Distinctive image features from scale-invariant keypoints*. Number 60, 2 (2004). International Journal of Computer Vision, 2004.

[35] S. Lucas and A. Amiri. *Statistical Syntactic Methods for High Performance OCR*. IEE Proceedings Vision, Image and Signal Processing 143(1), pages 23–30, 1996.

[36] Simon M. Lucas and Tzu-Kuo Huang. *Sequence Recognition with Scanning N-Tuple Ensembles*. International Conference on Pattern Recognition, pages 410–413, 2004.

[37] J. Luo, Y. Ma, E. Takikawa, S. Lao, M. Kawade, , and B.L. Lu. *Person-specific SIFT Features for Face Recognition*. International Conference on Acoustics, Speech, and Signal Processing, 2007.

[38] K. Mikolajczyk and C. Schmid. *An affine invariant interest point detector*. European Conference on Computer Vision, 2002.

[39] R. Neal, P. Dayan, G. E. Hinton, and R. S. Zemel. *The Helmholtz Machine*. Neural Computation, pages 1022–1037, 1995.

[40] P. J. Phillips, A. Martin, C.l. Wilson, and M. Przybocki. *An Introduction to Evaluating Biometric Systems*. Number Vol. 21 No. 2. Computer, 2000.

[41] J. Platt. *Probabilities for SV Machines*. Advances in Margin Classifiers, Cambridge, MA, MIT Press, 2000.

[42] V. Popovici, J.P. Thiran, Y. Rodriguez, and S. Marcel. *On performance evaluation of face detection and localization algorithms*. Proceedings of the International Conference on Pattern Recognition, 2004.

[43] F. M. Porikli. *Trajectory Distance Metric Using Hidden Markov Model Based Representation*. European Conference on Computer Vision (ECCV), May 2004.

[44] L. Prevost and M. Milgram. *Modelizing Character Allographs in Omniscriptor Frame: A New Non-supervised Clustering Algorithm*. Pattern Recognition Letters, pages 295–302, 2000.

[45] E. H. Ratzlaff. *Methods, Report and Survey for the Comparison of Diverse Isolated Character Recognition Results on the UNIPEN Database*. International Conference on Document Analysis and Recognition, 2003.

[46] T. Sauquet, S. Marcel, , and Y. Rodriguez. *Multiview Face Detection*. Internal Report IDIAP-RR-05-49, IDIAP, 2005.

[47] B. Schölkopf and A. J. Smola. *Learning with Kernels, Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 2002.

[48] M. Seeger. *Bayesian Methods for Support Vector Machines and Gaussian Processes*. University of Edinburgh, Division of Informatics, 1999.

[49] J. Shawe-Tayler and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[50] H. Shimodaira, K. Ichi, N. M. Nakai, and S. Sagayama. *Dynamic time-alignment kernel in support vector machines*. Advances in Neural Information Processing Systems, Vol. 14, 2002.

[51] Y. Shin and D. Fussell. *Parametric Kernels for Sequence Data Analysis*. International Joint Conference on Artificial Intelligence, 2007.

[52] P. Sollich. *Probabilistic Interpretation and Bayesian Methods for Support Vector Machines*. ICANN'99 - Ninth International Conference on Artificial Neural Networks, 1999.

[53] Johan W. H. Tangelder and Ben A. M. Schouten. *Learning a Sparse Representation from Multiple Still Images for On-Line Face Recognition in an Unconstrained Environment.* International Conference on Pattern Recognition, 2006.

[54] E. Tapia and Raúl Rojas. *Recognition of On-line Handwritten Mathematical Formulas in the E-Chalk System*. IDCAR, 2003.

[55] M. Turk and A. Pentland. *Face Recognition Using Eigenfaces*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1991.

[56] I. Ulusoy and C.M. C. M. Bishop. *Generative versus Discriminative Methods for Object Recognition*. IEEE Conference on Computer Vision and Pattern Recognition, 2005.

[57] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[58] P. Viola and M. Jones. *Rapid object detection using boosted cascade of simple features*. Computer Vision and Pattern Recognition, 2001.

[59] L. Vuurpijl, R. Niels, M. van Erp, L. Schomaker, and E. Ratzlaff. *Verifying the UNIPEN devset*. Ninth International Workshop on Frontiers in Handwriting Recognition, 2004.

[60] L. Vuurpijl and L. Schomaker. *Two-state Character Classification: A Combined Approach of Clustering and Support Vector Classifiers*. The Seventh International Workshop on Frontiers in Handwriting Recognition, 2000.

[61] C. Watkins. *Kernels from Matching Operations*. Technical Report CSD-TR-98-1, Royal Holloway College, Computer Sciences Department, 1999.

[62] C. Watkins. *Dynamic Alignment Kernels*. In A. J. Smola, P. L. Barlett, B. Schölkopf, and D. Schuurmans, editors, Advances in Large Margin Classifiers, pages 39–50, 2002.

[63] J. Weston and C. Watkins. *Multi-class Support Vector Machines*. ESANN, 1999.

[64] B. Weyrauch, J. Huang, B. Heisele, , and V. Blanz. *Component-based Face Recognition with 3D Morphable Models*. 4th Conference on Audio- and Video-Based Biometric Person Authentication, 2003.

[65] L. Wiskott, J.M. Fellous, , and C. von der Malsburg. *Face Recognition by Elastic Bunch Graph Matching*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, pp 775-779, 1997.

[66] M.H. Yang, D. J. Kriegman, , and N. Ahuja. *Detecting Faces in Images: A Survey*. IEEE Transactions on Patter Analysis and Machine Intelligence, Vol. 24, No. 1, January 2002.

[67] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. *Face Recognition: A Literature Survey*. Number Vol. 35, No. 4. ACM Computing Surveys, 2003.

Vita

Youngin Shin was born in Busan, The Republic of Korea on 29 May 1974, the son of Tae-gon Shin and Hyang-seon Shim. He received the Bachelor of Engineering degree in Computer Engineering from the Pusan National University and was commissioned an Officer in the Republic of Korea army in 1997. He entered active duty as a software engineer in September, 1997, at Busan. Separating from the Republic of Korea army, he was accepted and started the graduate studies in the fall of 2000.

Permanent Address: 23425 SE Black Nugget Road APT R302

Issaquah, WA 98029

This dissertation was typeset with LaTeX 2_ε by the author.