View-based Models for Visual Tracking and Recognition

Haihong Zhang

NATIONAL UNIVERSITY OF SINGAPORE
2005

View-based Models for Visual Tracking and Recognition

HAIHONG ZHANG
(*M.Eng, University of Science and Technology of China*)

A THEIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

2005

Acknowledgements

I would like to thank Dr Huang Weimin and Dr Huang Zhiyong who were my supervisors and provided many ideas together with large amounts of enthusiasm, motivation, and really useful technical help.

A big thank you to others acted as my mentors or colleagues, especially my previous supervisor, Dr Guo Yan, who led me to interesting research fields in computer vision and pattern recognition. Dr Zhang Bailing also deserves a special thank you for his valuable instructions plus his vital role in my PhD program. Often, I am also reminded of a lot of kind help from Dr Li Liyuan who plays an important role in my work on visual tracking.

Main part of this thesis was done in the Institute for Infocomm Research (I2R), Singapore. And I would like to take this opportunity to express my great appreciation to I2R for its help and support.

My family all live far from Singapore but are close in other ways. In fact their help should be much more appreciated than they realized, and I would like to give a thousand thanks to Mum, Dad, Jili and Haiyan. In particular, I am fully grateful to my wife, Lin Hong. During most of my life in Singapore, we were far apart but she was always offering me a great deal of happiness, encouragement and inspiration. I am so happy that I married here just before finishing my dissertation.

Abstract

The objective of the thesis is to develop efficient view-based models for determining the states and the identities of target objects in images.

The thesis first proposes a kernel-based method for tracking objects under affine transformation. The basis of the method is a spatially-and-spectrally smooth affine matching technique. By precisely characterizing each object's spatial and spectral features, the technique can distinguish similar objects in cluttered scenes and provides the posture information of the objects that is useful for motion understanding and subsequent visual processing such as recognition. Tracking is formulated as optimizing the matching with respect to affine parameters. An efficient, iterative optimization method is then proposed, and its superior performance is demonstrated in extensive experiments.

For generic pattern classification, the thesis presents a learning and classification model called kernel autoassociators. The model takes advantage of kernel feature space to learn the nonlinear dependencies among multiple samples. It is easier to implement than conventional autoassociative networks, while providing better performance. In addition, the thesis proposes a Gabor wavelet associative memory model that inherits advantages of Gabor wavelet networks in face representation as well as that of kernel autoassociators in nonlinearity learning. The model can dramatically improve the capability of kernel autoassociators in learning faces, yielding a high-performance face recognition system.

Note that the following web site provides video sequences and accessory materials related to the thesis.

http://www1.i2r.a-star.edu.sg/~hhzhang/PhDThesis

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In one's daily life, visual recognition plays a leading role in the process of information acquisition from the environment. The huge amount of visual information is continuously received by approximately 130 million photosensitive cells, rods and cones in the retina, which then transfers the active image via the optic nerve to the brain [Hubel and Wiesel, 1994]. Still in mystery, the brain exhibits an excellent capability in processing the data, abstracting an idea of the dynamic world in relation to oneself, and identifying the immediate situation.

In the computer vision community, people have been pursuing the capability of human vision for a few decades, especially by developing computational approaches to automatic visual recognition. Here the term "automatic visual recognition" refers to using computers to find and identify known objects (given physical objects such as the computer I am using) in the perceived images of the environment. It is recognized that automatic visual recognition has a broad range of applications such as video surveillance, vehicle navigation, advanced human-computer interface, virtual/mixed reality, biometric person identification and medical image analysis.

Automatic visual recognition in general remains a very difficult problem primarily due to the sheer complexity of visual tasks. To understand the difficulties, let's consider a specific recognition task, namely, face recognition from a sequence of images.

First of all, the system needs to locate the faces of interest (called targets) in the images, and to keep attention on them when they are moving around. This is referred

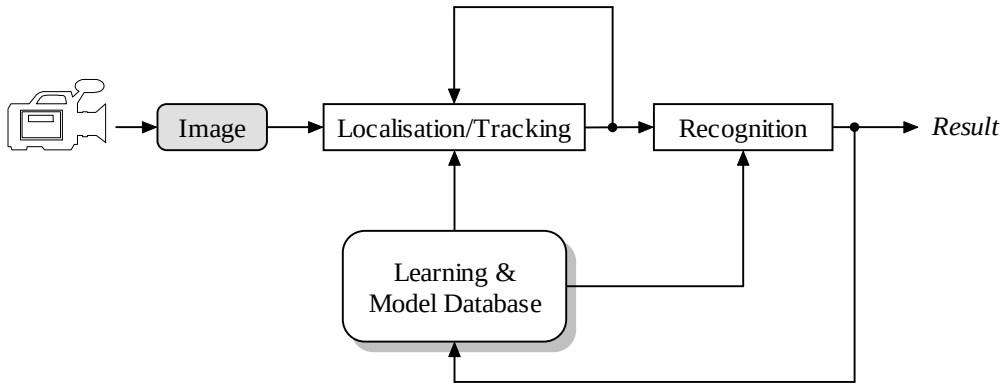


Figure 1.1: Automatic Visual Recognition System

The localization/tracking module serves visual attention. Note that tracking is a recursive inference problem. By learning given images either on-line or off-line, the learning module acquires the essential knowledge about each object of interest and stores the pattern into the database. The knowledge is then used by the recognition module to identify the object image recovered by the localization/tracking module.

to as visual attention in biological context, or visual detection and tracking in computer vision. James [James, 1950] describes the attention as "the taking possession by the mind, in clear and vivid form, of one out of what seem several simultaneously possible objects". He also believes that during the attention, one principal object comes into focus while others are temporarily suppressed. However, because the attention task involves dynamic imagery and scene analysis which is not fully understood, visual localization/tracking is a problem of especial difficulty [Toyama, 1998].

After the system locks on the faces, the subsequent recognition process is essentially to match the observed face images with known faces. In fact, face matching is never a trivial task since computers need to distinguish a number of faces that have subtle difference while being subject to considerable variations in terms of facial expressions, poses and imaging conditions [Zhao et al., 2000].

With the above problems in mind, we need a visual recognition system consisting of a few basic components (Figure 1.1). The learning/database module learns (either online or offline) objects of interest from given samples. After image acquisition (frame grabbing), the localization/tracking module determines the present state of a target object. The recognition module identifies the tracked object by comparing it with object

models learned in advance.

Clearly, both tracking and recognition require object models that can distinguish a particular object from the others and determine its states in given images. In respect of visual object modeling, two methodologies are prevalent in the computer vision community.

The first methodology is based on 3D model representation in which we assume that an object can be represented by a mathematical model (such as a 3D generic face model [Parke and Waters, 1996]) consisting of a set of feature points/surfaces in 3D space, while there are corresponding features in 2D images. When a 2D image is presented for tracking and recognition, one needs to rebuild the correspondence between the 2D features and their 3D counterparts of the object, and this process is called *alignment*. In reality, due to the variability in object shape as well as limited sensor resolution, 2D image features may not occur exactly in the positions predicted by the mathematical model. Thus, alignment program should allow a small, bounded amount of displacement of the feature points, and such a methodology is often referred to as *bounded error alignment* [Grimson, 1990]. In facial motion analysis, for example, Terzopoulos and Waters employed complex and physical face models that account for both skin and muscle dynamics [Terzopoulos and Waters, 1993].

It is noteworthy that the movie industry is calling for realistic 3D models. Correspondingly, recent years have seen a surge of research on realistic 3D models which are designed to meet the industrial demand. For example, Dimitrijevic et al. presented a fast, model-based structure-from-motion approach to reconstructing faces from uncalibrated video sequences [Dimitrijevic et al., 2004].

In the field of visual tracking and recognition, realistic models may not be required. In fact, many researchers prefer to relatively simpler 3D models. For example, La Cascia et al. proposed a texture-mapped 3D cylindrical model for head tracking [Cascia et al., 2000], and Wiles et al. suggested using hyper-patches to model a head [Wiles et al., 2001]. Many works on articulated human-body modeling resort to using a set of blobs/elements to describe a figure. In [Plankers and Fua, 2003], for example, Plänkers and Fua developed a body-modeling framework that relies on attaching im-

plicit surfaces to an articulated skeleton, leading to a differentiable model that permits efficient implementation of minimization for the purpose of tracking.

The second methodology for visual object modeling is view-based. A view-based model consists simply of a collection of 2D views of a 3D object. One does not need to establish the explicit 3D configuration of feature points on the object. To account for 3D movements of the object, certain transformations in the 2D views are considered. For recognition, the presented image would be compared either directly with sample views or with their high level representations (e.g. principal components [Turk and Pentland, 1991]).

In comparison with 3D models, view-based models have two important advantages. First, they greatly simplify model acquisition – the representation of physical surfaces. Thus, they avoid the potential of modeling error caused by incomplete or inaccurate 3D representation. Second, view-based models allow visual problems to be solved in a simpler 2D framework. Thus, they are particularly suited to computer vision tasks in which the computation of precise correspondence between images and 3D space is not feasible. Furthermore, Aloimonos has asserted that general 3D scene recovery is a very hard problem and many recovery systems are inherently unstable. He believes that complete and accurate recovery of scenes is not necessary for many of the problems we need to solve using vision [Aloimonos and Rosenfeld, 1991].

Many view-based models consider each image as a two dimensional random pattern or merely a vector after concatenation of rows or columns, and resort to learning the statistical features of the patterns. They may face the problems caused by image deformations of visual objects. Since transformations such as posture change can yield complicated variations in the images, it remains rather difficult for statistical models to handle. The problems will become more serious when only a few samples per object are available for system training. In tracking an unknown or unfamiliar target, for instance, perhaps just one image sample is available for reference.

Hence, there is a need to develop efficient view-based models, which can learn from one or a few samples to recover certain image deformations of visual objects and to determine their identities despite possibly large image variations.

1.2 Objective and Contributions

With the above motivation, the fundamental objective of the thesis is to develop efficient view-based models for reasoning the states and the identities of (moving and transforming) target objects in image sequences.

The thesis comprises two major contributions to the visual tracking and classification disciplines. The first contribution is an efficient view-based tracking method that can infer the posture state (position, size, non-uniform scaling factors, orientation, etc.) of a target object from images. The basis of the method is a kernel-based spatially-and-spectrally smooth similarity measure, which can precisely characterize the spatial and spectral features of the object under affine transformation while being robust against motion blurs, heavy noise, or visible artifacts in images. The measure is suitable for accurately describing the relation in terms of affine transformation between object images, and leads to an efficient, iterative optimization procedure to tracking. Furthermore, the tracking method depends on merely one sample image for reference. Thus, it is easy to implement and is widely applicable. By combining posture estimation with accurate spatial-spectral representation, the method has two major advantages. First, it can identify and distinguish between similar objects in cluttered scenes. Second, with the recovered information about transformation, it naturally leads to better motion understanding than non-posture-estimation methods that may just recover the object translation.

The second contribution is to the theory of autoassociators - a special type of neural networks, and their use in computer vision applications. In particular, the thesis proposes a generic learning machine called the kernel autoassociator model which takes advantage of kernel feature space to learn the nonlinear dependencies among multiple samples. The model is much easier to implement than conventional autoassociative networks, while providing better performance for novelty detection and multi-class classification. In addition, we also put emphasis on the extension of kernel autoassociators for face recognition. A novel face representation model called Gabor wavelet associative memories is presented that dramatically improves the capability of kernel autoassociators in learning face images, yielding a high-performance face recognition system.

1.3 Overview

The thesis will address the aforementioned problems in visual tracking and classification. Due to the diverse nature of these problems, a unified literature survey may tend not to elucidate but to confuse. Instead, I would like to include literature surveys in their corresponding chapters. In particular, Chapter 2 presents a brief survey on object representation models for tracking, Chapter 3 reviews visual tracking systems, Chapter 4 starts by surveying classification algorithms, and Section 5 begins with a review of face recognition algorithms.

The content of the thesis is as follows. Chapter 2 is a self-contained description of a kernel-based image matching technique for objects under transformation. The technique is based on a spatially-and-spectrally smooth similarity measure that offers capability for accurate and robust posture estimation. To account for image deformations caused by posture changes, we develop the similarity measure under a typical type of transformation – affine transformation which is formulated as a combination of a few geometrical operations in terms of translation, rotating, (non-uniform) scaling and shearing. The chapter carefully investigates the properties of the affine matching technique, especially in real situations where an object candidate in the form of an image region may include a number of background pixels. We show that the background interference may pose a serious problem to affine matching. Our further study, by investigating how the interference affects matching with respect to individual transformation factors, favorably suggests a solution to the interference problem.

Chapter 3 follows the study on affine matching in Chapter 2, and emphasizes developing and assessing a robust tracking method. We derive an iterative and analytical procedure for maximizing the similarity, with respect to the parameters of affine transformation, between an object candidate and a given model. A tracking algorithm is developed by combining the similarity-maximization procedure and the knowledge about the properties of affine matching. We then discuss the computational complexity and efficient implementation of the algorithm.

The chapter further describes extensive experiments on the proposed method. Using computer-generated image sequences, we examine the robustness of the method against

image noise. Moreover, we assess the tracker with a variety of real-life objects such as faces, hands, cars and camouflaged tanks. Positive and convincing experimental results are obtained. In addition, the last section discusses the importance of using explicit physical operators (regarding scaling, rotation, shearing and translation) in the affine tracking system, by showing that an affine transformation model without explicitly accounting for physical operators would hardly lead to practical tracking algorithms.

With the tracking method described in the above two chapters, we are able to recover target object images under affine transformation. In the following two chapters, we study how to identify the objects from the recovered images.

Chapter 4 presents for generic pattern classification a novel nonlinear model referred to as kernel autoassociators. While conventional nonlinear autoassociation models emphasize searching for the nonlinear representations of input patterns, a kernel autoassociator takes a kernel feature space as the nonlinear manifold, and places emphasis on the reconstruction of input patterns from the kernel feature space. Two methods are proposed to address the reconstruction problem, using linear and multivariate polynomial functions respectively. We apply the proposed model to novelty detection with or without novel examples, and study it on the Promoter detection and Sonar Target recognition problems. We also apply the model to multi-class classification problems including wine recognition, glass recognition, handwritten digit recognition and face recognition. The experimental results show that kernel autoassociators can provide better or comparable performance for concept learning and classification in various domains than conventional autoassociators or other state-of-the-art generic classification systems.

In Chapter 5, we study how to extend kernel autoassociator models for face recognition. We propose a novel face representation model called Gabor Wavelet Associative Memory (GWAM) by incorporating domain knowledge with a subject dependent Gabor wavelet network. The domain knowledge used here is that an individual face has a certain configuration of local and global image features such that we can develop a set of special image kernels (Gabor wavelets) to represent them. Finally, we carry out extensive experiments to evaluate a GWAM-based face recognition system, in comparison with other state-of-the-art face recognition systems. Our scheme demonstrates excellent

performance on three popular databases, namely, the FERET (Release 2), the ORL and the AR face database.

Chapter 6 presents the conclusion, followed by some brief speculations on future development of the proposed models/approaches.

The thesis includes some material that has been presented in a few papers, namely, [Zhang et al., 2004a], [Zhang et al., 2005], [Zhang et al., 2004c][1], [Zhang et al., 2004b], [Zhang et al., 2004d]. Besides, the following web site also provides video sequences and accessory materials related to the thesis.

http://www1.i2r.a-star.edu.sg/~hhzhang/PhDThesis

[1] under review

Chapter 2

Kernel-based Affine Matching

2.1 Background

A fundamental problem addressed by this thesis is to search for dynamic target objects in the *pose space* (the terminology follows [Grimson, 1990]) while only one sample image per object is provided for reference. Here the pose space refers to the set of all possible state of an object in terms of, e.g., position, orientation or size. A critical problem in tracking is object matching which tells the likelihood of an object's pose from a given observation. A matching program would serve prominent functions in identifying the target object's state and distinguishing the object from the cluttered background, while the basis of the matching program would be the model for the object that describes the object's characteristics with respect to its poses.

In the field of view-based approaches, an object model is usually set up over image regions in terms of spatial and color features. The literature has seen a great deal of relevant research on region modeling and tracking. Table 2.1 summarizes them into two rough categories – color models and spatial-color models.

The first category emphasizes the color features of target objects. Various parametric statistical techniques have been used for exploiting essential spectral statistics of objects' image appearance. In [Wern et al., 1997] a unimodal Gaussian was engaged to model the color properties of a blob region. Oliver et al. [Oliver et al., 2000] and Yang and Waibel [Yang and Waibel, 1996] also employed a Gaussian distribution to represent a skin color cluster of thousands of skin color samples taken from different races. The facial color features, if put in appropriate color spaces [Lee et al., 1996, Dai and Nakano, 1996], have

Category	Methods	Translation	Rotation	Deformation*	Accuracy
Color Models	Blobs	√	√	×	low
	Color-Hist.&Kernels	√	√	×	average
Spatial-Color Models	Image templates	√	×	×	high
	Spatial-feature kernels	√	×	×	high
	Our method	√	√	√	high

* Deformation here refers to shearing and non-uniform scaling.

Table 2.1: Categorization of Appearance-based Methods for Visual Tracking.

been shown to be robust against changes in environment factors such as illumination conditions and imaging characteristics (cf. Terrilon's comparative study on several widely used color spaces for face detection [Terrillon et al., 2000]). Furthermore, multi modal Gaussian using Expectation-Maximization algorithm allows one to model blobs with a mixture of colors [Raja et al., 1998a, Raja et al., 1998b], while it is still an open problem how to choose the right number of Gaussians.

Non-parametric techniques such as color histograms have also been extensively studied with visual tracking. Unlike parametric techniques, they do not rely on presumed probability distribution models. In particular, color histograms appear to be very popular in video-processing systems for face and head tracking/detection [Birchfield, 1998, Pei and Tseng, 2002, Cho et al., 2001], hand tracking [Martin et al., 1998], and people tracking [Withagen et al., 2002, Lee et al., 2003], or in the field of color indexing [Funt and Finlayson, 1995]. A recent remarkable work in the area was presented by Comaniciu et al. [Comanicui et al., 2000] who combined spatial kernels and color histograms to obtain a spatially-smooth similarity function which leads to an efficient, mean-shift [Cheng, 1995] optimization procedure to tracking. The mean-shift tracking method has demonstrated excellent performance in various, difficult tracking scenarios [Comaniciu et al., 2003]. Moreover in [Collins, 2003] it was extended to deal with scaling objects.

Some of the reasons for color histograms' wide applicability are that it can be computed easily and fast, it achieves significant data reduction, and it is robust to noise and local image transformations [Hadjidemetriou et al., 2001]. A general drawback with color histograms is the lack of convergence to the right density function if the data set is

small. Therefore, in a recent work [Elgammal et al., 2001] another non-parametric technique called kernel density estimation was preferred for modeling color features. The authors applied the technique to people segmentation, and also extended the technique to people tracking [Elgammal et al., 2003a].

The above methods mostly avoid explicit or accurate spatial feature exploration. On one hand, they are robust against variations to some extent in scale and pose; on the other hand, they may be incapable of distinguishing color objects which have similar color distributions so that the characterization of spatial features is critical. Furthermore, they cannot provide posture information which is important for motion understanding.

The second category, spatial-spectral based methods, may be used to infer the posture of target objects by exploiting the correlation between spatial and color features in an object image. For detection and tracking, they may take all image windows of a particular shape and test them to tell if the relevant object is present. Thus, many of them are related to template matchers. While many objects appear hard to find with simple template matchers, there is some evidence that reasoning about relations between many different kinds of templates can be an effective way to find objects. In [Viola and Jones, 2004], for instance, Viola and Jones presented a fast face detection scheme that searches all image windows for faces using an Adaboost classifier trained with a large amount of face and non-face data. In the literature, as a matter of fact, learning image templates provides a basis for many tracking systems [Avidan, 2004, Mohan et al., 2001, Nguyen and Smeulders, 2004].

It is recognized that simple template methods may not be robust against image variations caused by object deformation. An effective methodology by using deformable templates thus was introduced. Typical examples range from snakes [Blake and Isard, 1998] to more recent models such as active shape models [Cootes et al., 1993] and active appearance models [Cootes et al., 2001]. The active models are capable of extracting complex and non-rigid features. A drawback is that the setup of deformable models requires the use of expert knowledge and expensive job in training.

The present work emphasizes spatial-spectral based methods to involve posture es-

timation in tracking, as the incorporation of posture estimation has two important advantages. First, it can improve the system performance for distinguishing similar objects and cluttered background. Second, it would lead to better motion-understanding. The main challenge can be identified as the combination of accurate spatial-spectral representation and robust pose estimation. The posture here concerns orientation, scaling and possibly other transformation factors. The above review shows that, however, posture estimation for tracking is not well solved especially when one has quite limited knowledge about the target object. In a generic tracking system, for instance, perhaps only one sample image per object (some unfamiliar objects) is given for reference.

This chapter presents a novel method to address the problem, by proposing a kernel-based matching technique for objects under transformation. The basis of the technique is a representation model using kernel density estimation to characterize the spatial-spectral features of target objects, since much research has been done on the theoretical properties of the kernel estimator and its superiority over other estimators such as histograms is well-established [Scott, 1992]. (Interestingly, Terell has rigorously proved that virtually all nonparametric algorithms are asymptotically kernel methods [Terrell and Scott, 1992]). Based on the representation model, we propose an l_2 norm similarity measure that is spatially-and-spectrally smooth. The measure is suitable for accurate and robust object modeling, and offers capability for posture estimation.

Unlike the work [Elgammal et al., 2001] mentioned earlier that uses kernel density technique to address color modeling, the present technique addresses the correlation between spatial and spectral features. Thus, it is capable of providing precise representation and posture information for our tracking and classification purposes. Another related technique is by using local histograms [Lowitz, 1983, Bressan et al., 2003] which relies on heuristic knowledge about region segmentation or feature detection to combine spatial and spectral features. By contrast, the present technique fuses spatial and spectral information in a more accessible and effective way without the need for heuristic knowledge, thanks to the non-parametric kernel density method. Furthermore, unlike local histograms, our technique allows a spatially-and-spectrally smooth similarity measure that can give rise to an efficient optimization procedure for tracking, as will be

shown in the next chapter.

More importantly, the technique is suited to address a special image matching issue in which the objects are subject to affine transformation. We refer to this type of image matching as affine matching. It should be mentioned that kernel-based representation and tracking methods have been concerned earlier in [Elgammal et al., 2003b] where Elgammal et al. used a similar representation model to formulate a similarity measure and subsequently a tracking system. However, the computation of that similarity measure is difficult. As a result, even for tracking merely translational objects, the technique has to depend on a few critical approximations and assumptions that may not be well suited for object images under deformations (see Section 2.5.2). By contrast, our matching technique is directly derived from the kernel-based representation model and the affine transformation formulation in such a manner that the matching is easy and straightforward and does not rely on critical assumptions for describing images undergoing affine transformations.

The chapter also investigates the performance of the matching technique by using a few computer simulations. The consequent findings will contribute enormously to the development of a practical tracking system in the next chapter, where the excellent performance of affine matching (and affine tracking as an extension) will be demonstrated.

2.2 Kernel Density Estimation

An important point of the representation model to be proposed is that it resorts to using probability density function to characterize a target object's appearance in spatial-spectral space. The shear complexity of real world objects implies that it is hard to describe the density function with generic parametric methods. Instead, non-parametric methods especially kernel density estimation are favored for our purposes.

In this section we revisit a general case of density estimation using kernel methods. Given a set of samples of a random variable x, say $\{x_i\}, i = 1, \ldots, N$, one can estimate the cumulative distribution function (CDF) $\hat{F}(x)$, empirically by

$$\hat{F}(x) = \frac{1}{N} \sum_{i=1}^{N} U(x - x_i) \tag{2.1}$$

where $U(x)$ is a step function:

$$U(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2.2}$$

Hence, the empirical distribution function $\hat{F}(x)$ takes a staircase form. Being its derivative, the probability density estimation is a sum of Dirac delta functions which are not suited for smooth density functions in real world applications. To obtain a smooth estimation of a density function, an effective way is to use a regularization technique which aims to minimize an error function given by

$$e(F) = \int ||F(x_i) - \hat{F}(x_i)||^2 dx + \int ||f(x)||^2 dx \tag{2.3}$$

Here F is an estimator of cumulative distribution function. $f(x) = \frac{\partial F}{\partial x}$ denotes the density function. Thus, the first integral stands for the empirical estimation error, while the second integral about f imposes smoothing on the estimator F. A well-known solution to the optimization is known as Parzen windows, also called kernel density estimation [Parzen, 1979]

$$f(x) = \frac{1}{N\sigma} \sum_{i=1}^{N} k(x - x_i) \tag{2.4}$$

where $k()$ is a kernel or window function with bandwidth σ. Probably the most popular form of k is Gaussian:

$$k(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp\left[-\frac{1}{2}(\mathbf{x} - \mu)\Sigma^{-1}(\mathbf{x} - \mu)^T\right] \tag{2.5}$$

Here μ is the mean of $\mathbf{x}$ – a multivariate random vector, Σ is the covariance matrix that determines the scatter of $\mathbf{x}$ across the n-dimensional space. In particular, a large $|\Sigma|$ corresponds to a large bandwidth that in turn imposes strong smoothing on the density function, and vise versa.

In real applications, the multivariate Gaussian kernels are often simplified as product kernels ([Scott, 1992], p150)

$$k(\mathbf{x}) = \frac{1}{N\sigma_1 \cdots \sigma_d} \sum_{i=1}^{N} \left\{ \prod_{j=1}^{d} k\left(\frac{x_j - x_{ij}}{\sigma_j}\right) \right\} \tag{2.6}$$

Here the kernels k are univariate Gaussian functions. A vector $\mathbf{x}$ consists of n elements: $\{x_j\}, j = 1, \ldots, d$, and $\{\sigma_j\}$ is the corresponding bandwidth. Geometrically, the estimate places a probability mass of size $1/N$ centered on each sample point.

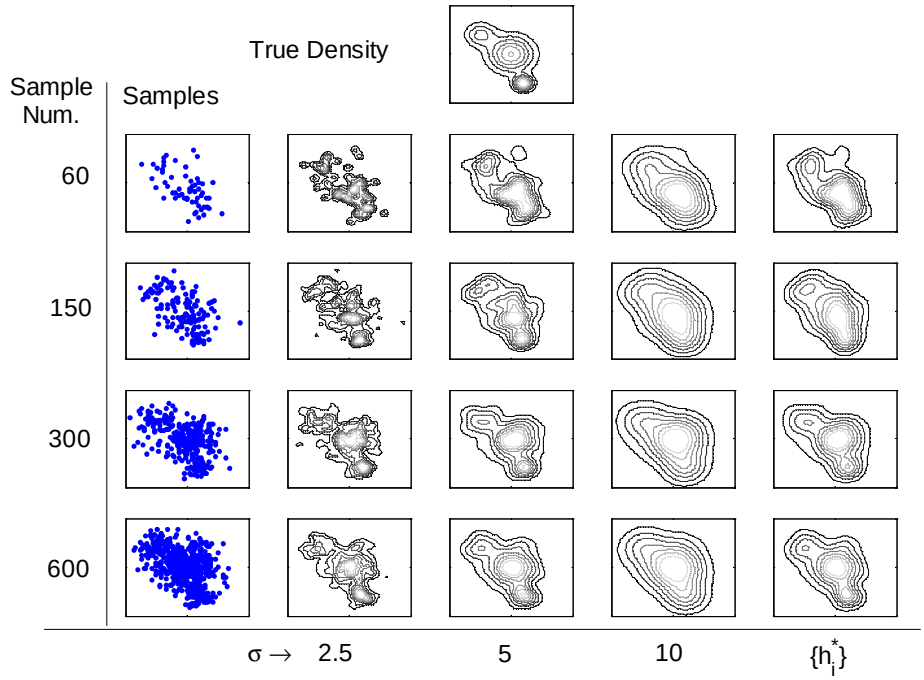


Figure 2.1: Kernel density estimates of a multi-Gaussian distribution. *The leftmost column shows the sample sets, to the right their corresponding estimate with different kernel bandwidth σ.*

Figure 2.1 shows kernel density estimation of a bivariate Gaussian distribution. The distribution comprises two uncorrelated variables with three major modes centered at $(0, 0)$, $(-25, 20)$, $(10, -30)$ whose standard variances are $(12, 12)$, $(8, 8)$ and $(5, 5)$ respectively. The three modes have the same prior probability. From this distribution, we generate 4 sample sets at size 60, 150, 300 or 600. The product kernels are used to estimate the density function. In the tests, we examine different kernel bandwidth: 2.5, 5, 10 or h_i^* where h_i^* is an automatically selected bandwidth according to [Scott, 1992]

$$\hat{h}_i^* = \hat{\sigma}_i N^{-\frac{1}{d+4}} \tag{2.7}$$

where $\hat{\sigma}_i$ is the empirical standard variance of variable x_i. It is evident from the figure that a large bandwidth tends to provide a global picture of the distribution but has the potential to over-smooth the data. On the other hand, a small bandwidth emphasizes local structures but may result in some false modes not really present in the true density. Other important points include: a) a larger sample set tends to produce more accurate estimation, b) the automatically selected kernel bandwidth h_i^* yields favorable estimation

across various size of sample sets.

The present work tentatively puts emphasis on fixed-bandwidth kernel density estimation. It is possible that variable-bandwidth kernels will extend the current work especially for true density functions with quite complicated local structures [Terrell and Scott, 1992]. However, variable kernel estimation remains challenging [Devroye and Lugosi, 2000] and is beyond the scope of this study.

2.3 The Spatial-Spectral Representation Model

The goal of tracking is to find and determine the state of target objects that appear similar to given models through an image sequence. From the perceptual point of view, the appearance features that distinguish the objects are characterized by their particular color and texture patterns. Therefore, an effective representation model is crucial for the success of visual tracking.

The present work takes a statistical approach to appearance representation. Consider a given object that appears as an image region consisting of a set of pixels $\{\mathbf{x}_i\}$ and the colors $\{\mathbf{u}_i = \mathbf{u}(\mathbf{x}_i)\}$, $i = 1, \ldots, N$. We refer to such an image region as an *observation* denoted by $\Omega = \{(\mathbf{x}_i, \mathbf{u}_i)\}$. To represent Ω, we use kernel density estimation to describe the probability density of a pixel's position and color:

$$f(\mathbf{x}, \mathbf{u}|\Omega) = \frac{\alpha}{N} \sum_{i=1}^{N} k_s(||\mathbf{x} - \mathbf{x}_i||^2) k_u(||\mathbf{u} - \mathbf{u}_i)||^2) \text{ for } (\mathbf{x}_i, \mathbf{u}_i) \in \Omega \tag{2.8}$$

where k_s and k_u are kernel functions with bandwidth h_s and h_u, respectively for spatial and spectral component. Besides, $\alpha = \alpha_s \alpha_u$ is a normalization constant (α_s or α_u is the normalization constant for k_s or k_s) giving $\int f d\mathbf{u} d\mathbf{x} = 1$.

It should be mentioned that a similar kernel-based representation model has been studied in [Elgammal et al., 2003b] where Elgammal et al. used the representation model to formulate a similarity measure and subsequently a tracking system. However, our study shows that their formulation is not suited to address our affine matching problems (see Section 2.5.2).

The probability density across spatial-spectral space essentially characterizes the joint spatial-spectral correlation in the appearance data. And the kernel density estimation, especially with Gaussian kernels for their favorable properties in terms of

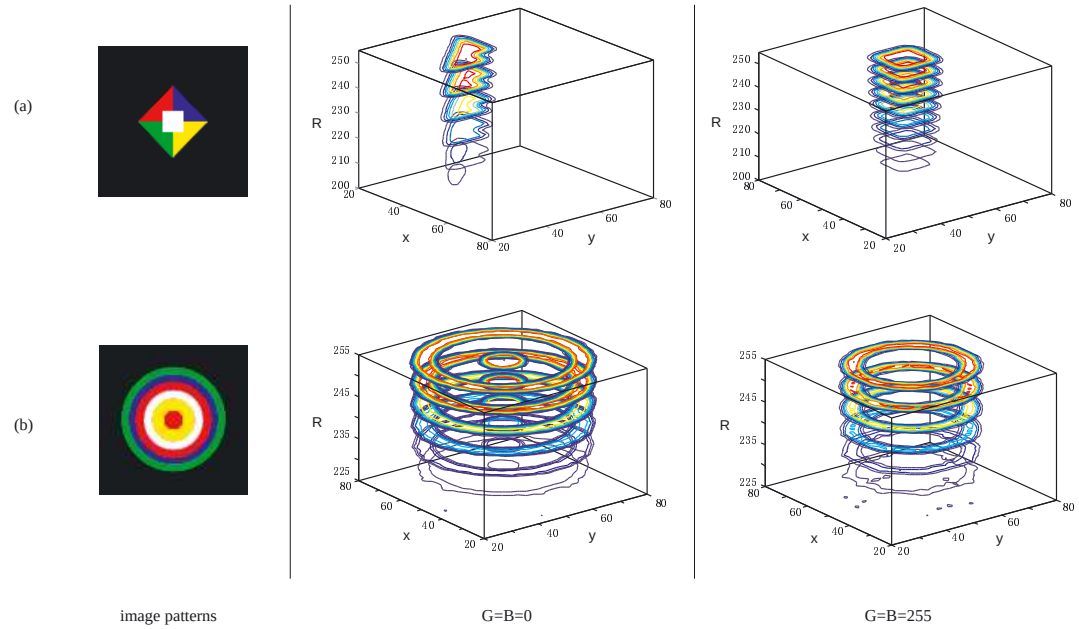


Figure 2.2: Examples of spatial-spectral models for object representation.
The left column displays two objects to be represented, to the right the learned density surfaces as 3D profiles that characterize especially the red and white regions. Red means high and blue means low density.

scalability and differentiability, can produce good approximations to natural distributions [Scott, 1992]. In addition, it allows one to describe different levels of details in the spatial-spectral pattern, by choosing appropriate kernel bandwidth h_s and h_u. For example, it would have an advantage over other models when accurate spatial-spectral representations are of importance for distinguishing the objects.

From [Micchelli, 1986], it is known that no Gaussian can be written as a linear combination of Gaussians centered at other points. It naturally follows that the above representation model is different from another one, unless they correspond to the same appearance data. In other words, the model is effective in identifying and distinguishing the represented object.

Figure 2.2 illustrates two examples to demonstrate how the kernel model can effectively represent visual objects with similar color features. In the left column are shown two synthetic objects. Since their color features are similar, it will be difficult for color-distribution techniques such as color histograms to differentiate between them. Furthermore, the complex concentric structure of the second object will handicap blob

models in correctly describing the special spatial-spectral pattern.

It can be seen from the figure that the proposed representation model can be used to discriminate between the objects despite their close similarity in color distribution. To visualize the estimated density function in the 5-dimensional space $\{x, y, r, g, b\}$, we display two 3D profiles of the density surface for each object. In particular, the middle column plots the profiles of the density functions at $(g = 0, b = 0)$ while the r component is variable, and the density surface at a fixed r value is drawn as a set of contours in the (x, y) plane. Similarly, the right column plots the profiles of the density function at $(g = 255, b = 255)$. These profiles clearly show that the proposed model accurately captured the special spatial-spectral modes of red and white regions. In other words, the model produced special and distinctive representations for the objects by correlating their spatial and spectral features.

2.4 The Similarity Measure

As mentioned earlier, visual tracking is to find an object of similar appearance to a target model through the image sequence. An important component of the tracking process is the similarity measure which tells the likelihood of an observation – called a *target candidate* – Ω_p to be the target Ω_q. Hereafter the target candidate and the target model are represented by the corresponding representation functions $p = f(\mathbf{x}, \mathbf{u}|\Omega_p)$ and $q = f(\mathbf{x}, \mathbf{u}|\Omega_q)$ respectively.

A natural way to define the similarity is to use a l_2 norm measure

$$\begin{aligned} D_0(p, q) &= -\int ||p - q||^2 d\mathbf{u}d\mathbf{x} \\ &= -\int ppd\mathbf{u}d\mathbf{x} - \int qqd\mathbf{u}d\mathbf{x} + 2\int pqd\mathbf{u}d\mathbf{x} \end{aligned} \tag{2.9}$$

It is obvious that the possibly largest value of the similarity measure is zero which happens if and only if the two representations are exactly identical. Besides, the larger the measure, the closer the similarity between q and p.

With the following general relation

$$\begin{aligned} & \exp(-\frac{1}{2\sigma^2}(\xi - \xi_1)^2)\exp(-\frac{1}{2\sigma^2}(\xi - \xi_2)^2) \\ = \; & \exp(-\frac{2}{2\sigma^2}(\xi - \frac{\xi_1 + \xi_2}{2})^2)\exp(-\frac{1}{2\sigma^2}\frac{(\xi_1 - \xi_2)^2}{2}) \end{aligned} \tag{2.10}$$

there is

$$\begin{aligned} pp &= \frac{\alpha^2}{N_p^2}\sum_{i,j} k_s(2||\mathbf{x}-\frac{(\mathbf{x}_i^{(p)}+\mathbf{x}_j^{(p)})}{2}||^2)\cdot k_s(\frac{||(\mathbf{x}_i^{(p)}-\mathbf{x}_j^{(p)})||^2}{2}) \\ &\quad \cdot k_u(2||\mathbf{u}-\frac{\mathbf{u}_i^{(p)}+\mathbf{u}_j^{(p)}}{2}||^2)k_u(\frac{||\mathbf{u}_i^{(p)}-\mathbf{u}_j^{(p)}||^2}{2}) \end{aligned} \tag{2.11}$$

where $\mathbf{x}_i^{(p)}$ $(i=1\ldots N_p)$ or $\mathbf{x}_i^{(q)}$ $(i=1\ldots N_q)$ denotes a pixel from the candidate p or the model q. The integral of pp is then derived as

$$\int pp d\mathbf{u} d\mathbf{x} = \frac{\alpha^2}{2^{\frac{1}{2}+\frac{1}{l}}N_p^2}\sum_{i,j} k_s(\frac{||(\mathbf{x}_i^{(p)}-\mathbf{x}_j^{(p)})||^2}{2})k_u(\frac{||\mathbf{u}_i^{(p)}-\mathbf{u}_j^{(p)}||^2}{2}) \tag{2.12}$$

where l is the length of color feature $\mathbf{u}$.

Applying similar manipulations to the integral of pq, we have

$$\int pq d\mathbf{u} d\mathbf{x} = \frac{\alpha^2}{2^{\frac{1}{2}+\frac{1}{l}}N_pN_q}\sum_{i,j} k_s(\frac{||\mathbf{x}_i^{(p)}-\mathbf{x}_j^{(q)}||^2}{2})k_u(\frac{||\mathbf{u}_i^{(p)}-\mathbf{u}_j^{(q)}||^2}{2}) \tag{2.13}$$

The integral of qq can be obtained in a similar way (here the details are omitted). By canceling the common factor $(\alpha_s\alpha_u/2^{\frac{1}{2}+\frac{1}{l}})$, the similarity measure becomes

$$\begin{aligned} D_0(p,q) &= -\frac{1}{N_p^2}\sum_{i,j} k_s(\frac{||(\mathbf{x}_i^{(p)}-\mathbf{x}_j^{(p)})||^2}{2})k_u(\frac{||\mathbf{u}_i^{(p)}-\mathbf{u}_j^{(p)}||^2}{2}) \\ &\quad +\frac{2}{N_pN_q}\sum_{i,j} k_s(\frac{||\mathbf{x}_i^{(p)}-\mathbf{x}_j^{(q)}||^2}{2})k_u(\frac{||\mathbf{u}_i^{(p)}-\mathbf{u}_j^{(q)}||^2}{2}) \\ &\quad -\frac{1}{N_q^2}\sum_{i,j} k_s(\frac{||(\mathbf{x}_i^{(q)}-\mathbf{x}_j^{(q)})||^2}{2})k_u(\frac{||\mathbf{u}_i^{(q)}-\mathbf{u}_j^{(q)}||^2}{2}) \end{aligned} \tag{2.14}$$

2.5 Matching Objects under Affine Transformation

During tracking, the target object usually keeps moving through consecutive frames, and the dynamics of the object may lead to considerable deformations in the object's image. The deformation is important for video understanding, but it poses a challenging problem to the accurate object representation as well as tracking. Here we set out to study how to address the problem by adapting the above kernel-based representation model to a particular class of image deformation described by affine transformation.

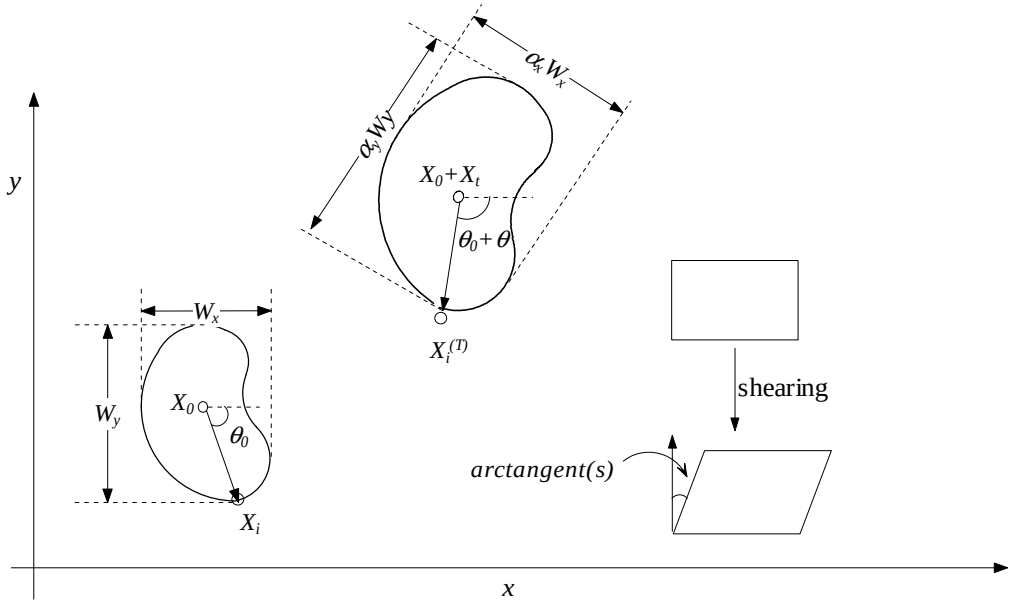


Figure 2.3: Affine Transformation

2.5.1 Affine Transformation

Let's first consider a set of points on a near-planar 3D surface that undergoes rigid body transformation and scaling. Such a transformation is defined by 7 parameters: three parameters to specify a translation, 3 parameters to specify a rotation and one parameter to denote a scaling factor. To simplify, the induced deformation in the 2D image can be described with an affine transformation [Blake and Isard, 1998], which in turn can be written in the form of a combination of 2D geometrical operators involving scaling, rotation, shearing and translation respectively.

Figure 2.3 gives two examples of affine transformation, where a_x and a_y represent the scaling factors, θ denotes the angle of rotation with respect to the center point ($\mathbf{x}_c$), $\mathbf{x}_t$ stands for the translation vector, and s controls the shearing effect that can transform a rectangular into a parallelogram.

These transformation operators can be written in matrix/vector form as

$$\text{Rotation} : \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \quad \text{Scaling} : \begin{pmatrix} a_x & 0 \\ 0 & a_y \end{pmatrix} \tag{2.15}$$

$$\text{Shearing} : \begin{pmatrix} 1 & s \\ 0 & 1 \end{pmatrix} \quad \text{Translation} : \begin{pmatrix} x_t \\ y_t \end{pmatrix} \tag{2.16}$$

In this work, we consider an affine transformation by combining scaling, shearing,

rotation and translation successively. The combination suffices for our purposes, as justified by our empirical study with many real-world tracking tasks. The formulation of the transformation is thus given by

$$\begin{aligned} x_i^{(T)} &= a_x(\hat{x}_i + s\hat{y}_i)\cos\theta - a_y\hat{y}_i\sin\theta + x_t \\ y_i^{(T)} &= a_x(\hat{x}_i + s\hat{y}_i)\sin\theta + a_y\hat{y}_i\cos\theta + y_t \end{aligned} \tag{2.17}$$

or in matrix form as

$$\mathbf{x}_i^{(T)} = M(\mathbf{a}, s, \theta)\hat{\mathbf{x}}_i + \mathbf{x}_t \tag{2.18}$$

Here $\hat{\mathbf{x}}_i = (\hat{x}_i, \hat{y}_i)$ is the relative position of the point $\mathbf{x}_i$ to the center point $\mathbf{x}_c$ of the object: $\hat{\mathbf{x}}_i = \mathbf{x}_i - \mathbf{x}_c$; $\mathbf{x}_i^{(T)}$ denotes the position after transformation; $\mathbf{x}_t$ represents the displacement of the center of the object; and $M(\mathbf{a}, s, \theta)$ stands for the deformation matrix

$$M(\mathbf{a}, s, \theta) = \begin{bmatrix} a_x\cos\theta & -a_y\sin\theta + sa_x\cos\theta \\ a_x\sin\theta & a_y\cos\theta + sa_x\sin\theta \end{bmatrix} \tag{2.19}$$

Without loss of generality, we set the center of the target model at origin, i.e. $\mathbf{x}_c = (0, 0)$. Therefore, $\mathbf{x}_t$ in Eq. (2.18) will represent the center of the deformed object image, and hereafter it is referred to as the position of the object.

2.5.2 Affine Matching with Kernel-based Models

Section 2.4 provides a similarity measure between two object images. When one image is subject to affine transformation, the similarity measure will become an affine matching problem: for a known model Ω_q and an acquired image observation Ω_p, how to describe their similarity with respect to affine transformation?

One may use computers to synthesize a deformed object image from the model Ω_p with each possible $T = \{M, \mathbf{x}_t\}$ and evaluate the similarity between the synthesized image and Ω_q. The method will be computationally expensive. An alternative, efficient approach is to first rewrite the representation model by combining the affine formulation Eq. (2.18) and the representation formulation Eq. (2.8), yielding

$$\begin{aligned} p_{_T}(\mathbf{x}, \mathbf{u}) &= \frac{\alpha}{N}\sum_{i=1}^{N} k_s(||\mathbf{x} - \mathbf{x}_i^{(T)}||^2)k_u(||\mathbf{u} - \mathbf{u}_i)||^2) \\ &= \frac{\alpha}{N}\sum_{i=1}^{N} k_s(||\mathbf{x} - M(\mathbf{a}, s, \theta)\mathbf{x}_i - \mathbf{x}_t||^2)k_u(||\mathbf{u} - \mathbf{u}_i)||^2) \end{aligned} \tag{2.20}$$

where p_T denotes the representation model of Ω_p that has undergone transformation T. It follows that the similarity measure between Ω_p and Ω_q with respect to T becomes

$$\begin{aligned} D_0(T) &= -\frac{1}{N_p^2}\sum_{i,j} k_s(\frac{||M(\mathbf{a},s,\theta)(\mathbf{x}_i^{(p)}-\mathbf{x}_j^{(p)})||^2}{2})k_u(\frac{||\mathbf{u}_i^{(p)}-\mathbf{u}_j^{(p)}||^2}{2}) \\ &\quad +\frac{2}{N_pN_q}\sum_{i,j} k_s(\frac{||M(\mathbf{a},s,\theta)\mathbf{x}_i^{(p)}+\mathbf{x}_t-\mathbf{x}_j^{(q)}||^2}{2})k_u(\frac{||\mathbf{u}_i^{(p)}-\mathbf{u}_j^{(q)}||^2}{2}) \\ &\quad -\frac{1}{N_q^2}\sum_{i,j} k_s(\frac{||(\mathbf{x}_i^{(q)}-\mathbf{x}_j^{(q)})||^2}{2})k_u(\frac{||\mathbf{u}_i^{(q)}-\mathbf{u}_j^{(q)}||^2}{2}) \\ &\triangleq -I_1(p_T,p_T)+I_2(p_T,q)-I_3(q,q) \end{aligned} \tag{2.21}$$

where I_1, I_2 and I_3 represent the three addenda respectively.

In a related work [Elgammal et al., 2003b], Elgammal et al. used a similar kernel-based representation model to formulate a similarity measure and subsequently a tracking system. In specific, they used Kullback-Leibler information distance between regions Ω_p and Ω_q.

$$\begin{aligned} D_e(q||p) &= \int q\log\frac{q}{p}d\mathbf{x}d\mathbf{u} \\ &= \int q\log q d\mathbf{x}d\mathbf{u}-\int q\log p d\mathbf{x}d\mathbf{u} \\ &= -H_q-L_q \end{aligned} \tag{2.22}$$

The first term is the entropy with the distribution q, and the second is the expectation of function $\log p$ under the density q.

The major problem of the method is that the two terms (and their derivatives) are difficult to compute precisely in close form. In their work, Elgammal et al. firstly resorted to approximating the second one with an empirical likelihood given by $L_q \approx \sum_{\{\mathbf{x}_i,\mathbf{u}_i\}\in\Omega_q}\log p(\mathbf{x}_i,\mathbf{u}_i)$ that requires a large number of object pixels for good approximation. As to the entropy term H_q, they divided the entropy into two parts: $H_q(\mathbf{x},\mathbf{u}) = H_q(\mathbf{x})+H_q(\mathbf{u}|\mathbf{x})$. To approximate $H_q(\mathbf{u}|\mathbf{x})$, they assumed that the local distribution of the feature $\mathbf{u}$ at point $\mathbf{x}$ can be approximated as a Gaussian distribution. However, due to the interaction with other feature points in the neighbourhood, the local distribution may not be a simple Gaussian, especially when the kernel bandwidths h_s and h_u are not very small (i.e. strong interactions between neighbouring points are present).

They also made a critical assumption that $H_q(\mathbf{x})$ is invariant upon any region hypothesis. However, it is quite questionable because the shape of the region will clearly determine the entropy. Only if the regions are with the same size and shape, the entropies can be the same. In other words, that similarity measure may apply well to non-deformation object images, but can be seriously deteriorated especially by shearing and scaling operators.

In short, the computation of the similarity measure in [Elgammal et al., 2003b] is difficult and the approximations rely on a few critical assumptions and may not be well-suited for object images under deformation. By contrast, our similarity measure is directly derived from the kernel-based representation model and the formulation of affine transformation in such a way that the computation is straightforward in close form and does not depend on critical assumptions. More importantly, it is easily applicable to images undergoing affine transformations, as will be demonstrated in the next section.

2.6 Properties of Affine Matching

With T varying across the state space of transformation, the above affine matching will produce a similarity hyper-surface $D_0(T)$. In practice, the true transformation T is generally unknown between image patterns, while the system would use affine matching to determine it. For this purpose, the affine matching should be able to indicate the state by having a corresponding maximum on $D_0(T)$.

2.6.1 The Ideal Case

The ideal case here means that the two observations Ω_p and Ω_q correspond to the same object. It implies that they do not include any pixels from the background. In practice, the prerequisite to such observations is a perfect segmentation that remains a very challenging problem and is beyond the scope of this thesis.

We have conducted computer simulations to investigate affine matching in this case (Figure 2.4). Panel (a) draws the object before transformation, and Planel (b) draws the transformed counterpart. Specifically, the object shears by $s = 1$, scales by $\mathbf{a} = (0.85\ 1.25)$, rotates by $\theta = \pi/6$, and finally translates by $\mathbf{x}_t = (8\ 8)$. Panel (b)-(e) plot the similarity surface of the two objects, with each graph for a particular transformation

operator. It can be seen that the similarity surface is smooth and has a maximum corresponding to the true state T^*. It implies that one can correctly determine the transformation T using the affine matching method.

2.6.2 The Real Case

Obtaining an ideal observation mentioned above is not feasible in practice. Instead, it is quite possible to extract a *candidate* Ω_p which covers the target object while including a number of background pixels. In other words, $\Omega_p = \Omega_q^{T^*} \bigcup \Omega_b$, where $\Omega_q^{T^*}$ denotes the deformed target model (N_q pixels) and Ω_b represents the included background region (N_b pixels). We represent $\Omega_q^{T^*}$ by f_{q^*} and Ω_b by f_b (Eq. 2.8).

Since the representation model Eq. (2.8) takes an additive form, we can describe the observed pattern p by

$$p = \frac{1}{N_q + N_b} \left[N_q f_{q^*} + N_b f_b \right] \tag{2.23}$$

For convenient manipulation, we now consider the similarity measure $D_0(p, q_{_T})$ instead of $D_0(p_{_T}, q)$. (Note that due to the physical nature, the two measures are interchangeable for the purposes of matching and tracking.) With the following expression

$$\begin{aligned} ||p - q_{_T}||^2 &= ||\frac{1}{N_q + N_b} \left[N_q f_{q^*} + N_b f_b \right] - q_{_T}||^2 \\ &= \left[q_{_T}^2 - 2 f_{q^*} q_{_T} + f_{q^*}^2 \right] + \Big[2N_b (N_q + N_b)^{-1} (f_{q^*} q_{_T} - f_b q_{_T}) \\ &\quad + (N_q + N_b)^{-2} (N_b^2 f_b^2 + 2N_q N_b f_b f_{q^*} - (2N_q N_b + N_b^2) f_{q^*}^2) \Big] \\ &\triangleq ||f_{q^*} - q_{_T}||^2 + s_a \end{aligned} \tag{2.24}$$

where s_a stands for those enclosed in the second pair of square brackets, we have a new expression of the similarity measure

$$\begin{aligned} D_0(p, q_{_T}) &= \int ||p - q_{_T}||^2 d\mathbf{u} d\mathbf{x} \\ &= D_0(q_{_T}, f_{q^*}) + \int s_a d\mathbf{u} d\mathbf{x} \end{aligned} \tag{2.25}$$

It can be seen that the first addendum is essentially a similarity measure in ideal case by taking the hidden object f_{q^*} as the model and the target model as the candidate. In other words, the first term corresponds to a measure with a minimum at the desired T^*. The second term $\int s_a d\mathbf{u} d\mathbf{x}$, therefore, determines whether or not the similarity measure has a minimum at T^*.

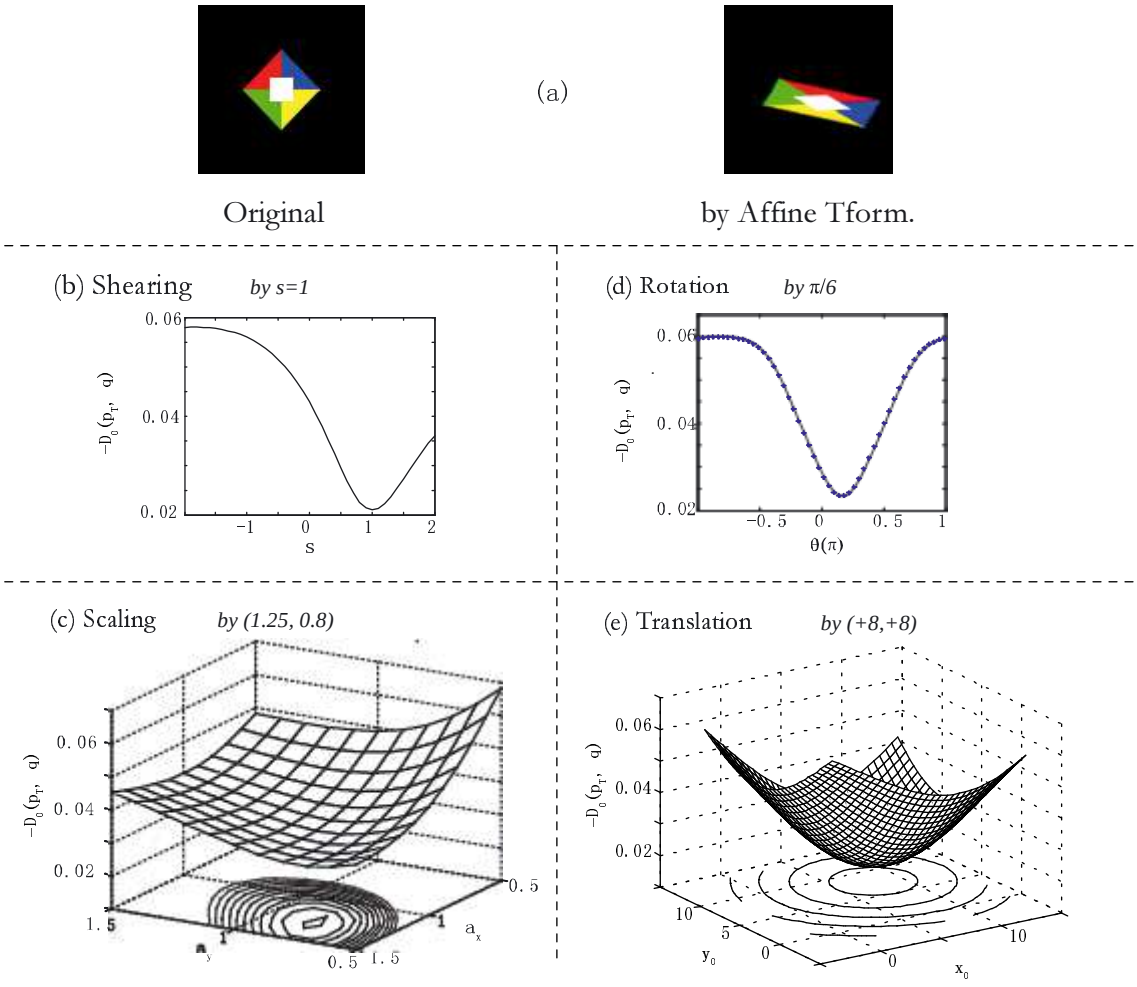


Figure 2.4: Affine matching in an ideal case.

The object and its transformed image are shown in the top row. They are both perfectly segmented from the background, resulting in two ideal observations. Below then are drawn the similarity surfaces with respect to each particular transformation factor. Note that we draw $-D_0$ instead of D_0 in the graphs.

Let's study the derivatives of the term with respect to transformation T. From Eq. (2.24), we have

$$\begin{aligned} \nabla_T \int s_a d\mathbf{u}d\mathbf{x} &= \frac{2N_b}{N_q+N_b}\int (f_{q^*}-f_b)\frac{\partial q_{_T}}{\partial T}d\mathbf{u}d\mathbf{x}+0 \\ &= 2\tau\int (f_{q^*}-f_b)\frac{\partial q_{_T}}{\partial T}d\mathbf{u}d\mathbf{x} \end{aligned} \tag{2.26}$$

where the ratio τ represents the proportion of background pixels in the candidate: $\tau = \frac{N_b}{N_q+N_b}$. Thus, the overall gradient of $D_0(T)$ with respect to T at T^* can be written as

$$\frac{\partial D_0(T)}{\partial T}\Big|_{T=T^*} = 2\tau\,(\int (f_{q^*}-f_b)\frac{\partial q_{_T}}{\partial T}d\mathbf{u}d\mathbf{x})\Big|_{T=T^*} \tag{2.27}$$

The equation shows that the similarity bias induced by background pixels is proportional to τ which serves a similar function to Signal Noise Ratio. In other words, a candidate with fewer background pixels (thus smaller τ) is more likely to produce a correct similarity measure. Consider two types of candidate in (Figure 2.5). The first type, called coarse candidate, is simply a loose region around the predicted target position. The other one called fine candidate uses a region precisely covering the predicted target object. Obviously, the coarse candidate may include a large number of background pixels, while the fine candidate includes possibly only a few background pixels. The fine candidate, however, relies heavily on the precise information about geometrical object state. On the other hand, the coarse one just needs a position prediction close to the true target.

Now consider a simple case in which the background pixels have a uniform distribution in the spectral-spatial space. In other words, $f_b = \frac{1}{\alpha_b}$ where α_b is a normalization constant for $\alpha_b \int f_b d\mathbf{u}d\mathbf{x} = 1$. Consider the similarity bias as Eq. 2.27. There is

$$2\tau\,\frac{\partial \int f_b q_{_T} d\mathbf{u}d\mathbf{x}}{\partial T}\Big|_{T=T^*} \propto \frac{\partial \int q_{_T} d\mathbf{u}d\mathbf{x}}{\partial T}\Big|_{T=T^*} = 0 \tag{2.28}$$

Moreover, because $f_{q^*} = q^*_{_T}$, we have

$$\begin{aligned} 2\tau\,\frac{\partial \int f_{q^*} q_{_T} d\mathbf{u}d\mathbf{x}}{\partial T}\Big|_{T=T^*} &\propto \int q_{_T}\frac{\partial q_{_T}}{\partial T}d\mathbf{u}d\mathbf{x}\Big|_{T=T^*} = \frac{1}{2}\frac{\partial \int q_{_T}^2 d\mathbf{u}d\mathbf{x}}{\partial T}\Big|_{T=T^*} \\ &\propto \frac{\partial\left[\sum_{i,j} k_s\left(||M(\mathbf{x}_i-\mathbf{x}_j)||^2/2\right) k_u\left(||\mathbf{u}_i-\mathbf{u}_j||^2/2\right)\right]}{\partial T}\Big|_{T=T^*} \end{aligned} \tag{2.29}$$

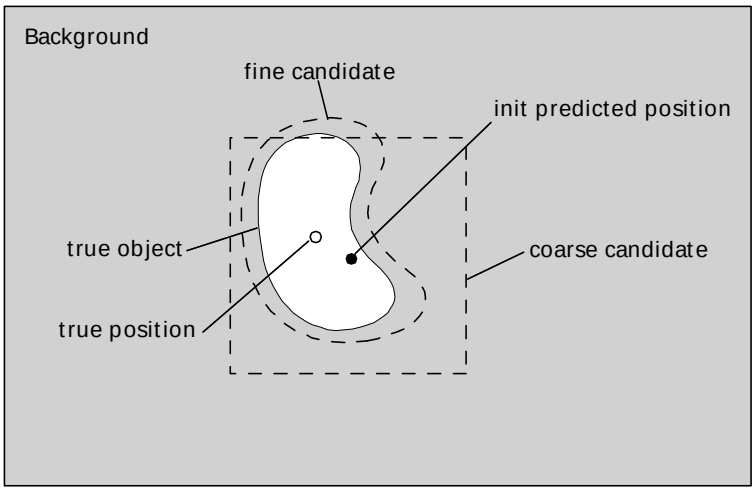


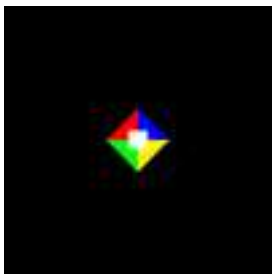
Figure 2.5: Two types of candidate for Tracking.

Although the term still takes a rather complex form, it is clearly irrelevant to the translation vector $\mathbf{x}_t$. Hence, background interference induced by a uniform-distribution Ω_b will have a trivial effect on affine matching with respect to translation.

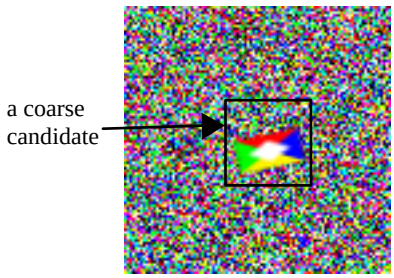
We have investigated the effect of background interference in the similarity measure with simulations. Figure 2.6 shows the results with coarse candidates. Panel (b) plots an extracted candidate as a bounding box centered at the predicted target position $\mathbf{x}_t = (-4, -4)$ (while the true position is given by $\mathbf{x}_t^* = (0, 0)$). It can be seen that a large portion of the candidate is background pixels (τ is around 0.6). The corresponding similarity measure (a hypersuface) is shown in Panel (e) on translation vector $\mathbf{x}_t$, and it has a minimum at $\mathbf{x}_t^*$ despite the background interference. The similarity measure on rotation angle θ are plotted in Panel (c). The curves were generated from the candidates extracted at $\mathbf{x}_t = (-4, -4)$, $\mathbf{x}_t = (-2, -2)$, $\mathbf{x}_t = (0, 0)$, respectively. It is evidence that with the predicted position in a small error range (e.g. $x_t - x_t^* < 4$), the candidate could produce a similarity measure that has a maximum at the desired T^*. Similar results are obtained on affine matching with respect to shearing.

However, it is evidence that the similarity measure on scaling is sensitive to background interference (Panel (d)). With a coarse candidate even at the true target position, i.e. $\mathbf{x}_t = (0, 0)$, the minimum of the similarity measure largely strayed from T^*.

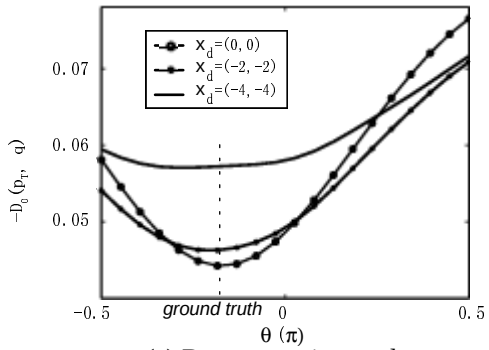
Favorably, additional simulations demonstrate how fine candidates can produce cor-



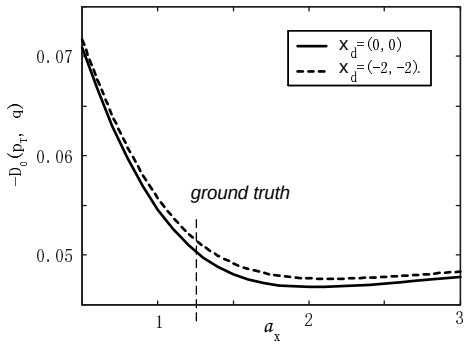
(a) The original object



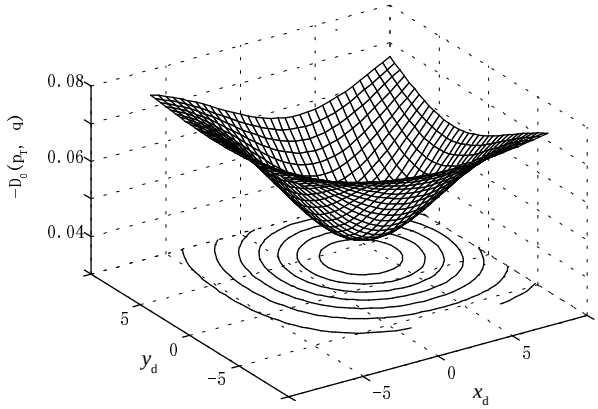
(b) Transformation with cluttering background



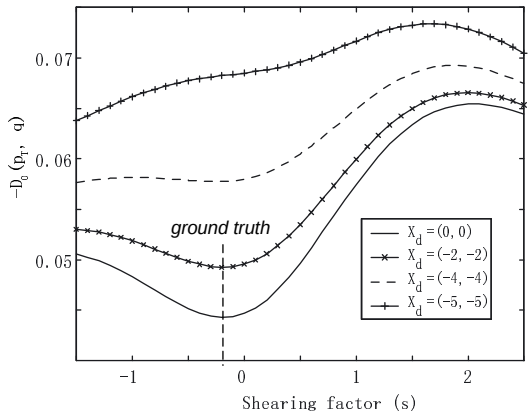
(c) D over rotation angle



(d) D over a scaling factor



(e) D over translation parameters



(f) D over shearing factor

Figure 2.6: Affine matching in real case.

$\mathbf{x}_d$ *represents the difference between a predicted position and the true position. Note that we draw* $-D_0$ *instead of* D_0 *in the graphs.*

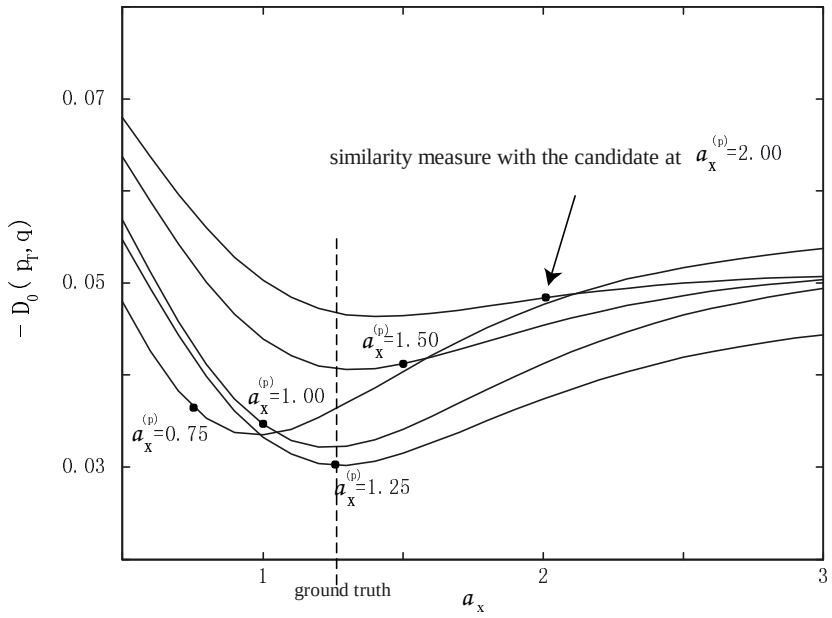


Figure 2.7: Similarity surfaces with various scaling factors.

The figure draws the similarity measures with 5 candidates at $\alpha_x^{(p)}$ =0.75, 1, 1.25, 1.5 or 2, where $\alpha_x^{(p)}$ denotes a predicted scaling factor which is plotted as a black dot on the similarity measure curve. Note that we draw $-D_0$ instead of D_0 in the graphs.

rect similarity measures on scaling. Five candidates at predicted scaling factor ($a_x^{(p)}$ =0.75, 1.0, 1.25, 1.5 or 2.0, true scaling factor a_x^* is 1.25) were used, resulting in 5 different similarity curves shown in Figure 2.7. It can be seen that, with the candidates close to the true object in scaling (e.g. with $a_x^{(p)}$ from 1 to 1.5), the similarity measure has a minimum precisely at a_x^*. The simulation results also suggest that one can use the similarity measure with a candidate at a given scaling factor to obtain a more accurate estimation of the scaling factor. Intuitively, deploying this method iteratively will eventually lead the system to finding the true scaling factors.

The results of the above investigation can be summarized as below:

1. When an ideal candidate is provided that corresponds exactly to the target object, the similarity surface produced by affine matching accurately reflects the ground truth.

2. In real cases where perfect candidates are not accessible, affine matching may be seriously interfered by involved background pixels. Nevertheless, affine matching

with respect to translation is still consistent and robust.

3. In real cases, affine matching with respect to rotation and shearing can be affected to some extent by background interference. If one knows the position of the object, nevertheless, better candidates can be extracted to produce good affine matching.

4. In real cases, affine matching with respect to scaling is sensitive to background interference. A feasible remedy is to resort to fine candidates.

2.7 Summary

This chapter proposes a novel affine matching technique based on a kernel-based representation model. The representation model uses kernel density estimation to characterize the spatial and spectral correlation in a single image of an object, and has given rise to a spatially-and-spectrally smooth similarity measure. The similarity measure has been adapted to visual objects under affine transformation, yielding an affine matching technique. In addition, a few computer simulations have demonstrated the performance of the matching technique in various cases.

The kernel-based model and the affine matching technique indeed provide a basis for a robust visual tracking approach to be elaborated in the next chapter. And their performance will be demonstrated by a variety of experiments therein.

Chapter 3

Visual Affine Tracking

3.1 Introduction

As discussed in the last chapter, the thesis considers an important type of transformation called affine transformation that is able to describe the image variations of near-planar objects' 3D movements. And the last chapter has proposed an affine matching model based on a kernel-based representation model. A problem naturally arises: how to efficiently search for a target object and determine its transformation state by using the kernel based representations?

This chapter follows the kernel-based object modeling scheme, and focuses on how to use the model to infer the transformation state of an object in a given image. We first extend the kernel-based affine matching model to formulate the objective of affine tracking (Section 3). Next, we derive an analytical optimization procedure to maximize the similarity between an observation and a given target model (Section 4). Moreover, a practical tracking algorithm (Section 5) is developed by incorporating the knowledge about the properties of affine matching (elaborated in the last Chapter) into the iterative maximization procedure. We also discuss the computational complexity and efficient implementation of the algorithm.

We conduct extensively experimental study on the proposed method. Using computer-generated image sequences, we examine the robustness of the method against image noise (Section 6). Moreover, we assess the tracker with a variety of real-life objects such as faces, hands, cars and camouflaged tanks (Section 7). The experimental results are positive and convincing. The last section discusses why it is important to use explicit

physical operators (regarding scaling, rotation, shearing and translation) in the affine tracking system (Section 8). We show that an affine model without explicitly accounting for physical operators would hardly give rise to practical tracking algorithms.

3.2 Related Work

As we have mentioned earlier, visual tracking is of paramount importance in many scientific and engineering fields such as surveillance and targeting [Collins et al., 2000], motion capture and recognition from motion [Aggarwal and Cai, 1999]. In principle, it means to determine the dynamic state of one or multiple objects when a set of observations – image frames – become available on-line.

To recover the dynamics of an target object, one may start by describing the evolution of the object and its image by a state-space model which consists of a state transition model (i.e. dynamics model) and a measurement model (i.e. observation model).

$$\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{v}_{t-1}) \tag{3.1}$$

$$\mathbf{y}_t = \mathbf{h}(\mathbf{x}_t, \mathbf{n}_t) \tag{3.2}$$

Here $\mathbf{f}$ represents the state transition function, and $\mathbf{h}$ represents the measurement function. $\mathbf{y}_t$ denotes an observation such as an acquired image. The state transition process and the measurement process are both subject to noise $\mathbf{v}_{t-1}$ or $\mathbf{n}_t$ that represents uncertainties in the dynamics or the measurement. The state model often assumes a first order Markov state process $\{\mathbf{x}_t\}$ which means that, given the previous state of an object, its present state is independent upon earlier history.

At time t, the objective of tracking is to estimate the object's present state $\mathbf{x}_t$ by inferring from the present observation $\mathbf{y}_t$ and the target's previous state $\mathbf{x}_{t-1}$ (with Markovian assumption). The tracking process is essentially a fusion of two inferences: the inference from history using the dynamics model $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, and the inference from the present observation using the observation model whose probabilistic form $p(\mathbf{y}_t|\mathbf{x}_t)$ is called the likelihood.

The inference fusion can be carried out with recursive Bayesian filtering. Particularly if the state transition model and the measurement model are both linear and Gaussian,

Kalman filter provides a natural and perfect solution [Bar-Shalom and Fortmann, 1988]. The technique has been widely used in a number of visual applications such as tracking contours [Blake and Isard, 1998], vehicles [Boykov and Huttenlocher, 2000] and faces [Qian et al., 1998]. Moreover, a variant of the Kalman technique called extended Kalman filter was introduced to address low-order nonlinear filtering problems. More importantly for visual tracking, a sequential Monte Carlo technique called particle filtering was introduced to dealing with complex dynamics [Isard and Blake, 1996], since then the visual tracking field has seen an explosion of interest in particle filters.

The topic of filtering is well rooted in control theory, while the topic of observation modeling is more relevant to the computer vision discipline. An observation model essentially describes the correlation between the observation and the state, thus allowing one to infer the object's state from perceived images. With effective representation models in hand, one may accomplish tracking even without using sophisticated filtering programs. Moreover in many applications, changes in location and appearance of a target object are small across consecutive frames, and this fact suggests one to use special procedures to search for the object in a certain state subspace around a predicted state of the object.

An underlying problem is, the searching-based tracking scheme may be very computationally expensive, especially when the object's state space is of high dimension (i.e. a high degree-of-freedom object) that gives rise to a huge state space. Thus, one needs to develop efficient methods such as gradient based schemes [Bascle and Deriche, 1995, Hager and Belhumeur, 1996] for seeking the objects in the state space.

This chapter concentrates on efficient tracking methods for searching the affine transformation states of target objects. Tracking affine movements is in fact a topic that has been extensively studied. For instance, [Vacchetti et al., 2004] presented a method that tracks an object in 3D space by tracking local images subject to affine transformation. Moreover, Ferrari proposed a system that uses edge and texture information to determine the affine transformation of simple planar shapes [Ferrari et al., 2001]. Bascle and Deriche introduced a method to track complex shapes, by combining deformable contours and deformable region models [Bascle and Deriche, 1995].

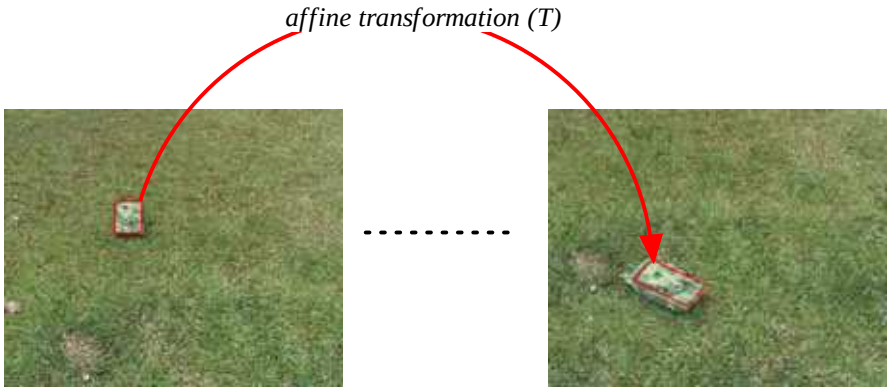


Figure 3.1: An affine tracking problem

However, existing affine tracking systems do not address precise spatial-spectral representation and its extension to efficient searching program. In many situations, a precise spatial-spectral representation may provide more reliable clues for inference than e.g. simple edges and colors. Furthermore, it can offer a good description of textures. As Landy put it [Landy, 1996], the occurrence of texture in a scene is useful in a number of ways: the characteristics of a single texture may be used to identify the surface material [Beck et al., 1983]; texture segregation can be an important component in identifying objects [Yan et al., 2004, Zhu et al., 2001]; characteristics of the texture in the image may be used to infer properties of the 3D layout of objects and object shape (shape-from-texture) [Todd and Akerstrom, 1987, Suen and Healey, 2000].

3.3 Extending Kernel-based Affine Matching to Tracking

Consider tracking a tank shown in Figure 3.1. At frame t, we know the previous state (T^*_{t-1}) of the tank plus the given target model – a reference image of the object, and we need to compute the present tank state T^*_t by inferring from its present appearance – here a quadrangle image region.

However, the tank appearance in the form of a region in the image I_t is hidden to the tracking system. How to extract the appearance, therefore, becomes a critical problem. It can be seen that if the underlying state T^*_t is known, one can easily extract the appearance by projecting the model to the present image and picking out those pixels covered by the model. Consider an object state T and a model Ω_q. We use the

following criterion to select the pixels belonging to the object.

$$\Omega_p(T) : \{\mathbf{x}_i, \mathbf{u}_i | T^{-1}(\mathbf{x}_i) \in \Omega_q\} \tag{3.3}$$

Thus the candidate region Ω_p is a function on T.

From an image region Ω_p with T to the target model Ω_q, the similarity measure D_0 is given by Eq. 2.9 and here is a copy:

$$\begin{aligned} D_0(T(\Omega_p), \Omega_q) &= -\frac{1}{N_p^2}\sum_{i,j} k_s(\frac{||M(\mathbf{a}, s, \theta)(\mathbf{x}_i^{(p)} - \mathbf{x}_j^{(p)})||^2}{2}) k_u(\frac{||\mathbf{u}_i^{(p)} - \mathbf{u}_j^{(p)}||^2}{2}) \\ &\quad + \frac{2}{N_p N_q}\sum_{i,j} k_s(\frac{||M(\mathbf{a}, s, \theta)\mathbf{x}_i^{(p)} + \mathbf{x}_t - \mathbf{x}_j^{(q)}||^2}{2}) k_u(\frac{||\mathbf{u}_i^{(p)} - \mathbf{u}_j^{(q)}||^2}{2}) \\ &\quad - \frac{1}{N_q^2}\sum_{i,j} k_s(\frac{||(\mathbf{x}_i^{(q)} - \mathbf{x}_j^{(q)})||^2}{2}) k_u(\frac{||\mathbf{u}_i^{(q)} - \mathbf{u}_j^{(q)}||^2}{2}) \\ &\triangleq -I_1(p_T, p_T) + I_2(p_T, q) - I_3(q, q) \end{aligned} \tag{3.4}$$

where I_1, I_2 and I_3 represent the three addenda on the right hand respectively. The components of T including M and $\mathbf{x}_t$ refer to Section 2.5.1.

We denote by $D_0(T)$ the similarity between an image region Ω_p with T and a target model Ω_q. Generally, the object's true appearance Ω_{T} among all possible image regions in the present frame has a closest similarity D_0^* to the target model Ω_p. Hence, we formulate the affine tracking as below.

For a target model Ω_q and an image I, affine tracking aims to find the affine transformation state T^* which corresponds to via Eq. 3.32 the true object image region $\Omega_p = \Omega_{T}$ which has, among all possible regions in I, the maximal similarity D_0^* to the model.

$$T^* = \underset{T}{argmax}\{D_0(T(\Omega_p), \Omega_q) | \Omega_p, \Omega_q\} \tag{3.5}$$

Clearly, the problem demands efficient searching procedures to find such a T^*. In the below we first consider a simplified problem: how to compute the affine transformation T^* so as to maximize the similarity between a model and a given observation Ω_p.

3.4 The Optimization Procedure

Given a model Ω_q and an observation (a *target candidate*) Ω_p, because the term $I_3 = \int qqd\mathbf{u}d\mathbf{x}$ in the similarity measure (Eq. 3.4) is fixed, we can simplify the similarity

formulation by

$$D(p_T, q) = -I_1(p_T, p_T) + I_2(p_T, q) \tag{3.6}$$

With a given target candidate Ω_p, a possible way to maximize the similarity $D(p_T, q)$ is by achieving

$$\nabla D(p_T, q) = \frac{\partial D}{\partial \mathbf{x}_t}\Delta\mathbf{x}_t + \frac{\partial D}{\partial \theta}\Delta\theta + \frac{\partial D}{\partial \mathbf{a}}\Delta\mathbf{a} + \frac{\partial D}{\partial s}\Delta s = 0 \tag{3.7}$$

which implies that

$$\nabla_{\mathbf{x}_t} = 0; \quad \nabla_{\mathbf{a}} = 0; \quad \nabla_{\theta} = 0; \quad \nabla_{s} = 0 \tag{3.8}$$

are to be satisfied. Now, we derive the respective solutions.

3.4.1 Computing Translation Vector $\mathbf{x}_t$

From Eq. (3.4), it can be seen that $I_1(p_T, q)$ is independent of $\mathbf{x}_t$. So there is

$$\begin{aligned}
\nabla_{\mathbf{x}_t} D(p_T, q) &= -\nabla_{\mathbf{x}_t} I_1(p_T, p_T) + \nabla_{\mathbf{x}_t} I_2(p_T, q) \\
&= 0 + \nabla_{\mathbf{x}_t}\left(\sum_{i,j}\frac{2}{N_p N_q} k_s(\frac{||M(\mathbf{a}, s, \theta)\mathbf{x}_i^{(p)} + \mathbf{x}_t - \mathbf{x}_j^{(q)}||^2}{2}) k_u(\frac{||\mathbf{u}_i^{(p)} - \mathbf{u}_j^{(q)}||^2}{2})\right) \\
&= 2\sum_{i,j} w_{ij}(\mathbf{x}_t + M(\mathbf{a}, s, \theta)\mathbf{x}_i^{(p)} - \mathbf{x}_j^{(q)})
\end{aligned} \tag{3.9}$$

where the weight w_{ij} is given by

$$w_{ij} = \frac{2}{N_p N_q}\, g_s(\frac{||M(\mathbf{a}, s, \theta)\mathbf{x}_i^{(p)} + \mathbf{x}_t - \mathbf{x}_j^{(q)}||^2}{2}) k_u(\frac{||\mathbf{u}_i^{(p)} - \mathbf{u}_j^{(q)}||^2}{2}) \tag{3.10}$$

with $g_s(\cdot)$ representing the derivative $\partial k_s(x)/\partial x$. If k_s is a Gaussian kernel function, g_s would also take a Gaussian form.

Therefore, $\nabla_{\mathbf{x}_t} D(p_T, q) = 0$ leads to the following solution.

$$\mathbf{x}_t^* = \frac{\sum_{i,j} w_{ij}(\mathbf{x}_j^{(q)} - M(\mathbf{a}, s, \theta)\mathbf{x}_i^{(p)})}{\sum_{i,j} w_{ij}} \tag{3.11}$$

Since $\mathbf{x}_t$ is involved in the weights $\{w_{ij}\}$ (see Eq. 3.10) on the right side, this is indeed an iterative solution for computing the translational vector $\mathbf{x}_t$.

3.4.2 Computing Rotation Angle θ

Consider the partial derivative of $I_1(p_T, p_T)$ with respect to θ. Let

$$\begin{aligned} f_{ij}^{(1)} &\triangleq ||M(\mathbf{a}, s, \theta)(\mathbf{x}_i^{(p)} - \mathbf{x}_j^{(p)})||^2 \\ &= a_x^2(x_i^{(p)} + sy_i^{(p)} - x_j^{(p)} - sy_j^{(p)})^2 + a_y^2(y_i^{(p)} - y_j^{(p)})^2 \end{aligned} \tag{3.12}$$

which is independent upon θ. From Eq. (3.4), we have

$$\frac{\partial I_1(p_T, p_T)}{\partial \theta} = \frac{1}{N_p^2}\sum_{i,j}\frac{\partial k_s(f_{ij}^{(1)}/2)}{\partial \theta}k_u(\frac{||\mathbf{u}_i^{(p)} - \mathbf{u}_j^{(p)}||^2}{2}) = 0 \tag{3.13}$$

So $I_1(p_T, p_T)$ is also independent upon θ.

Consider the partial derivative of $I_2(p_T, q)$ with respect to θ. Let

$$\hat{\mathbf{x}}_j^{(q)} = \mathbf{x}_j^{(q)} - \mathbf{x}_t, \;\; \mathbf{x}_i^{(a)} = M(\mathbf{a}, s, \theta)\mathbf{x}_i^{(p)} \tag{3.14}$$

and

$$f_{ij}^{(2)} = ||\mathbf{x}_i^{(a)} - \hat{\mathbf{x}}_j^{(q)}||^2 \tag{3.15}$$

Then there is

$$\begin{aligned} f_{ij}^{(2)} &= (x_i^{(a)} - \hat{x}_j^{(q)})^2 + (y_i^{(a)} - \hat{y}_j^{(q)})^2 \\ &= (2a_y y_i^{(p)}\hat{x}_j^{(q)} - 2a_x(x_i^{(p)} + sy_i^{(p)})\hat{y}_j^{(q)})\sin\theta \\ &\quad +(-2a_x(x_i^{(p)} + sy_i^{(p)})\hat{x}_j^{(q)} - 2a_y y_i^{(p)}\hat{y}_j^{(q)})\cos\theta \\ &\quad +(a_x^2(x_i^{(p)} + sy_i^{(p)})^2 + a_y^2 y_i^{(p)2} + \hat{x}_j^2 + \hat{y}_j^2) \\ &\equiv a_{ij}sin\theta + b_{ij}cos\theta + c_{ij} \end{aligned} \tag{3.16}$$

where c_{ij} is a variable independent upon θ.

Now, from Eq. (2.13) there is

$$\frac{\partial I_2(p_T, q)}{\partial \theta} = \frac{2}{N_p N_q}\sum_{i,j}\frac{\partial k_s(f_{ij}^{(2)}/2)}{\partial \theta}k_u(\frac{||\mathbf{u}_i^{(p)} - \mathbf{u}_j^{(q)}||^2}{2}) \tag{3.17}$$

$$= \sum_{ij} w_{ij}\frac{\partial f_{ij}^{(2)}}{\partial \theta}$$

$$= \beta_1 cos\theta + \beta_2 sin\theta \tag{3.18}$$

Here w_{ij} has been given in Eq. 3.10, and $\beta 1$, $\beta 2$ are given by

$$\beta 1 = \sum_{i,j} w_{ij}a_{ij} \;,\; \beta 2 = \sum_{i,j} -w_{ij}b_{ij} \tag{3.19}$$

Therefore, $\nabla_\theta D(p_T, q) = 0$ would lead to $\beta_1 cos\theta + \beta_2 sin\theta = 0$. And the solution is

$$\theta^* = -sin^{-1}(\frac{\beta_1}{\sqrt{\beta_1^2 + \beta_2^2}}) \tag{3.20}$$

Since in fact $\beta_{1,2}$ on the right side indirectly involve (via w_{ij} in Eq. 3.10) the parameter θ, the solution implies an iterative procedure to achieving $\nabla_\theta D(p_T, q) = 0$. Besides, because of the property of arcsine, Eq. (3.20) would suggest two values: θ and $\theta + \pi$. In practice we can choose the one that produces relatively larger similarity distance $D(p_T, q)$.

3.4.3 Computing Scaling Factors a

Let's first consider the partial derivatives of $I_1(p_T, p_T)$ with respect to $\mathbf{a}$. By introducing Eq. (3.12) to Eq. (2.12) and Eq. (2.13), we have

$$\begin{aligned}
\nabla_{a_x} I_1(p_T, p_T) &= \nabla_{a_x} \left(\frac{1}{N_p^2} \sum_{i,j} k_s(\frac{f_{ij}^{(1)}}{2}) k_u(\frac{||\mathbf{u}_i^{(p)} - \mathbf{u}_j^{(p)}||^2}{2}) \right) \\
&= \sum_{i,j} \frac{1}{2} v_{ij} \frac{\partial f_{ij}^{(1)}}{\partial a_x} = \sum_{i,j} v_{ij} (x_i^{(p)} + s y_i^{(p)} - x_j^{(p)} - s y_j^{(p)})^2 a_x
\end{aligned} \tag{3.21}$$

and

$$\begin{aligned}
\nabla_{a_y} I_1(p_T, p_T) &= \nabla_{a_y} \left(\frac{1}{N_p^2} \sum_{i,j} k_s(\frac{f_{ij}^{(1)}}{2}) k_u(\frac{||\mathbf{u}_i^{(p)} - \mathbf{u}_j^{(p)}||^2}{2}) \right) \\
&= \sum_{i,j} \frac{1}{2} v_{ij} \frac{\partial f_{ij}^{(1)}}{\partial a_y} = \sum_{i,j} v_{ij} (y_i^{(p)} - y_j^{(p)})^2 a_y
\end{aligned} \tag{3.22}$$

where v_{ij} is given by

$$v_{ij} = \frac{1}{N_p^2} g_s(\frac{f_{ij}^{(1)}}{2}) k_u(\frac{||\mathbf{u}_i^{(p)} - \mathbf{u}_j^{(p)}||^2}{2}) \tag{3.23}$$

Similarly, for $I_2(p_T, q)$ we have

$$\begin{aligned}
\nabla_{a_x} I_2(p_T, q) &= \nabla_{a_x} \left(\frac{2}{N_p N_q} \sum_{i,j} k_s(\frac{f_{ij}^{(2)}}{2}) k_u(\frac{||\mathbf{u}_i^{(p)} - \mathbf{u}_j^{(p)}||^2}{2}) \right) \\
&= \sum_{ij} \frac{1}{2} w_{ij} \frac{\partial f_{ij}^{(2)}}{\partial a_x} \\
&= \sum_{ij} w_{ij} \Big[(x_i^{(p)} + s y_i^{(p)})^2 a_x - \sin\theta (x_i^{(p)} + s y_i^{(p)}) \hat{y}_j^{(q)} \\
&\quad - \cos\theta (x_i^{(p)} + s y_i^{(p)}) \hat{x}_j^{(q)} \Big]
\end{aligned} \tag{3.24}$$

and

$$\begin{aligned}\nabla_{a_y} I_2(p_T, q) &= \nabla_{a_y}\left(\frac{2}{N_p N_q}\sum_{i,j} k_s(\frac{f_{ij}^{(2)}}{2}) k_u(\frac{||\mathbf{u}_i^{(p)} - \mathbf{u}_j^{(p)}||^2}{2})\right) \\ &= \sum_{ij}\frac{1}{2} w_{ij}\frac{\partial f_{ij}^{(2)}}{\partial a_y} \\ &= \sum_{ij} w_{ij}(y_i^{(p)2} a_y + \sin\theta y_i^{(p)}\hat{x}_j^{(q)} - \cos\theta y_i^{(p)}\hat{y}_j^{(q)}) \end{aligned} \tag{3.25}$$

where w_{ij} is given by Eq. (3.10).

Therefore, for $\nabla_{\mathbf{a}} D(p_T, q) = \nabla_{\mathbf{a}}(-I_1(p_T, q) + I_2(p_T, q)) = 0$ we have

$$a_x^* = \frac{\sum_{ij} w_{ij}(\sin\theta(x_i^{(p)} + s y_i^{(p)})\hat{y}_j^{(q)} + \cos\theta x_i^{(p)}\hat{x}_j^{(q)})}{\left(\sum_{ij} w_{ij}(x_i^{(p)} + s y_i^{(p)})^2\right) - \sum_{ij} v_{ij}(x_i^{(p)} + s y_i^{(p)} - x_j^{(p)} - s y_j^{(p)})^2} \tag{3.26}$$

$$a_y^* = \frac{\sum_{ij} w_{ij}(\cos\theta y_i^{(p)}\hat{y}_j^{(q)} - \sin\theta y_i^{(p)}\hat{x}_j^{(q)})}{\left(\sum_{ij} w_{ij} y_i^{(p)2}\right) - \sum_{ij} v_{ij}(y_i^{(p)} - y_j^{(p)})^2} \tag{3.27}$$

Since $a_{x,y}$ is involved in both w_{ij} and v_{ij} on the right sides, the equations imply an iterative procedure to computing scaling factors.

3.4.4 Computing Shearing Factor s

Consider now the partial derivatives of $I_1(p_T, p_T)$ with respect to s. By introducing Eq. (3.12) to Eq. (2.12) and Eq. (2.13), we have

$$\begin{aligned}\nabla_s I_1(p_T, p_T) &= \nabla_s\left(\frac{1}{N_p^2}\sum_{i,j} k_s(\frac{f_{ij}^{(1)}}{2}) k_u(\frac{||\mathbf{u}_i^{(p)} - \mathbf{u}_j^{(p)}||^2}{2})\right) = \sum_{i,j}\frac{1}{2} v_{ij}\frac{\partial f_{ij}^{(1)}}{\partial s} \\ &= \sum_{i,j} v_{ij}\left[(y_i^{(p)} - y_j^{(p)})^2 s - a_x^2(x_i^{(p)} - x_j^{(p)})(y_i^{(p)} - y_j^{(p)})\right] \\ &\equiv z_1 s + z_2 \end{aligned} \tag{3.28}$$

Consider $I_2(p_T, q)$. We first rewrite $f_{ij}^{(2)}$ from Eq. (3.16) as

$$\begin{aligned} f_{ij}^{(2)} &= (a_x^2 y_i^{(p)2}) s^2 + (2a_x^2 x_i^{(p)} x_y^{(p)} - 2a_x y_i^{(p)}\hat{y}_j^{(q)}\sin\theta - 2a_x y_i^{(p)}\hat{x}_j^{(q)}\cos\theta) s \\ &\quad + \Big[(a_y y_i^{(p)}\hat{x}_j^{(q)} - 2a_x x_i(p)\hat{y}_j^{(q)})\sin\theta + (2a_x x_i^{(p)}\hat{x}_j^{(q)} - 2a_y y_i(p)\hat{y}_j^{(q)})\cos\theta + \\ &\quad a_x^2 x_i^{(p)2} + a_y^2 y_i^{(p)2} + \hat{x}_j^{(q)2} + \hat{y}_j^{(q)2}\Big] \\ &\equiv \mu_1 s^2 + \mu_2 s + \mu_3 \end{aligned} \tag{3.29}$$

It follows that $\nabla_s I_2(p_T, q)$ would become

$$\begin{aligned}
\nabla_s I_2(p_T, q) &= \nabla_s \left(\frac{2}{N_p N_q} \sum_{i,j} k_s(\frac{f_{ij}^{(2)}}{2}) k_u(\frac{||\mathbf{u}_i^{(p)} - \mathbf{u}_j^{(p)}||^2}{2}) \right) \\
&= \sum_{ij} \frac{1}{2} w_{ij} \frac{\partial f_{ij}^{(2)}}{\partial s} \\
&= \sum_{ij} w_{ij} [\mu_1 s + \frac{1}{2}\mu_2] \\
&\equiv z_3 s + z_4 \qquad (3.30)
\end{aligned}$$

Therefore, the solution to $\nabla_s D(p_T, q) = \nabla_s(-I_1(p_T, p_T) + I_2(p_T, q)) = 0$ can be derived as

$$s^* = -\frac{z_2 - z_4}{z_1 - z_3} \qquad (3.31)$$

This solution implies an iterative procedure to computing shearing factors because both $w_i j$ and $v_i j$ on the right side involve the shearing factor s.

3.4.5 Discussion on Optimization

In the above subsections we have derived a set of iterative functions for maximizing the similarity function Eq. 3.7. Though the iterative functions are derived and expressed for a few parameters respectively, it is also straightforward to write the iterations in one joint function $T^* = F_{iter}(T)$ where F_{iter} is a vector of functions. And by using F_{iter} one may update all parameters at once in one iteration step.

However, direct application of the joint iterative function F_{iter} may not lead to the desired tracking results. As will be discussed in the next section, the highly complexity of similarity surfaces in real applications usually would make it inappropriate to use joint-iteration for all parameters at once. Later on in Section 3.10.2 we will see such an example. Instead, we would better use the individual iteration functions (associated with respective physical meanings) to navigate the optimization toward the desired true states on the similarity surfaces.

3.5 The Tracking Algorithm

The previous section derived an optimization procedure to the maximization of similarity measure $D(p_T, q)$, with respect to T, between the model Ω_q and a fixed candidate

Ω_p of object image, where Ω_p is not necessarily the true object image. According to the properties of affine matching (see Chapter 2.6), the procedure would lead to the true transformation state T^*, provided that the observation Ω_p is identical (or sufficiently close) to the true appearance Ω_T of the present object. Recall the affine tracking formulation elaborated in Section 3.3. One needs to search in the image for the true object image Ω_T by optimizing both the candidate Ω_p and the correlated transformation state T. In the below we propose an optimization scheme based on the proposed similarity-maximization procedures and the essential properties of affine matching.

Suppose the object is moving continuously across consecutive frames. In other words, the present object would lie in the vicinity of its previous position in the image. One may simply use the previous state as the initial estimation of the current state. Note that if the system considers dynamics, a more sophisticated prediction of the current state may be available. The initial estimation, though may not be very accurate, still enables one to obtain a *coarse candidate* (see Section 2.6) that can serve as an appropriate candidate for infering the translation (i.e. the object position), using the proposed similarity maximization procedure for computing translational vector. This is justified by the properties of affine matching that affirm the robustness of affine matching with respect to translation. The recovered position information then allows one to extract a *fine candidate* whose affine matching to the model would indicate the correct shearing and rotation state. That's to say, with the fine candidate, one can use the similarity maximization procedures for computing the shearing and rotation state. With the recovered information about translation, shearing and rotation, one can proceed to acquire even better *fine candidates* that finally lead one to the accurate estimation of scaling factors.

It is easy to see that the candidate extraction is a crucial task in the tracking scheme. Here we use a flexible method to extract a candidate: for an target model Ω_q and an estimated transformation T, we extract the candidate defined by both T and a relaxation factor ϵ.

$$\Omega_p : \{\mathbf{x}_i, \mathbf{u}_i | \left(\min_{\mathbf{x}_j \in \Omega_q} ||T^{-1}(\mathbf{x}_i) - \mathbf{x}_j||^2 \right) < \epsilon\} \tag{3.32}$$

Here ϵ controls the size of the neighborhood of the estimated object region. Clearly, larger ϵ will produce a larger (i.e. coarser) candidate, and vise versa.

The tracking method mentioned above is essentially a coarse-to-fine scheme, and it pursues better and better candidates in the searching process. The following details the proposed tracking algorithm. The particular ϵ should be selected according to the true situations involving e.g. the object's size and uncertainties in motion.

1. Initialization: create the object model Ω_q using a sample image; set frame no. $k = 0$ and the object's initial state T_0;

2. Proceed to next frame: $k = k + 1$;

3. Obtain the prediction of the transformation state $T_k = \{\mathbf{x}_0^{(k)}, \theta^{(k)}, s^{(k)}, \mathbf{a}^{(k)}\}$; here we may just set them to $T_k = T_{k-1}$;

4. Estimating the transformation parameters:

   (a) Estimating translation vector $\mathbf{x}_0^{(k)}$:

      i. Extract a candidate region Ω_p according to T_k, with appropriate ϵ (Eq. (3.32));

      ii. Update $\mathbf{x}_0^{(k)}$ by Eq. (3.11);

      iii. If the iteration converges, i.e. $d(\mathbf{x}_0^{(k)}) < \epsilon_x$, where ϵ_x is a preset small value and $d(\mathbf{x}_0^{(k)})$ is the change of $\mathbf{x}_0^{(k)}$ in current iteration step, proceed to Step (b); otherwise go back to Step 4.a.i;

   (b) Estimating rotation angle and shearing factor $\{\theta^{(k)}, s^{(k)}\}$:

      i. Extract a candidate region Ω_p according to T_k, with appropriate ϵ (Eq. (3.32));

      ii. Update $\theta^{(k)}$ by Eq. (3.20);

      iii. Update $s^{(k)}$ by Eq. (3.31);

      iv. If $d(\theta^{(k)}) < \epsilon_\theta$ and $d(s^{(k)}) < \epsilon_s$, proceed to Step (c); otherwise return to Step 4.b.i;

   (c) Estimating scaling factors $\mathbf{a}^{(k)}$:

      i. Extract a candidate region Ω_p according to T_k, with appropriate ϵ (Eq. (3.32));

      ii. Update $\mathbf{a}^{(k)}$ by Eq. (3.26);

iii. If $d(\mathbf{a}^{(k)}) < \epsilon_a$, proceed to Step 5; otherwise go to Step 4.c.i.

5. If (a)(b)(c) yield no change in T_k, go to Step 2 for next frame; otherwise go to Step 4 to further optimize the estimations.

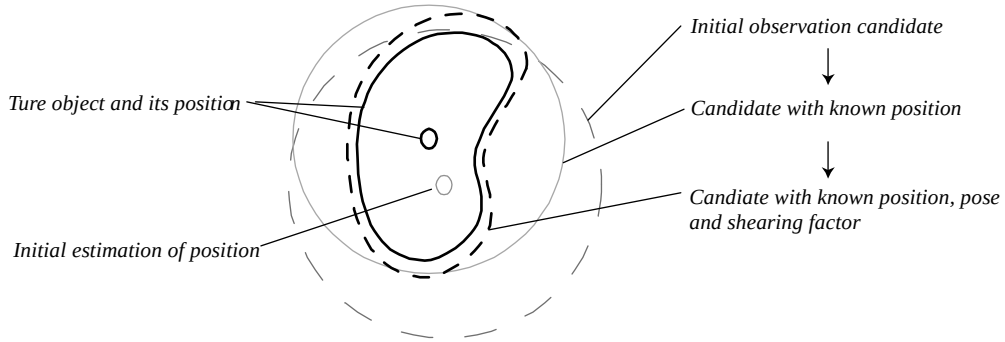


Figure 3.2: Coarse-to-fine affine tracking scheme

Figure 3.2 illustrates the coarse-to-fine algorithm for affine tracking. When a new frame is presented, the tracking system has merely a rough prediction of the object position and pose. Thus, only coarse candidate extraction is feasible. The coarse candidate in turn provides a clue to obtain a more reliable estimation of the object position. The system is then able to extract a better candidate (with possibly fewer background pixels in the candidate) to infer the shearing and rotation parameters. Finally, the system extracts an even better candidate to compute scaling factors. It can be seen that, along with the coarse-to-fine searching, the candidate becomes continuously better in quality.

In brief, the coarse-to-fine search scheme above is able to greatly enhance the tracking system by choosing the right pixels for processing. There are three facts to the advantage of the scheme:

1. Since the target object's region is supposed to be known in the first frame, we can accuratly know the exact shape of the object;

2. Because the object usually undergoes continuous transformations, initial candidate region can be extracted to effectively encompass the true object image while including only a few, if any, background pixels;

3. Real objects usually show coherent spatial-temporal image features. Our object

models are well-suited to describe the features coherences, as they are designed to capture the smooth correlations between the features by using kernels. In addition, they can also follow the continuous changes in the features by using adaptation frame by frame.

3.6 Computational Complexity and Efficient Implementation

Because the algorithm is iterative and it usually takes a few iterations (usually less than 4 in each step) to converge, the overall computational complexity would mainly depend on the cost of each iteration. In Step 4 for computing translation vector, each iteration according to Eq. 3.11 requires $O(N_p N_q)$ computational time (Recall that $N_q(N_p)$ is the number of target(candidate) pixels). Similarly, in Step 5 for computing rotation angle according to Eq. 3.20, the computational cost is approximately given by $O(N_p N_q)$. For computing both scaling factors according to Eq. 3.26 and shearing factor according to Eq. 3.31, it takes additional $O(N_q^2)$ time to complete each iteration. Hence, we conclude that the overall computational cost for scaling/shearing factors is approximately $O(N_p N_q + N_q^2)$.

In the implementation, an efficient scheme can be employed to reduce the computational complexity. Since kernel function yields almost trivial value for those input vectors distant from the kernel centroid, the scheme would omit those vectors in the computation. For instance, the scheme chooses only the pixels j that satisfy $k(||\mathbf{x}_j - \mathbf{x}_i||^2) < \epsilon$ in the real computation of $\mathbf{x}_t^*$ (Eq. 3.11). With a certain Gaussian bandwidth and a certain ϵ (taken as 0.01 in the tests), obviously, the number of these pixels chosen for computation is bounded and the bound can be easily calculated. Suppose this number be η, then the cost of each iteration of computing translation or rotation parameters would be reduced to $O(\eta N_q)$. Similarity, the cost of computing the shearing/scaling parameters would also be reduced to $O(\eta N_p + \eta N_q)$.

In a real implementation on a conventional 1.3GHz Pentium-M PC, the tracking algorithm achieved a frame rate of 2fps for a target object of around 1000 pixels in RGB images.

3.7 Tracking Synthetic Objects

The purpose of this section is to examine the tracking algorithm's performance against image noise, by using a synthetic diamond-shaped object. The object randomly moves through computer-generated image sequences, with each sequence corrupted by additive zero-mean Gaussian noise with a particular variance (σ_n^2). Some images of the object are shown in Figure 3.3, where high level of noise (especially for $\sigma_n > 50$) clearly poses a serious challenge to the estimation of the object state.

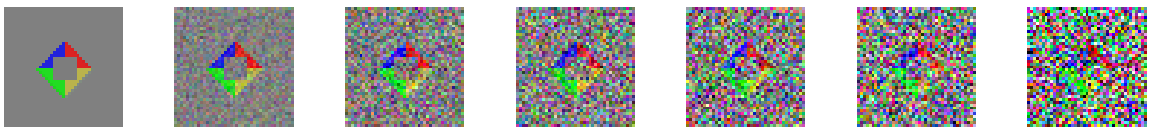


Figure 3.3: Synthetic objects under various levels of noise.
The number under each image denotes the standard variance of the Gaussian noise.

The first evaluation compares the proposed method with the mean-shift tracker [Comanicui et al., 2000] in tracking a translational object. In order to obtain accurate evaluation, we ran 8 independent trials at each noise level and averaged the results. Figure 3.4 plots the tracking errors as functions over noise level. The proposed method could accurately track the object under noise level $\sigma \leq 70$, while the mean-shift tracker worked well only if noise level is much lower ($\sigma \leq 30$).

The second evaluation examines the proposed method in tracking the object under a particular type of transformation (rotation, shearing or scaling) in combination with translation. We also ran 8 independent trials at each noise level. Figure 3.5 plots the experimental results. The method performed well under noise level $\sigma \leq 60$ (for shearing or rotating) or $\sigma \leq 40$ (for scaling). The results suggest that the method is insensitive to noise in scaling or rotating objects. In the presence of noise in scaling objects, the method appears to be less robust but the performance is still favorable in view of the heavy noise it can handle well at $\sigma = 40$ (Note that even without object scaling in the earlier test, the mean-shift tracker could not deal with such noise).

The last evaluation examines the proposed method in tracking the object under

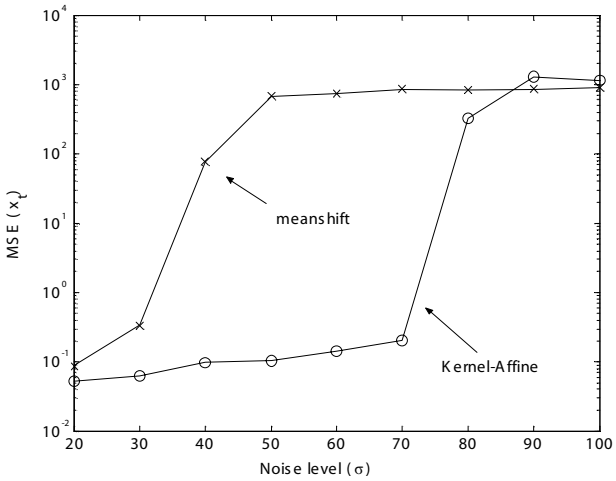


Figure 3.4: Comparative results of tracking synthetic object, with the proposed method or mean-shift.

fully affine transformation. Eight independent trials at each noise level were conducted. Figure 3.6 plots the results which show that the method can accurately determine the object state despite heavy noise corruption in the images.

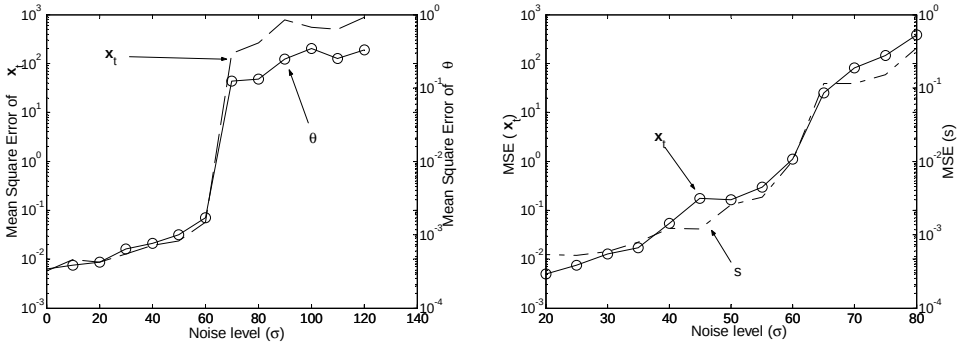
3.8 Tracking Real-world Objects

The purpose of this section is to examine the proposed method in tracking real-world objects, in comparison with the state-of-the-art mean-shift tracker. A variety of target objects are considered, including hands, faces, cars, tanks and a special circular object.

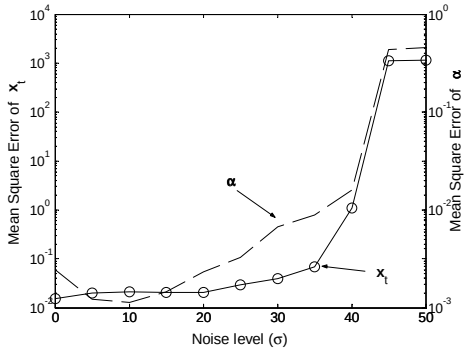
Hand and Face Tracking

The hand video (Figure 3.7 and 3.8) was captured in a lab environment using a video camera. The test sequence is at 4fps and 360×288 pixels. In the videos, the two hands exhibit the same color features which are also similar to the background. The experimental results show that the mean-shift tracker failed to track the object, while the proposed method accurately computed the hand state throughout the sequence. The results can be attributed to the fact that precise spatial-spectral representation models in the proposed method are critical for distinguishing the objects with similar color features.

The face video is at 4fps and 160×120 pixels. The face is continuously moving and rotating, resulting in large variations in size and pose angle. Figure 3.9 draws some



(a) Translation and Rotation **(b)** Translation and Shearing



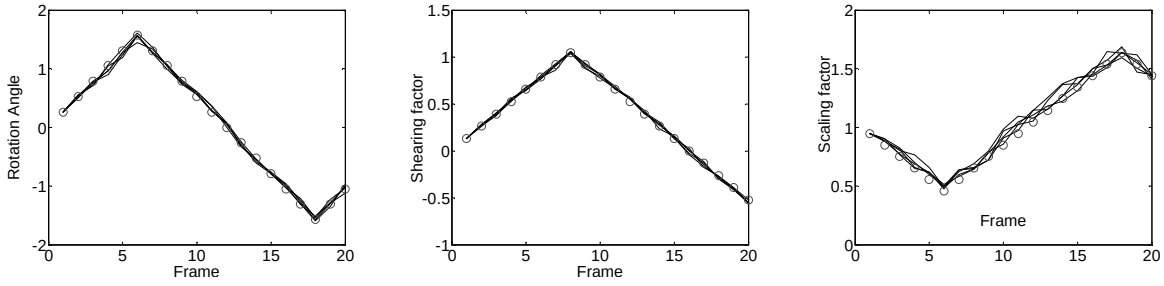
(c) Translation and Scaling

Figure 3.5: Tracking synthetic objects over various levels of noise

tracking results. It can be seen that the tracker was able to accurately determine the position, orientation and size of the face, despite the fact that the face was subject to slightly out-of-plane rotation.

Circle Tracking

The target object in this experiment is a red-white circular object that moves and rotates on a red-white chessboard (Figure 3.10 and 3.11). The tracking task appears to be very difficult, since the circle looks quite similar to the background. Nevertheless, the results (Figure 3.10) show that the proposed method can accurately determine the object state throughout the sequence. It is also evident from Figure 3.11 and 3.12 that neither the mean-shift tracker nor Condensation [Isard and Blake, 1996] (a contour tracker with particle filtering) could perform well in this setting. These suggest that precise spatial-spectral representation is critical for distinguishing objects with similar color features.



(a) Rotation **(b)** Shearing **(b)** Scaling

Figure 3.6: Tracking synthetic objects with affine transformation under image noise at $\sigma = 40$.

Circles denote the true state, while the curves represent tracking results in different trials.

Vehicle Tracking

The vehicle video (Figure 3.13) was acquired from KOGS/IAKS Universität Karlsruhe. The target objects are cars turning at a road crossing. The image quality is poor due to high-ratio video compression and snowy weather, plus the small size of the object image in the video. Comparing Frame 0 and Frame 36, it can be seen that the object image undergoes a considerable deformation especially in terms of rotation and shearing. The tracking results show that the tracker could still accurately capture the car's movement and recover the trajectory.

We captured another car video (Figure 3.14) at 360×288 pixels and 15fps. Note that only the image regions around the target are extracted for display, as the target is rather small in the images. The movement of the car is out-of-plane, posing a challenge to single-example based systems to recover affine transformation. In spite of that, the proposed tracker favorably determined the car's state in terms of orientation, dimension and position.

Tank Tracking

The tank videos (Figure 3.15 and 3.16) are at 15fps and 356×288 pixels. The tracking is made very challenging by the close similarity between the appearances of the camouflaged tanks and the meadow background. Note that due to considerable out-of-plane rotation, some parts of the tanks are absent in the first frame but are present later,

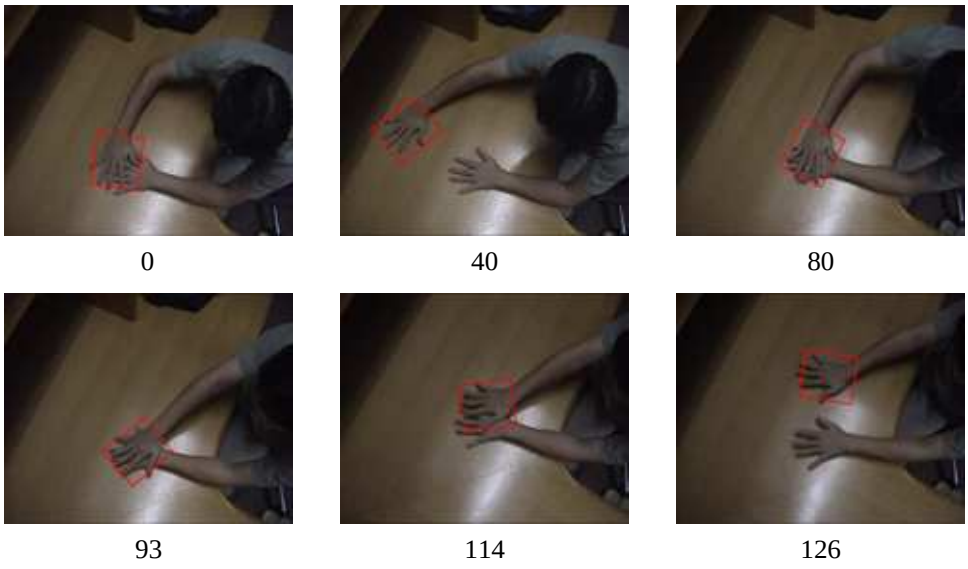


Figure 3.7: Hand tracking with the proposed method.

Figure 3.8: Hand tracking with the mean-shift tracker.

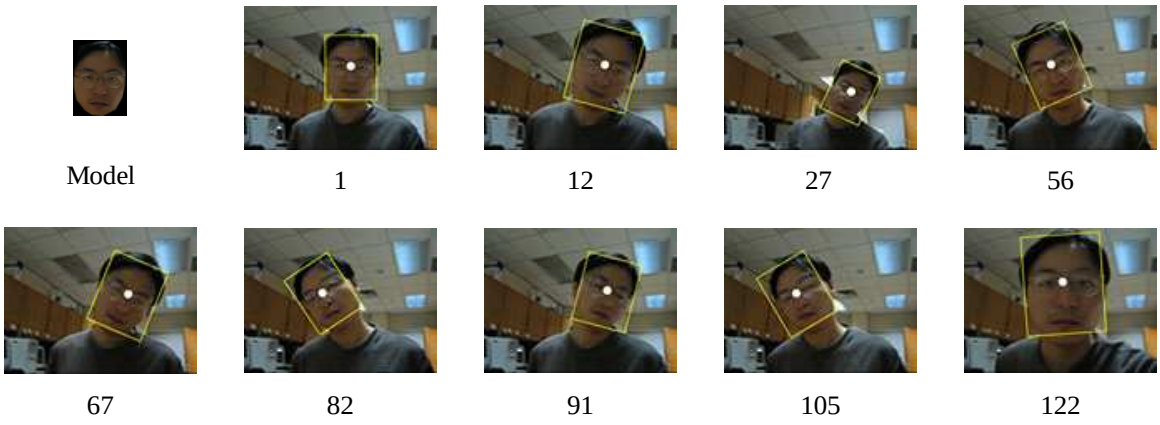


Figure 3.9: Face Tracking

and they are not taken into consideration in the representation models. Nevertheless, the proposed method was still able to accurately recover the affine transformation of tanks. On the other hand, the same tracking task will largely perturb other systems using smooth blobs or contour models, due to the similar, complex patterns in both the target and the background.

3.9 Summary

The experimental results show that the proposed tracking method is capable of tracking affine transformation of a variety of synthetic and real world objects. In particular, the proposed method appeared to be very robust against image noise and significantly outperformed the state-of-the-art mean-shift tracker in the simulations. Besides, the method could accurately recover the state of moving hands and a moving circle in cluttered background, while the mean-shift tracker could not identify them. In addition, a few experiments on vehicles and camouflaged tanks also demonstrate that the method is able to deal with affine transformation in various scenarios.

The good performance can be attributed to three important components of the method. The first important component is the kernel-based representation model, which can accurately characterize an object and effectively identify the object in cluttered background. For instance, the camouflaged tank has been well identified though it appears very similar to the background. In addition, the smoothness of Gaussian kernels in both spectral and spatial spaces allows minor variations in the object image. Thus,

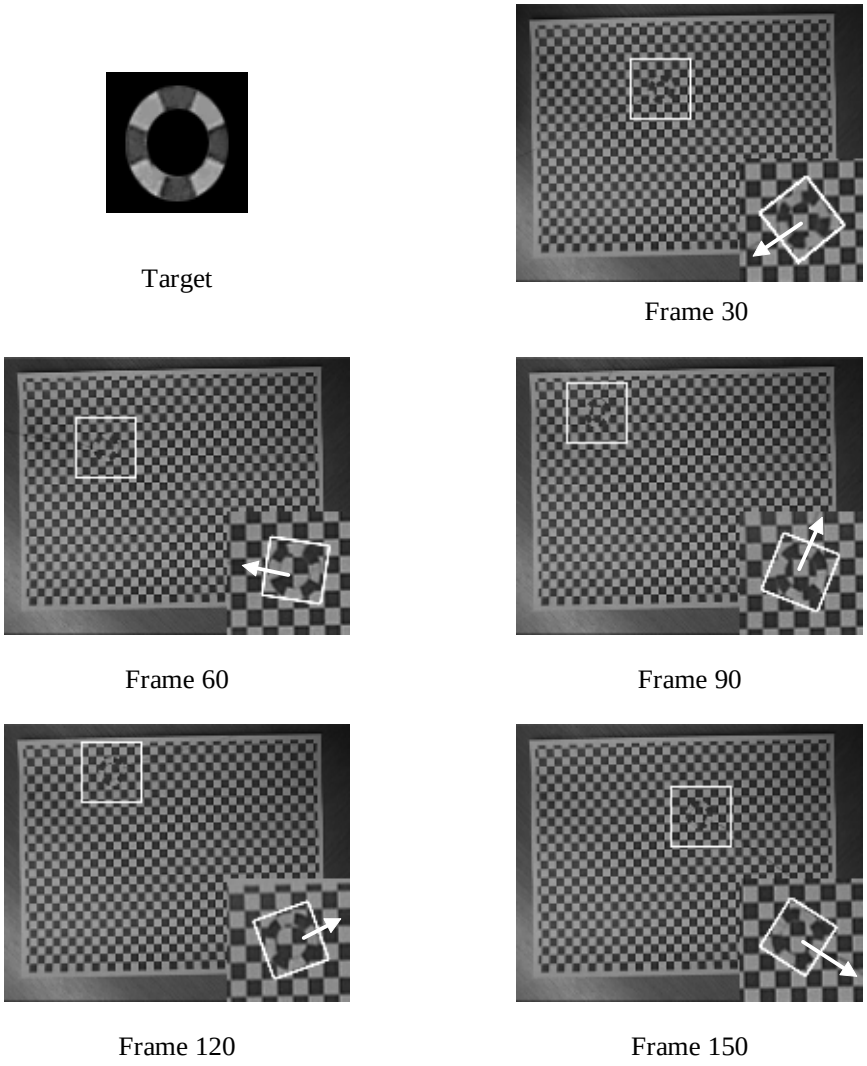


Figure 3.10: Tracking circle with proposed method.

The figure shows the target model as well as the tracking results by the proposed affine tracker, where the tracking results are denoted by the outlines and the directions. The lower right sub-images detail the white-framed regions in the original images that encompass the object.

Figure 3.11: Tracking circle with the mean-shift tracker

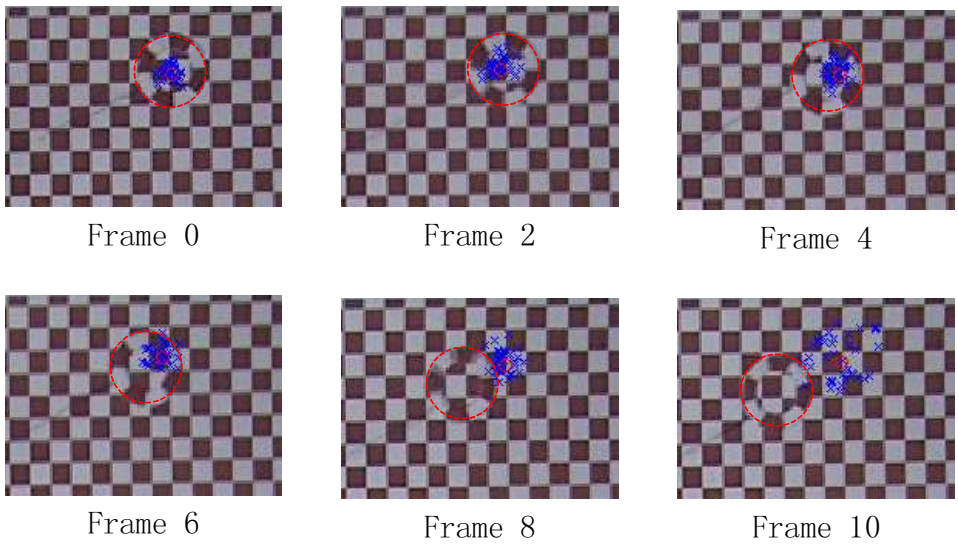


Figure 3.12: Tracking circle with the Condensation. Here we show only cropped images to bring out the details of random samples used for Condensation. True objects are outlined by red circles.

despite the presence of motion blurs, heavy noise, or visible artifacts in the images, the model is still robust and accurate for representation.

The second important component of the method is the incorporation of affine transformation in the tracking formulation. It allows us to describe and capture transformation of near-planar objects through image sequences, in terms of shearing, scaling, rotation and translation. The recovered image deformation not only provides rich information about the object movement, but also enables us to continuously take advantages of precise spatial-spectral features that are essential for identifying the objects.

The third component which attributes greatly to the good performance is the optimization procedure. The procedure is effective for seeking the transformation state. When implemented in a special searching-based tracking algorithm, it enables us to iteratively improve the estimation of the state while continuously improving the acquisition of the object image.

At present, setting up a representation model is a trivial task upon the acquisition of a sample image of a target object. Hence, the tracking system is easy to implement. There are a few possible ways to enhance the representation model. As suggested by Collins [Collins and Liu, 2003], for example, the system may select, for target object representation, distinctive features from the background. We need to mention that this

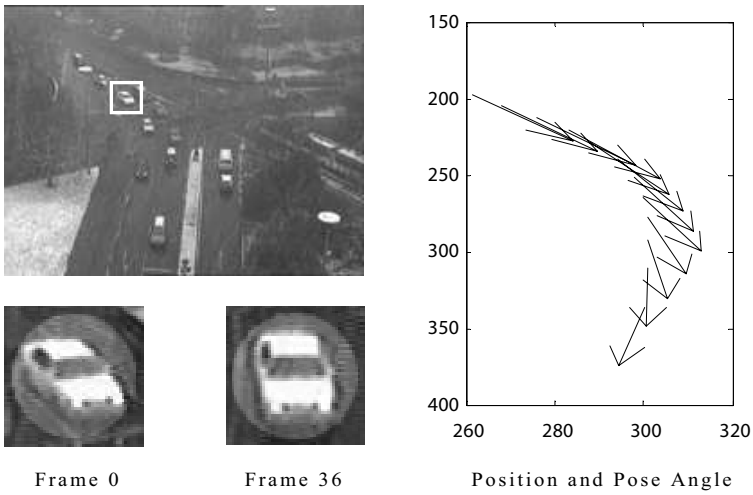


Figure 3.13: Vehicle Tracking Experiment 1

The upper left panel shows the first frame of the traffic video, with white frame denoting the car to be tracked. The right panel plots the tracking results as a sequence of arrow. The start point of an arrow represents the estimated position, while the arrow points to the estimated pose angle.

scheme has a side effect in that it may loss some essential information about the shape deformation. Therefore, it will be promising but challenging to incorporate feature selection techniques into our kernel-based affine tracking method.

The presented tracking method emphasizes not on tracking complex 3D objects but on tracking near-planar objects' rigid deformation in terms of affine transformation. Therefore, it may face problems in dealing with complex objects that show different views in an image sequence. One may extend the presented tracking scheme to dealing with full 3D objects, by setting up an 3D representation model. But obviously, that will be a demanding and challenging job, since complete 3D modeling requires a great deal of prior knowledge about the particular objects plus the problem is yet to be well-solved in general.

This work also pays scant attention to efficient implementation. As mentioned earlier, tracking speed on a 1000-pixel target is about 2 fps which may be insufficient for many real applications. A possible way to dramatically improve the computational efficiency is to reduce the number of kernels [Girolami and He, 2003] in the kernel-based representation model, while the topic is beyond the scope of this thesis.

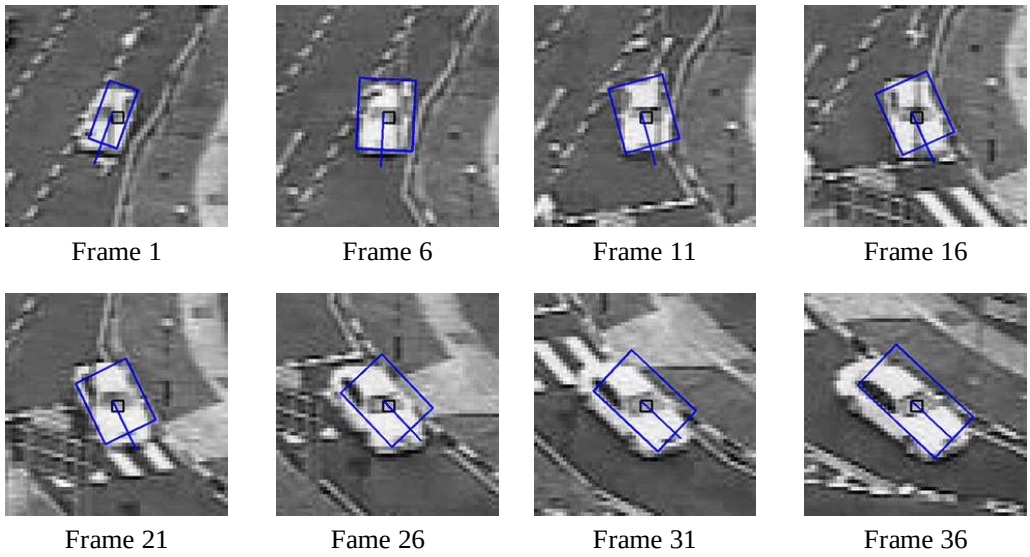


Figure 3.14: Vehicle Tracking Experiment 2

The black boxes denote the estimated image region of the car. The small squares represent the positions of the moving vehicle, while the segments pointing out from the positions show the changing direction of the car.

3.10 Discussions

3.10.1 A brief discussion on other affine-invariant tracking methods

Affine-invariant tracking has long been interesting to the computer vision community. And the thesis has mentioned a number of existing methods for it in the introduction section of the previous chapter. We have also seen that in many cases color features alone could not provide sufficient characteristic information and spatial-color feature based approaches would be more appropriate.

A prominent problem with the feature-based approaches is that it is generally very difficult to handle complex variations of the features caused by transformations. More specifically, the problem mostly lies in the design of a searching function for properly and efficiently updating the parameters based on observed image features.

In a conventional way, many trackers (e.g. [Ferrari et al., 2001]) do not use analytical searching (or iterative) functions, instead they scan possible values for the object's state and pick the most-likely one. But the approach tends to be less effective and elegant than analytical searching methods such as the active appearance models (AAMs) method

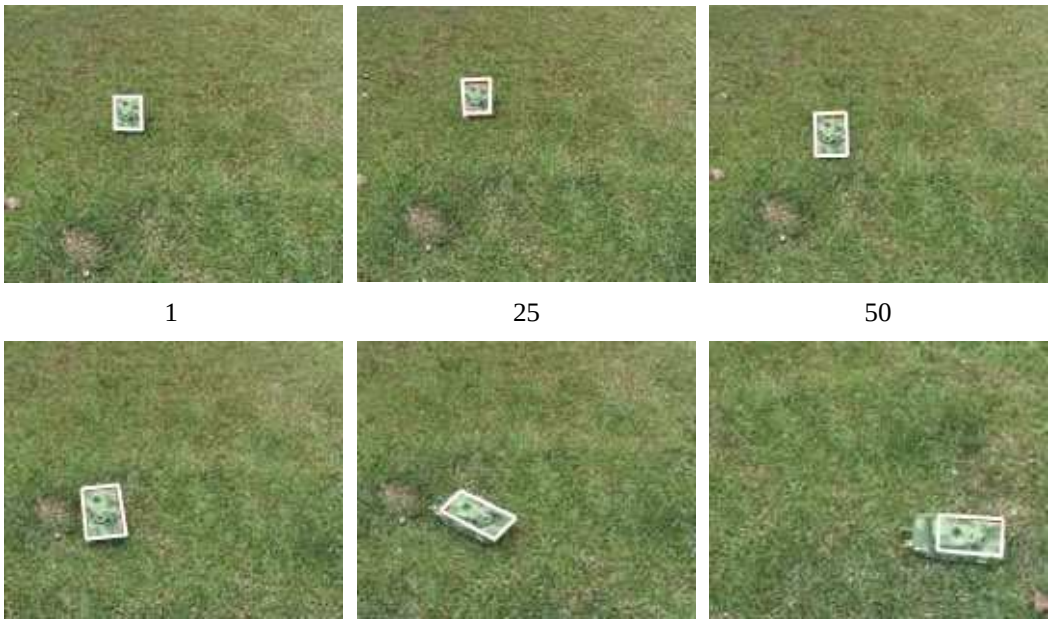


Figure 3.15: Tank tracking 1

The white frame denotes the estimated outlines of the tank model.

[Cootes et al., 2001]. By learning a number of image samples for each deformable object, the AAMs can well represent dynamic and complex image features. Importantly for tracking, the authors presented a linear, iterative function to adapt the models to the images. But the convergence of the linear iterations has yet to be proven in a strict sense.

More recently, J. Winn and A. Blake [Winn and Blake, 2004] introduced a Bayesian network for affine tracking. Their inference algorithm includes a global search over a discretised transform space followed by a local optimisation with a trust-region Newton method. In addition, it uses bottom-up cues to restrict the space of possible affine transformations in order to aid correct convergence. Similar to our work, they also decomposed the transformation matrix into some element matrices respectively for translation, rotation and scaling, and a freeform linear transformation. A small regret is that their searching objective function can not be optimised analytically.

Unlike the previous tracking methods, the presented tracker in the paper takes advantages of both the kernel-based spatial-color representation model and the analytical

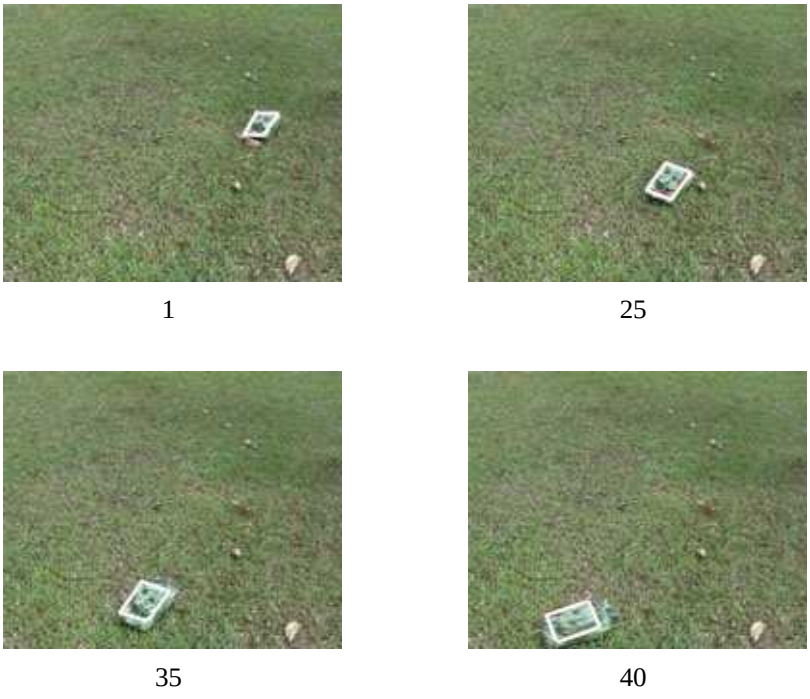


Figure 3.16: Tank tracking 2

The white frame denotes the estimated outlines of the tank model.

search algorithm. The algorithm is derived from a mathematical analysis and inherits the favorable properties of the traditional iterative programming for solving non-linear problems. Besides, thanks to the full differentiability of the objective function, it is possible to use more advanced, analytical optimization techniques in the tracking framework.

3.10.2 About a non-physically-parameterized transformation model

The affine transformation model presented involves a few geometrical operations including translation, rotation, scaling and shearing. In the proposed method we use a transformation that explicitly describes those factors. On the other hand, the transformation can also be written in such a form that those physical operators are transparent to the user. This affine transformation model, here called the physically-implicit model, may be applicable to some tracking tasks where the transformation parameters with physical meaning are not required.

In this section, we show that the physically-implicit model also lead to an iterative optimization procedure for similarity maximization and tracking as well. However, we

show that the optimization procedure in practical tracking settings would hardly converge to the true states of target objects. In other words, the physically-implicit model is not as applicable as the proposed physically-explicit model to real tracking tasks.

First, we can write the physically-implicit affine transformation as

$$\mathbf{x}^{(T)} = \begin{pmatrix} \rho_{11} & \rho_{12} \\ \rho_{21} & \rho_{12} \end{pmatrix} \mathbf{x} + \mathbf{x}_t = \begin{pmatrix} \rho_1^T \mathbf{x} \\ \rho_2^T \mathbf{x} \end{pmatrix} = M\mathbf{x} + \mathbf{x}_t \tag{3.33}$$

where the matrix M consists of four elements that implicitly reflect the geometrical operations except translation. A tracking system based on this new form of affine transformation may be appealing in some applications where the explicit transformation parameters are not desired. Now let's derive a tracking method from the new affine formulation and investigate how it works in practice.

Similar to the optimization procedures described in Section 3.4, a tracking method based on the affine formulation Eq.3.33 will also resort to optimization procedures for finding the transformation factors.

Given an observation Ω_p, for the purpose of tracking, one needs to find a transformation T that maximizes its similarity to the target model. We adapt the similarity measure in Eq 3.6 to the new affine formulation, yielding

$$\begin{aligned} D(T) &= -\frac{1}{N_p^2}\sum_{i,j} k_s\left(\frac{||M(\mathbf{x}_i^{(p)} - \mathbf{x}_j^{(p)})||^2}{2}\right) k_u\left(\frac{||\mathbf{u}_i^{(p)} - \mathbf{u}_j^{(p)}||^2}{2}\right) \\ &\quad + \frac{2}{N_p N_q}\sum_{i,j} k_s\left(\frac{||M\mathbf{x}_i^{(p)} + \mathbf{x}_t - \mathbf{x}_j^{(q)}||^2}{2}\right) k_u\left(\frac{||\mathbf{u}_i^{(p)} - \mathbf{u}_j^{(q)}||^2}{2}\right) \\ &\triangleq -D_1(p_{_T}, p_{_T}) + D_2(p_{_T}, q) \end{aligned} \tag{3.34}$$

Thus, maximizing the similarity amounts to finding a transformation state T that satisfies $\nabla_{_T} D(p_{_T}, q) = 0$.

Consider the term D_1. By defining

$$f_{ij}^{(1)} = ||M\mathbf{x}_i^{(p)} - M\mathbf{x}_j^{(p)}||^2 \tag{3.35}$$

we have

$$\begin{aligned} f_{ij}^{(1)} &= (\rho_1^T \mathbf{x}_i^{(p)} - \rho_1^T \mathbf{x}_j^{(p)})^2 + (\rho_2^T \mathbf{x}_i^{(p)} - \rho_2^T \mathbf{x}_j^{(p)})^2 \\ &= \rho_1^T (\mathbf{x}_i^{(p)} - \mathbf{x}_j^{(p)})(\mathbf{x}_i^{(p)} - \mathbf{x}_j^{(p)})^T \rho_1 + \rho_2^T (\mathbf{x}_i^{(p)} - \mathbf{x}_j^{(p)})(\mathbf{x}_i^{(p)} - \mathbf{x}_j^{(p)})^T \rho_2 \end{aligned} \tag{3.36}$$

Therefore, the derivatives are

$$\frac{\partial f_{ij}^{(1)}}{\partial \rho_1} = 2(\mathbf{x}_i^{(p)} - \mathbf{x}_j^{(p)})(\mathbf{x}_i^{(p)} - \mathbf{x}_j^{(p)})^T \rho_1 \tag{3.37}$$

$$\frac{\partial f_{ij}^{(1)}}{\partial \rho_2} = 2(\mathbf{x}_i^{(p)} - \mathbf{x}_j^{(p)})(\mathbf{x}_i^{(p)} - \mathbf{x}_j^{(p)})^T \rho_2 \tag{3.38}$$

And the derivatives of D_1 with respect to T can be computed by

$$\begin{aligned} \nabla_{\rho_1} D_1(p_T, p_T) &= \nabla_{\rho_1} \left(\frac{1}{N_p^2} \sum_{i,j} k_s(\frac{f_{ij}^{(1)}}{2}) k_u(\frac{||\mathbf{u}_i^{(p)} - \mathbf{u}_j^{(p)}||^2}{2}) \right) \\ &= \sum_{i,j} \frac{1}{2} v_{ij} \frac{\partial f_{ij}^{(1)}}{\partial \rho_1} = \sum_{i,j} v_{ij} (\mathbf{x}_i^{(p)} - \mathbf{x}_j^{(p)})(\mathbf{x}_i^{(p)} - \mathbf{x}_j^{(p)})^T \rho_1 \\ &= A_1 \rho_1 \end{aligned} \tag{3.39}$$

$$\begin{aligned} \nabla_{\rho_2} D_1(p_T, p_T) &= \nabla_{\rho_2} \left(\frac{1}{N_p^2} \sum_{i,j} k_s(\frac{f_{ij}^{(1)}}{2}) k_u(\frac{||\mathbf{u}_i^{(p)} - \mathbf{u}_j^{(p)}||^2}{2}) \right) \\ &= \sum_{i,j} \frac{1}{2} v_{ij} \frac{\partial f_{ij}^{(1)}}{\partial \rho_2} = \sum_{i,j} v_{ij} (\mathbf{x}_i^{(p)} - \mathbf{x}_j^{(p)})(\mathbf{x}_i^{(p)} - \mathbf{x}_j^{(p)})^T \rho_2 \\ &= A_2 \rho_2 \end{aligned} \tag{3.40}$$

Now consider the term D_2. By defining

$$f_{ij}^{(2)} = ||M\mathbf{x}_i - \mathbf{x}_j||^2 \tag{3.41}$$

we have

$$\begin{aligned} f_{ij}^{(2)} &= (\rho_1^T \mathbf{x}_i^{(p)} - x_j^{(q)})^2 + (\rho_2^T \mathbf{x}_i^{(p)} - y_j^{(q)})^2 \\ &= \rho_1^T \mathbf{x}_i^{(p)} \mathbf{x}_i^{(p)T} \rho_1 - 2\rho_1^T \mathbf{x}_i^{(p)} x_j^{(q)} + x_j^{(q)2} + \\ &\quad \rho_2^T \mathbf{x}_i^{(p)} \mathbf{x}_i^{(p)T} \rho_2 - 2\rho_2^T \mathbf{x}_i^{(p)} y_j^{(q)} + y_j^{(q)2} \end{aligned} \tag{3.42}$$

Hence, the derivatives are

$$\frac{\partial f_{ij}^{(2)}}{\partial \rho_1} = 2\mathbf{x}_i^{(p)} \mathbf{x}_i^{(p)T} \rho_1 - 2x_j^{(q)} \mathbf{x}_i^{(p)} \tag{3.43}$$

$$\frac{\partial f_{ij}^{(2)}}{\partial \rho_2} = 2\mathbf{x}_i^{(p)} \mathbf{x}_i^{(p)T} \rho_2 - 2y_j^{(q)} \mathbf{x}_i^{(p)} \tag{3.44}$$

So, the derivatives of D_1 with respect to T can be computed by

$$\begin{aligned}
\nabla_{\rho_1} D_2(p_T, q) &= \nabla_{\rho_1} \left(\frac{2}{N_p N_q} \sum_{i,j} k_s(\frac{f_{ij}^{(2)}}{2}) k_u(\frac{||\mathbf{u}_i^{(p)} - \mathbf{u}_j^{(p)}||^2}{2}) \right) \\
&= \sum_{ij} \frac{1}{2} w_{ij} \frac{\partial f_{ij}^{(2)}}{\partial \rho_1} \\
&= \sum_{ij} w_{ij} (\mathbf{x}_i^{(p)} \mathbf{x}_i^{(p)T} \rho_1 - x_j^{(q)} \mathbf{x}_i^{(p)}) \\
&= B_1 \rho_1 + \mathbf{c}_1 \qquad (3.45)
\end{aligned}$$

$$\begin{aligned}
\nabla_{\rho_2} D_2(p_T, q) &= \nabla_{\rho_2} \left(\frac{2}{N_p N_q} \sum_{i,j} k_s(\frac{f_{ij}^{(2)}}{2}) k_u(\frac{||\mathbf{u}_i^{(p)} - \mathbf{u}_j^{(p)}||^2}{2}) \right) \\
&= \sum_{ij} \frac{1}{2} w_{ij} \frac{\partial f_{ij}^{(2)}}{\partial \rho_2} \\
&= \sum_{ij} w_{ij} (\mathbf{x}_i^{(p)} \mathbf{x}_i^{(p)T} \rho_2 - 2 y_j^{(q)} \mathbf{x}_i^{(p)}) \\
&= B_2 \rho_1 + \mathbf{c}_2 \qquad (3.46)
\end{aligned}$$

Therefore, letting $\nabla D(p_T, q) = 0$ amounts to having $\nabla - D_1(p_T, p_T) + \nabla D_2(p_T, q) = 0$, which yields

$$(A_1 - B_1)\rho_1 + \mathbf{c}_1 = 0 \qquad (3.47)$$

$$(A_2 - B_2)\rho_2 + \mathbf{c}_2 = 0 \qquad (3.48)$$

So the maximization problem can be solved with linear algebra:

$$\rho_1 = (A_1 - B_1)^{-1} \mathbf{c}_1 \qquad (3.49)$$

$$\rho_2 = (A_2 - B_2)^{-1} \mathbf{c}_2 \qquad (3.50)$$

The equations imply an iterative solution, since $A_{1,2}$ and $B_{1,2}$ are variable matrices indirectly determined by $\rho_{1,2}$.

For the translation estimation ($\mathbf{x}_t$), we can still use the iteration equation Eq. 3.11, because the term $\mathbf{x}_t$ in Eq. 3.33 play the same role as in the original affine formulation (Eq. 2.18).

To examine the above method, we use the same artificial objects introduced earlier. The object is scaled by 1.25, sheared by $\pi/6$, rotated by $\pi/12$. So the resulting transformation matrix is $M^* = (\rho_{11} = 1.21, \rho_{12} = 0.44, \rho_{21} = 0.32, \rho_{22} = 1.15)$. An

observation Ω_p is a coarse candidate defined by an estimation of state $T = \{M, \mathbf{x}_t\}$. And, to its advantage, we set the position estimation $\mathbf{x}_t$ precisely at the true position. The optimization procedure is used to find an M that maximize the similarity between Ω_q and Ω_p with respect to T. Figure 3.17 shows the tracking results. It is evident that the procedure could not converge to the true transformation state.

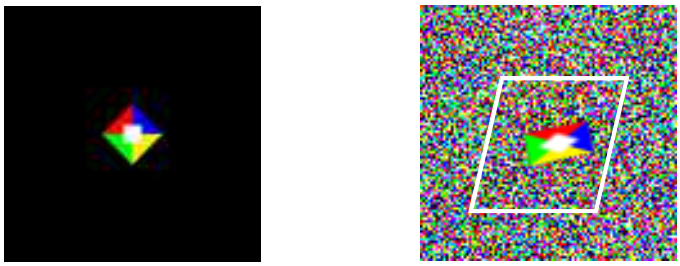


Figure 3.17: Affine tracking without explicitly accounting for transformation operations. *The left panel shows the object for modeling. The right panel shows the noise-corrupted image and the object subject to affine transformation. The white parallelogram denotes the estimated region of the object.*

To investigate the underlying problem, we compute the similarity surface and show it in Figure 3.18. From the graph, it can be seen that the similarity surface does not correctly indicate the state of transformation, since the maximum on the surface significantly strays away from the true state.

Here we would like to brief a possible reason. Let's consider the properties of affine matching. They assert that estimating different transformation factors has different requirements on the quality of object image candidate. In particular, the affine matching with respect to scaling factors is sensitive to background interference in an acquired candidate. The proposed method (Section 3.5) avoid simultaneously estimating all the transformation parameters with a candidate that may possibly include a considerable number of background pixels. Instead, it uses an effective scheme that only performs scaling-factor estimation when it acquires appropriate candidate by recovering the information about other parameters. On the other hand, the physically-implicit model does not differentiate the estimation of scaling factors from the others, and therefore can hardly deal with the background interference problem.

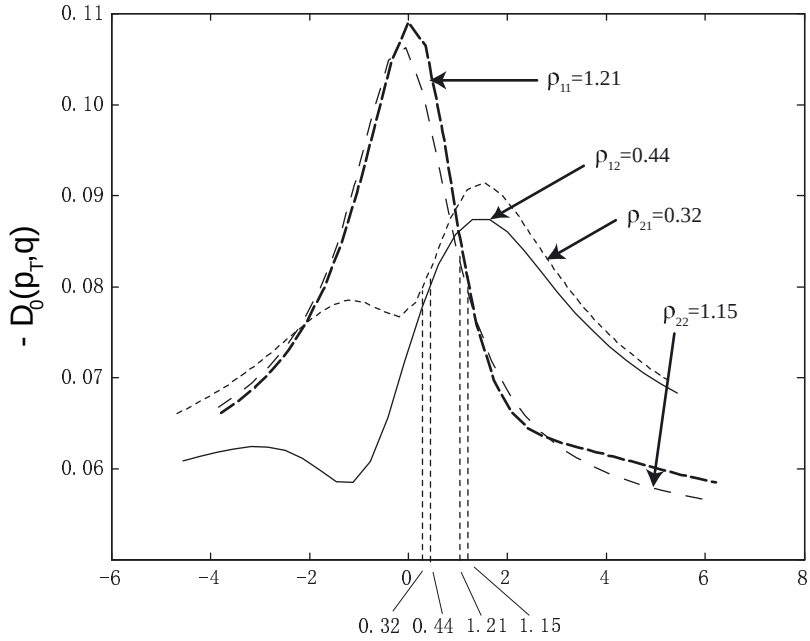


Figure 3.18: Similarity surface of affine matching

The horizontal axes represent the deformation factors (for all ρ). The similarity surface is drawn as four functions over the factors. True values of deformation factors are printed under the graph. Note that in the graph we plot $-D_0(p_T, q)$ instead of $D_0(p_T, q)$, therefore the minimal points on the curves correspond to the maximal points on the similarity surface.

Chapter 4

Kernel Autoassociators for Concept Learning and Recognition

A fundamental issue addressed by this thesis is visual recognition. The previous chapters have presented an approach to recovering the geometrical properties of target objects (in terms of affine transformation). One can use the recovered information to obtain normalized images/features of the object, and the normalization is a prerequisite to the success of many generic pattern recognition systems.

Suppose the image patterns of target objects are presented in normalized form. The objective of this chapter is to develop a generic classification scheme that discriminates among different classes of image patterns. To this end, we propose a neural network model based on kernel methods to learn the nonlinearity in the patterns. The chapter also studies its application to various binary-class and multi-class classification tasks. Its applications in combination with special domain knowledge will be described in the next chapter.

4.1 Background

Pattern classification is an important issue in a variety of scientific and engineering disciplines. Two elements play key roles in a classification process: concept and category [Medin and Coley, 1998]. *Concept* refers to an abstract representation of a category, while *category* refers to the set of entities picked out by the concept. The problem of concept learning is often made difficult by the shear complexity of patterns present in real

tasks. A well-known example is face recognition in which facial images can be very complicated and highly nonlinear especially when faces are subject to change in view angle or lighting condition [Zhao et al., 2000]. It is also known that linear approaches such as Fisher linear discriminant and principal component analysis [Turk and Pentland, 1991] are not well suited to learn the nonlinear concept.

This chapter emphasizes an alternative approach to nonlinear concept learning and pattern classification by using a special type of artificial neural networks called autoassociators. An autoassociator is a brain-like distributed network that learns from the samples in a category to reproduce each sample at the output by a mapping ([Haykin, 1999] p. 66)

$$\hat{\mathbf{x}}_i = F(\mathbf{x}_i) \tag{4.1}$$

where $\mathbf{x}_i$ is the ith sample (viewed as a vector herein) in a given sample set $\mathbf{x}_i$, $i \in [1, N]$ with size N.

The reproduction may seem pointless, whereas through learning the autoassociation the network may find the commonalities in the samples and thus grasp the underlying concept ([McLeod et al., 1998] p.72). For instance, Kohonen has demonstrated in an early work that an autoassociator can be used to store and retrieve face images [Kohonen, 1980]. Daunicht has also demonstrated that autoassociators are useful for modeling neuromechanics [Daunicht, 1991].

Autoassociators are generally used as one-class learning machines. In other words, each network corresponds to a particular category, and during training it receives only the samples within the category. An important consequence is that the network will learn to accurately reproduce *positive samples* (samples in the corresponding category), producing a reproduction error surface that reflects the distribution of the samples:

$$E_r(\vec{x}) = ||F(\mathbf{x}) - \mathbf{x}|| \tag{4.2}$$

where E_r denotes the error of reproduction through the autoassociation function F for a given pattern $\mathbf{x}$.

Thus, autoassociators provide an alternative approach to concept learning. In particular, the higher the reproduction quality for an input pattern, the more likely it belongs

to the category for which the autoassociator is constructed. That provides a basis for various autoassociator-based classifiers as below which depend on reproduction error surfaces to discriminate between classes.

Autoassociators are well suited to address a special binary classification issue called novelty detection in which novel or abnormal patterns are expensive or difficult to obtain, thus only a few or even no novel examples are available for learning. An extensive survey in the area of novelty detection can be found in [Markou and Singh, 2003a], and Markou and Singh have especially presented in [Markou and Singh, 2003b] an excellent review on autoassociator-based approaches. Autoassociator-based approaches rely on the fact that, since an autoassociator only learns to give high quality reproductions for normal patterns, the reproduction of a novel pattern will yield a large error which can be thresholded to signal novelty. This classification methodology has been applied to various detection problems such as face detection [Féraud et al., 2001], motor failure detection [Petsche et al., 1996], network security [Yeung and Chow, 2002], and natural language grammar learning [Hanson and Kegl, 1987].

Autoassociators are also useful for multi-class classification with a competitive scheme. The system creates a set of networks for each class, and then a probe pattern is reproduced by each network in testing phase. The respective reproduction error provides the basis for competition among the networks. The particular network with the smallest value of reproduction error is declared winner of the competition. This classification methodology has been successfully demonstrated in various applications, such as handwritten character recognition ([Schwenk and Milgram, 1995, Zhang, 2001]) and face recognition [Zhang et al., 2004a].

In the field of autoassociative networks, linear autoassociators such as correlation associative memories have been extensively studied. Kohonen has pointed out that using a linear autoassociator to store and recall patterns is equivalent to computing a principal component analysis of the cross-product matrix of the patterns and reconstructing them as a weighted sum of eigenvectors [Kohonen, 1980]. It's the same case with multi-layer linear autoassociative networks, according to Baldi and Hornik's theoretical study in [Baldi and Hornik, 1989]. As the consequence, linear autoassociators

have serious limitations in exploring high order dependency among data. Naturally, nonlinear autoassociators are favorable.

Existing nonlinear autoassociator models are generally based on a special type of back-propagation networks [Rumerlhart et al., 1986] called autoassociative multilayer perceptrons. Such a network includes nonlinear hidden units between the input and the output units. The input-to-hidden layer connections perform the encoding with the hidden units building for input patterns internal representations, while the hidden-to-output layer connections do the decoding. That's why this type of networks are often referred to in the literature as *autoencoders*.

Autoencoders with one hidden layer have demonstrated their capability for learning low-dimensional nonlinear features [Hanson and Gluck, 2000]. However, it has been claimed that in some image processing cases [Cottrell et al., 1987, Valentin et al., 1994, Hertz et al., 1991] they are comparable to linear PCA. An existing method to overcome this problem is by having multiple hidden layers[Kramer, 1991]. The consequent architecture called nonlinear principal component analysis (NLPCA) often consists of three hidden layers, and its capability has been demonstrated in various domains [DeMers and Cottrel, 1993, Usui et al., 1991].

According to Moghaddam's study on face recognition [Moghaddam, 1999], however, NLPCA can be considerably outperformed by other methods including independent component analysis (ICA) [Comon, 1994a] and even the linear method of principal component analysis. The author suggests that the NLPCA's poor performance can be attributed to the general difficulty of computing nonlinear manifolds and the complexity of cost functions riddled with local minima. Furthermore, Malthouse has also pointed out certain limitations of autoencoders [Malthouse, 1998]. For example, when the network's solution is used to extrapolate, or when there are training values close to ambiguity points on principal curves, the encoding results by the network will be incorrect.

The goal of this chapter is to propose an alternative, more effective approach to modeling nonlinear autoassociations. To this end, we emphasize a special type of nonlinear method called kernel methods which have been established as a context for solving a

variety of nonlinear problems [Scholkopf and Smola, 2002]. In brief, kernel methods operate a special class of functions called *reproducing kernels* [Saitoh, 1988] $k(\mathbf{x}_1, \mathbf{x}_2)$ in the input pattern space (R^N), which amounts to casting the data into a high-dimensional kernel feature space ($\mathcal{H}$) by a possibly implicit map Φ, and taking the dot product there:

$$k(\mathbf{x}_1, \mathbf{x}_2) =< \Phi(\mathbf{x}_1), \Phi(\mathbf{x}_2) > \tag{4.3}$$

By virtue of this property, many linear algorithms have been extended to the kernel feature space, with the outcomes being nonlinear in the input space. Well-known examples include support vector machines (SVMs) [Vapnik, 1995], kernel Fisher discriminant (KFD)[Baudat and Anouar, 2000] and kernel principal component analysis (KPCA) [Scholkopf et al., 1998].

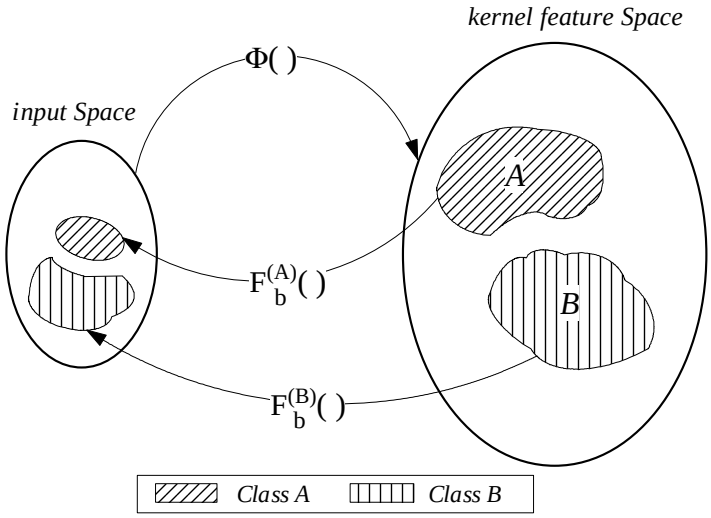


Figure 4.1: Illustration of kernel autoassocition.

Input patterns are projected through $\Phi(\cdot)$ to a kernel feature space $\mathcal{H}$. Then each kernel autoassociator learns to reconstruct a particular class of patterns from their kernel features. Here F_b (or F_{back} hereafter) denotes a reverse mapping function from the feature space to the original space.

The chapter provides a new perspective of kernel methods by using them to address the nonlinear autoassociation issue. In particular, we propose a kernel autoassociator model which associates the input and the output by mapping through the kernel feature space. Fig. 4.1 gives an illustration, where the autoassociation is accomplished through two phases: first, an input pattern $\mathbf{x}$ is cast into a kernel feature space by $\Phi(\mathbf{x})$: $\mathbf{x} \rightarrow \Phi(\mathbf{x})$, and it is then subsequently mapped backwards to the input space via F_{back}:

$\Phi(\mathbf{x}) \rightarrow \hat{\mathbf{x}}$, where F_{back} is class-dependent ($F_{back}^{(A)}$ for class A and $F_{back}^{(B)}$ for class B). The subscript b denotes that the function is for reverse mapping. Without otherwise specified, we will omit the class label m in describing reverse mapping functions in general.

Hence, the kernel autoassociation involves two important issues on the kernel mapping (Φ) and the reverse mapping (F_{back}) respectively. As mentioned earlier, autoencoders involve two comparable issues regarding the setup of nonlinear representations and the pattern reconstruction from the representations. Unlike autoencoders, kernel autoassociators put emphasis not on building nonlinear representations because choosing a reproducing kernel will automatically establish an associated kernel feature space. Instead, since the setup of reproducing kernel and kernel feature space has already been elegantly studied in the literature, we would like to pay particular attention to the reverse mapping.

Recall the two phases of kernel autoassociation as depicted in Figure 4.1. It can be seen that the first phase of kernel autoassociation through the kernel mapping $\Phi(\mathbf{x})$ is just a straightforward process without learning the particular concept from samples. Thus, the learning task rests on the modeling of the reverse mapping F_{back}, which should reflect the characteristics of the category.

It is worthwhile to mention that in a relevant study [Scholkopf et al., 1999], Schölkopf has proposed an algorithm for the reverse mapping F_{back}. However, it is not suited to model autoassociators due to two reasons: first, it does not involve the learning of class-specific reverse mapping; second, it is only applicable to Gaussian kernels.

This chapter addresses the above problem by proposing two reverse mapping methods. The first method uses linear functions in the kernel feature space, while the second one uses multivariate polynomial functions. The two methods are both applicable to arbitrary kernel types, and more importantly, they allow learning particular concepts of classes. Besides, a regularization method is proposed to improve the generalization performance of the kernel autoassociators with polynomial reverse mapping functions.

We apply kernel autoassociators to novelty detection and multi-class classification problems. Two detection schemes are developed for novelty detection with or without

novel examples, and they are tested on Promoter detection and Sonar Target recognition. The chapter proceeds to apply kernel autoassociators to multi-class classification problems in the domains of wine recognition, glass recognition, handwritten digit recognition and face recognition. The experimental results show that kernel autoassociators provide better or comparable performance for concept learning and recognition in various domains than conventional autoassociators and other existing recognition systems.

In summary, the chapter presents an alternative approach to nonlinear autoassociation. By making use of a kernel feature space, the approach resorts to relatively simpler functions (linear or polynomial) for learning, in contrast to conventional autoassociation machines that use a complex class of functions – such as a collection of sigmoid functions in multilayer perceptrons. The approach appears to be more accessible and easier to implement, and it is promising for a variety of novelty detection and multi-class classification applications.

The rest of the chapter is organized as follows. Section II introduces kernel autoassociator and elaborates two models for the reverse mapping. Section III presents a regularization method to improve the generalization performance of kernel autoassociators with polynomial reverse mapping functions. The proposed model is evaluated with simulations in Section IV, while Section V applied kernel autoassociators to novelty detection. Experiments on multi-class recognition are illustrated in Section VI, followed by the discussions and the conclusion in Section VII.

4.2 The Kernel Autoassociator Model

Kernel autoassociators produce pattern reproduction through Reproducing Kernel Hilbert Spaces (RKHS), which provide a unified context for solving a variety of statistical modeling and function estimation problems. Here, we review some basic concepts, the first one of which is about the positive definite function.

Let $\mathcal{T}$ be an index set, e.g Euclidean space E^N. A function $k(\mathbf{x}, \mathbf{t})$, $(\mathbf{x}, \mathbf{t}) \in \mathcal{T} \otimes \mathcal{T}$ is said to be a positive definite function on $\mathcal{T} \otimes \mathcal{T}$ if, for every number n, every set $\mathbf{t}_1, \ldots, \mathbf{t}_n$, and every $\mathbf{a} = [a_1, \ldots, a_n]^T \in \mathrm{R}^N$ we have

$$\sum_{i,j=1}^{n} a_i a_j k(\mathbf{t}_i, \mathbf{t}_j) \geq 0 \tag{4.4}$$

Let $k(\cdot,\cdot)$ be a positive definite function on $\mathcal{T} \otimes \mathcal{T}$, and $k_{\mathbf{t}}(\cdot) = k(\mathbf{t},\cdot)$ a functional with respect to $\mathbf{t}$. When $\mathbf{t}$ is fixed, $k_{\mathbf{t}}$ will be a determined function. According to the Moore-Aronszajn Theorem [Aronszajn, 1950], there exists, corresponding to $k(\cdot,\cdot)$, a unique collection of real valued functions on T, called a RKHS $\mathcal{H}_k$:

$$k_{\mathbf{t}} \in \mathcal{H}_k \quad \text{for each} \quad \mathbf{t} \in \mathcal{T} \tag{4.5}$$

$$\sum_{l=1}^{L} a_l k_{\mathbf{t}_l} \in \mathcal{H}_k \quad \text{for any finite } L \text{ and } \{a_l\} \tag{4.6}$$

The inner product in $\mathcal{H}$ is defined by

$$< k_{\mathbf{x}}, k_{\mathbf{t}} >= k(\mathbf{x},\mathbf{t}) \tag{4.7}$$

Let an arbitrary function in $\mathcal{H}_k$ be expressed in form of Eq. 4.6. Then we have

$$< k_{\mathbf{t}}, f >=< k_{\mathbf{t}}, \sum_{l=1}^{L} a_l k_{\mathbf{t}_l} >= \sum_{l=1}^{L} a_l < k_{\mathbf{t}}, k_{\mathbf{t}_l} >= \sum_{l=1}^{L} a_l k(\mathbf{t}_l,\mathbf{t}) = f(\mathbf{t}) \tag{4.8}$$

That is why $k(\cdot,\cdot)$ is called the reproducing kernel for $\mathcal{H}_k$.

Hence, by choosing a reproducing kernel function k one can cast a pattern $\mathbf{x}$ into a RKHS $\mathcal{H}_k$: $k(\mathbf{x},\cdot)$, and $\mathcal{H}_k$ is called a kernel feature space with respect to k.

The principle of kernel autoassociators, as mentioned earlier, is to perform autoassociation mapping via the kernel feature space, i.e. reconstructing patterns from their counterparts in $\mathcal{H}_k$. Hereafter a *reconstruction* (the reverse mapping) function is denoted by

$$\hat{\mathbf{x}} = F_{back}^{(m)}(\Phi(\mathbf{x})) \ , \ \text{for } \mathbf{x} \ \in \ \text{class } m \tag{4.9}$$

where $\Phi(\mathbf{x}) = k(\mathbf{x},\cdot)$ represents the feature in functional form in $\mathcal{H}_k$. As we have mentioned in Introduction, the subscript b denotes that the function is for reverse mapping. Note that without otherwise specified, we omit the class label m in describing a reverse mapping function hereafter.

A positive definite function $k(.,.)$ is associated with a unique RKHS, and thus can be used as a kernel function for kernel methods [Scholkopf and Smola, 2002]. In fact, the kernel autoassociator model does not confine the selection of kernel function k, while in the present chapter we tentatively examine two of the most popular kernel functions,

namely, Gaussian function and the polynomial function.

$$\text{Gaussian kernel:} \qquad k(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\frac{||\mathbf{x}_1 - \mathbf{x}_2||^2}{2\sigma^2}) \tag{4.10}$$

$$\text{Polynomial kernel:} \qquad k(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1^T \mathbf{x}_2 + 1)^p \tag{4.11}$$

where p denotes the power of the polynomial, σ the bandwidth of the Gaussian kernel.

A kernel autoassociator learns the concept of a category in a very high dimensional feature space $\mathcal{H}_k$, in contrast to conventional methods that learn in the input space or a low dimensional feature space. This methodology may raise doubts about whether it is good to resort to a higher dimensional feature space for learning, since the *curse of dimensionality* principle asserts that the difficulty of learning may increase drastically with the dimensionality.

The doubts can be resolved by the statistical learning theory which states: not the dimensionality but the complexity of the function class matters [Vapnik, 1998], and learning can be simpler if one uses a class of functions of low complexity. Another underlying justification is in Cover's theorem [Cover, 1965] on the separability of patterns. It states that a complex classification problem cast in a high dimensional space nonlinearly is more likely to be linearly separable than in a low dimensional space, provided two conditions are satisfied. First, the transformation is nonlinear. Second, the dimensionality of the feature space is high enough. A good example is a support vector machine that solves classification problems with linear decision rules in a high dimensional feature space instead of using nonlinear, complex decision rules in the input space.

It can be seen that our kernel feature mapping $\mathbf{x} \rightarrow \Phi(\mathbf{x})$ satisfies the above two conditions. As already shown in Figure 4.1, the kernel autoassociator model casts input patterns into a kernel feature space, and learns the class-specific dependencies between the feature space and the input space. The aforementioned theory suggests that we may use a simple class of functions such as linear functions or polynomials in the kernel feature space for concept learning.

4.2.1 Linear Functions for F_{back}

Let F_{back} be a linear mapping function from $\mathcal{H}_k$ to the input space E^N. The complete autoassociator is still nonlinear even though F_{back} is linear because of the intrinsic

nonlinearity of the feature mapping $\Phi(\cdot)$. When the patterns to be reproduced are multidimensional, F_{back} will be composed of a set of functions $\{f_{b_n}\}$, each corresponding to an element of the output space: $F_{back} = [f_{b_1}, \ldots f_{b_N}]^T$. Consider an element function f_{b_n}. We will omit the element-label n hereafter without otherwise specified. The function in linear form is given by

$$\hat{x} = f_{back}(\Phi(\mathbf{x})) =< \beta_\phi, \Phi(\mathbf{x}) > \tag{4.12}$$

Here $\hat{x}$ denotes an element of the output vector $\hat{\mathbf{x}}$, and β_ϕ is a vector in the feature space. Suppose the vector β_ϕ can be spanned by the images of M training samples [Scholkopf et al., 1999]

$$\beta_\phi = \sum_{i=1}^{M} b_i \Phi(\mathbf{x}_i) \tag{4.13}$$

we can rewrite the linear function f_{back} as

$$\hat{x} =< \sum_{i=1}^{M} b_i \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) >= \sum_{i=1}^{M} b_i k(\mathbf{x}_i, \mathbf{x}) = \mathbf{b}^T \mathbf{k} \tag{4.14}$$

where $\mathbf{b} = [b_1, \ldots, b_M]^T$ is the vector of expansion coefficients, and $\mathbf{k}$ represents the vector of kernel products $\mathbf{k} = [k(\mathbf{x}_1, \mathbf{x}), \ldots, k(\mathbf{x}_M, \mathbf{x})]^T$. Then the complete output vector $\hat{\mathbf{x}}$ is given by

$$\hat{\mathbf{x}} = B\mathbf{k} \tag{4.15}$$

where $B = [\mathbf{b}_1, \ldots, \mathbf{b}_N]$ denotes the collection of linear projections for each output element. Interestingly, it is the same as the expression of a kernel associative memory (KAM) [Zhang et al., 2004a], which however is derived in a different way as an extension of correlation associative memories. This finding suggests that KAMs can be considered as a special form of kernel autoassociators.

Given a set of samples, say, $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M)$ for training, one can first compute the kernel product vectors $\{\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_M\}$. The desired output of the network can then be expressed by

$$(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M) = B(\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_M) \text{ or } X = BK \tag{4.16}$$

Here X is the matrix with each column an example pattern, and K represents the matrix with each column a corresponding kernel product vector.

One way to learn the projection matrix B is by finding a matrix that minimizes the empirical square error $\sum_i ||\mathbf{x}_i - B\mathbf{k}_i||^2$. A method to the minimization is given by

$$B = XK^+ \tag{4.17}$$

where K^+ is the pseudo-inverse of the kernel product matrix K: $K^+ = (K^T K)^{-1} K^T$.

4.2.2 Polynomials for F_{back}

Let F_{back} consist of more complex functions, namely, 2nd order multivariate polynomials

$$\hat{x} = f_{back}(\Phi(\mathbf{x})) = \Phi^T(\mathbf{x}) W_\phi \Phi(\mathbf{x}) + \beta_\phi^T \Phi(\mathbf{x}) + c_\phi \tag{4.18}$$

where W_ϕ, β_π and c_ϕ are the polynomial coefficients. This formulation takes the feature $\Phi(\mathbf{x})$ as a column vector (i.e. $\beta_\phi^T \Phi(\mathbf{x}) =< \beta_\phi, \Phi(\mathbf{x}) >$), and allows exploring up to 2nd order nonlinearity in the reverse mapping f_{back}.

The direct calculation of Eq. (4.18) is not feasible because generally the kernel feature vector Φ is given in an implicit or extremely high dimensional form. Thus, we need to resort to approximation techniques to solve the problem.

Suppose that $\Phi(\mathbf{x})$ can be approximated by a low dimensional representation:

$$\Phi(\mathbf{x}) = \sum_{i=1}^{N_a} \alpha_i \mathbf{v}_i = (\mathbf{v}_1, \cdots, \mathbf{v}_{N_a})(\alpha_1, \cdots, \alpha_{N_a})^T = V\alpha \tag{4.19}$$

Here V is a matrix with each column a basis vector $\{\mathbf{v}_i\}$ of the N_a-dimensional subspace, and α denotes the projections of Φ onto V. We then have a new expression of the second order polynomial Eq. (4.18), given by

$$\hat{x} = \alpha^T W \alpha + \beta^T \alpha + c \tag{4.20}$$

where $c = c_\phi$, $W = V^T W_\phi V$ and $\beta = V\beta_\phi$. Clearly, it turns out to be a polynomial function with respect to the coefficient vector α.

Polynomials on Kernel Principal Components

Kernel Principal Component Analysis

A kernel subspace for the above representation can be set up by the KPCA technique [Scholkopf et al., 1998], which essentially performs a linear PCA in the kernel feature

space.

$$\Phi(\mathbf{x}) = \bar{\Phi} + (\Phi(\mathbf{x}) - \bar{\Phi}) = \bar{\Phi} + \sum_{i}^{N_a} \alpha_i \mathbf{v}_i \tag{4.21}$$

where $\bar{\Phi}$ is the average of all $\Phi(\mathbf{x}_i)$: $\bar{\Phi} = \frac{1}{M}\sum_{i=1}^{M}\Phi(\mathbf{x}_i)$ over M training samples. The set $\{\mathbf{v}_i\}$ consists of the orthogonal bases of the principal subspace, and $\mathbf{v}_i \perp \mathbf{v}_j$. That allows one to calculate the expansion coefficients α_i using direct projection

$$\alpha_i = < (\Phi(\mathbf{x}) - \bar{\Phi}), \mathbf{v}_i > \tag{4.22}$$

Now consider the calculation of $\mathbf{v}_i$ and α_i. Given a collection of training feature vectors $\Phi(\mathbf{x}_j)$, the corresponding principal components must lie in the subspace spanned by $\Phi(\mathbf{x}_j)$.

$$\mathbf{v}_i = \sum_{j=1}^{M} \gamma_{ij}(\Phi(\mathbf{x}_j) - \bar{\Phi}) \tag{4.23}$$

where the coefficient vector $\gamma_i := (\gamma_{i1}, \ldots, \gamma_{iM})^T$ is a vector of expansion coefficients . The coefficients are determined with an eigenvector problem [Scholkopf et al., 1998]

$$M\lambda\gamma = \tilde{K}\gamma \tag{4.24}$$

for nonzero eigenvalues λ. Here $\tilde{K}_{ij} := < (\Phi(\mathbf{x}_i) - \bar{\Phi}), (\Phi(\mathbf{x}_j) - \bar{\Phi}) >$. It follows that

$$\begin{aligned} \alpha_i &= \sum_{j=1}^{M} \gamma_{ij} < (\Phi(\mathbf{x}) - \bar{\Phi}), (\Phi(\mathbf{x}_j) - \bar{\Phi}) > \qquad (4.25) \\ &= \sum_{j=1}^{M} \gamma_{ij} \left[k(\mathbf{x}, \mathbf{x}_j) - \frac{1}{M}\sum_{n=1}^{M}(k(\mathbf{x}, \mathbf{x}_n) - k(\mathbf{x}_j, \mathbf{x}_n)) + \frac{1}{M^2}\sum_{n1}^{M}\sum_{n_2}^{M} k(\mathbf{x}_{n1}, \mathbf{x}_{n2}) \right] \end{aligned}$$

Applications to polynomials for F_{back}

It can be seen that, due to the presence of $\bar{\Phi}$, the expression of α in Eq. 4.21 is not compatible with the original polynomial formulations (Eq. 4.19 and Eq. 4.20). Thus, we need to rewrite the polynomial function by

$$\hat{x} = f_{back}(\Phi(\mathbf{x})) = (\Phi^T(\mathbf{x}) - \bar{\Phi})W_\phi(\Phi^T(\mathbf{x}) - \bar{\Phi}) + \beta_\phi^T(\Phi^T(\mathbf{x}) - \bar{\Phi}) + c_\phi \tag{4.26}$$

This formulation will lead to a polynomial on the subspace feature vector α, similar to Eq. (4.20). But the vector α here is given by Eq. (4.22) instead of Eq. (4.19).

Obviously, training an autoassociator amounts to estimating the parameters W,β and c from a given set of samples. Although the function Eq. (4.20) is nonlinear in the variable α, it can be favorably expressed as a linear function with respect to $\{W, \beta, c\}$

$$\hat{x} = \sum_{i=1}^{N_a} \sum_{j=1}^{N_a} \alpha_i \alpha_j w_{ij} + \sum_{i=1}^{N_a} \alpha_i \beta_i + c \tag{4.27}$$

Hence, the learning problem can be conveniently solved with linear algebra.

In the following, we show that there exists a polynomial function on kernel product $\mathbf{k}$ equivalent to that on α. In other words, one can avoid computing KPCA in running autoassociators, allowing fast implementation.

Let's first study the calculation of α in Eq. (4.25). A few terms there depend only on the training examples $\{\mathbf{x}_i\}$ while being irrelevant to the input pattern $\mathbf{x}$. They can be rewritten as

$$u_i = \sum_{n=1}^{M} k(\mathbf{x}_i, \mathbf{x}_n) + \frac{1}{M^2} \sum_{n1=1}^{M} \sum_{n2=1}^{M} k(\mathbf{x}_{n1}, \mathbf{x}_{n2}) \tag{4.28}$$

It follows that

$$\alpha_i = \gamma_i^T [\mathbf{k} + \mathbf{u}] - \frac{1}{M} (\gamma_i^T 1_M)(\mathbf{k}^T 1_M) \tag{4.29}$$

where 1_M is a M-unit long vector of all 1's. Denoting $\Gamma = (\gamma_1, \gamma_2, \ldots)$, the whole vector α reads

$$\alpha = \Gamma^T [\mathbf{k} + \mathbf{u}] - \frac{1}{M} (\Gamma^T 1_M 1_M^T)\mathbf{k} = \Gamma^T [\mathbf{k} + \mathbf{u}] - (\Gamma^T E)\mathbf{k} \tag{4.30}$$

where E is an $M \times M$ matrix consisting of all 1's.

Substituting the expression for α in Eq. (4.20), the equation becomes

$$\hat{x} = f_{back}(\Phi(\mathbf{x})) = \mathbf{k}^T W_k \mathbf{k} + \beta_k^T \mathbf{k} + c_k \tag{4.31}$$

which is a multivariate polynomial on the kernel product vector $\mathbf{k}$. Details of W_k, β_k and c_k are given in Table 4.1, which reveals the relationship between a polynomial function on α with its equivalence on the kernel product vector $\mathbf{k}$. Thus, running autoassociators will use pre-computed W_k, β_k and c_k without computing kernel principal analysis.

4.3 Regularization of Kernel Polynomials

Learning machines may face the so-called "over-fitting" problem in which the machines specialize well to training samples but generalize poorly to new patterns. To solve

Table 4.1: The polynomial in KPCA subspace versus that on kernel products.

$\alpha^T W\alpha + \beta^T\alpha + c$	$\mathbf{k}W_k\mathbf{k} + \beta_k\mathbf{k} + c_k$
W	$W_k = (\Gamma^T(I-E))^T W\Gamma^T(I-E)$
β	$\beta_k = 2(\Gamma^T(I-E))^T W\Gamma^T\mathbf{u} + (\Gamma^T(I-E)^T\beta$
c	$c_k = (\Gamma^T\mathbf{u})^T W\Gamma^T\mathbf{u} + \Gamma^T\beta + c$

Note: I denotes the identity matrix.

the problem and enhance the generalization performance, a common approach is by regularization which aims to stabilize the solution by means of some auxiliary non-negative functional that embeds prior information about the solution [Morozov, 1984]. The prior information usually involves an assumption that the input-output mapping function is smooth, in the sense that similar inputs correspond to similar outputs. Since autoassociator tries to produce the same pattern at output as input, this assumption is also justifiable for our kernel networks.

Now consider the regularization for kernel autoassociators with polynomial backward mapping f_{back}. We define the roughness of a kernel polynomial by

$$R_r(f_{back}) = \int ||\mathrm{D}f_{back}||^2 d\alpha = \int \left\| \frac{\partial f_{back}(\alpha)}{\partial \alpha} \right\|^2 d\alpha \tag{4.32}$$

where α is the kernel features by KPCA (see Eq. (4.25)). And D is a linear differential operator. For simplicity, all the feature vectors α are supposed to be normalized such that $0 \leq \alpha \leq 1$. That leads to an efficient way to compute the roughness as below.

4.3.1 Roughness of Polynomial Functions

Let's consider the polynomial function f_{back} in Equation 4.20. Its first order differential regulator reads

$$\begin{aligned} \mathrm{D}f_{back} &= \left\| \frac{\partial f_{back}(\alpha)}{\partial \alpha} \right\|^2 \\ &= (\beta + 2W\alpha)^T(\beta + 2W\alpha) \\ &= \beta^T\beta + 4\beta^T W\alpha + 4\alpha^T W^T W\alpha \\ &= s_1 + s_2 + s_3 \end{aligned} \tag{4.33}$$

where $s_{1,\ldots,3}$ represent accordingly the three terms above. Given the kernel feature vector α whose elements range in [0,1] (after normalization), the integral of the three

components of $\mathrm{D}f_{back}$, i.e. s_1, s_2 and s_3 would be

$$\begin{aligned}
\int s_1 d\alpha &= \int \beta^T \beta d\alpha = \beta^T \beta && (4.34)\\
\int s_2 d\alpha &= 4\int (\beta^T \mathbf{w}^{(1)} \alpha_1 + \beta^T \mathbf{w}^{(2)} \alpha_2 + \cdots) d\alpha && (4.35)\\
&= 2(\beta^T \mathbf{w}^{(1)} + \beta^T \mathbf{w}^{(2)} + \cdots) = 2\sum_{i=1}^{N_a} \beta^T \mathbf{w}^{(i)}\\
\int s_3 d\alpha &= \int 4\alpha^T (W^T W) d\alpha && (4.36)\\
&= 4\int (\mathbf{w}^{(1)T} \mathbf{w}^{(1)} \alpha_1 \alpha_1 + \mathbf{w}^{(1)T} \mathbf{w}^{(2)} \alpha_1 \alpha_2 + \cdots) d\alpha_1 \alpha_2 \ldots \alpha_{N_a}\\
&= (\int p_{11} d\alpha_1^3 + \int p_{12} d\alpha_1^2 \alpha_2^2 + \cdots) + \frac{1}{3}(\int p_{11} d\alpha_1^3 + \int p_{22} d\alpha_2^3 + \cdots)]\\
&= \sum_{i=1}^{N_a}\sum_{j=1}^{N_a} p_{ij} + \frac{1}{3}\sum_{i=1}^{N_a} p_{ii}
\end{aligned}$$

where $\mathbf{w}^{(i)}$ denotes the ith column vector in the matrix W, and p_{ij} represents $\mathbf{w}^{(i)T}\mathbf{w}^{(j)}$. Summing up the above equations gives rise to the expression of the roughness measure:

$$R_r(f_{back}) = \beta^T \beta + 2\sum_{i=1}^{N_a} \beta^T \mathbf{w}^{(i)} + \sum_{i=1}^{N_a}\sum_{j=1}^{N_a} \mathbf{w}^{(i)T}\mathbf{w}^{(j)} + \frac{1}{3}\sum_{i=1}^{N_a} \mathbf{w}^{(i)T}\mathbf{w}^{(i)} \qquad (4.37)$$

where $\mathbf{w}^{(i)}$ is the ith column vector of the matrix W.

4.3.2 Regularization Algorithm

During training, we will consider not only empirical reconstruction errors but also the roughness of the network. Thus, the objective function is defined by

$$G(f_{back}) = \sum_{i=1}^{M} [x_i - f_{back}(\mathbf{x}_i)]^2 + \lambda R_r(f_{back}) \qquad (4.38)$$

where λ determines the trade-off between empirical error and the roughness.

Let's first decompose the objective function into two components respectively for the empirical reconstruction error $e_t(f_{back})$ and the roughness measure $R_r(f_{back})$.

$$e_t(f_{back}) = \sum_{i=1}^{M} (x_i - f_{back}(\alpha^{(i)}))^2 \qquad (4.39)$$

$$e_s(f_{back}) = \int ||\mathrm{D}f_{back}||^2 d\alpha \qquad (4.40)$$

The derivatives of e_t would be

$$\frac{\partial e_t(f_{back})}{\partial c} = 2\sum_{i=1}^{M} (x_i - f_{back}(\alpha^{(i)})) \frac{\partial (x_i - f_{back}(\alpha^{(i)}))}{\partial c} = -2\sum_{i=1}^{M} (x_i - f_{back}(\alpha^{(i)}) \qquad (4.41)$$

$$\frac{\partial e_t(f_{back})}{\partial \beta} = 2\sum_{i=1}^{M}(x_i - f_{back}(\alpha^{(i)}))\frac{\partial(x_i - (\alpha^{(i)T}W\alpha^{(i)} + \beta^T\alpha^{(i)} + c))}{\partial \beta} \quad (4.42)$$

$$= -2\sum_{i=1}^{M}(x_i - f_{back}(\alpha^{(i)}))\alpha^{(i)}$$

$$\frac{\partial e_t(f_{back})}{\partial w_{jk}} = 2\sum_{i=1}^{M}(x_i - f_{back}(\alpha^{(i)}))\frac{\partial(x_i - (\alpha^{(i)T}W\alpha^{(i)} + \beta^T\alpha^{(i)} + c))}{\partial w_{jk}} \quad (4.43)$$

$$= -2\sum_{i=1}^{M}(x_i - f_{back}(\alpha^{(i)}))\frac{\partial(\sum\sum_{m,n=1}^{M} w_{mn}\alpha_m^{(i)}\alpha_n^{(i)})}{\partial w_{jk}}$$

$$= -2\sum_{i=1}^{M}(x_i - f_{back}(\alpha^{(i)}))\alpha_j^{(i)}\alpha_k^{(i)}$$

The derivatives of R_r would be

$$\frac{\partial R_r(f_{back})}{\partial c} = 0 \quad (4.44)$$

$$\frac{\partial R_r(f_{back})}{\partial \beta} = \frac{\partial(\beta^T\beta + 2\sum_{i=1}^{N_a}\beta^T\mathbf{w}^{(i)} + \cdots)}{\partial \beta} \quad (4.45)$$

$$= 2\beta + 2\sum_{i=1}^{N_a}\mathbf{w}^{(i)}$$

$$\frac{\partial R_r(f_{back})}{\partial w_{jk}} = \frac{\partial(2\sum_{i=1}^{N_a}\beta^T\mathbf{w}^{(i)} + \sum\sum_{m,n=1}^{N_a}\mathbf{w}^{(m)T}\mathbf{w}^{(n)} + \frac{1}{3}\sum_{m=1}^{N_a}\mathbf{w}^{(m)T}\mathbf{w}^{(m)})}{\partial w_{ij}}$$

$$= 2\beta_j + \frac{\partial(\sum\sum_{m,n=1}^{N_a} w_{jm}w_{jn} + \frac{1}{3}w_{jk}w_{jk})}{\partial w_{jk}}$$

$$= 2\beta_j + 2\sum_{m=1}^{N_a} w_{jm} + \frac{2}{3}w_{jk} \quad (4.46)$$

Hence, the above derivatives of e_t and e_s directly lead to the derivatives of $G = e_t + e_s$ as below Eq. (4.38) with respect to W, β and c, and we have

$$\frac{\partial G(f_{back})}{\partial c} = -2\sum_{i=1}^{M}(x_i - f_{back}(\mathbf{x}_i)) \quad (4.47)$$

$$\frac{\partial G(f_{back})}{\partial \beta} = [-2\sum_{i=1}^{M}(x_i - f_{back}(\mathbf{x}_i))\alpha^{(i)}] + \lambda[2\beta + 2\sum_{i=1}^{N_a}\mathbf{w}^{(i)}] \quad (4.48)$$

$$\frac{\partial G(f_{back})}{\partial w_{jk}} = [-2\sum_{i=1}^{M}(x_i - f_{back}(\mathbf{x}_i))\alpha_j^{(i)}\alpha_k^{(i)}] + \lambda[2\beta_j + 2\sum_{m=1}^{N_a} w_{jm} + \frac{2}{3}w_{jk}] \quad (4.49)$$

Training a kernel autoassociator means minimizing the objective function Eq. 4.38. Because the function is continuous and differentiable, the minimization can be obtained at $\frac{\partial G(f_{back})}{\partial c} = 0$, $\frac{\partial G(f_{back})}{\partial \beta} = 0$ and $\frac{\partial G(f_{back})}{\partial w_{jk}} = 0$. To solve the minimization problem, we tentatively use Matlab nonlinear optimization toolbox in the implementation. For

the minimization, the second order partial derivatives of G can be useful and easily computed by

$$\frac{\partial^2 G}{\partial c^2} = 2M \tag{4.50}$$

$$\frac{\partial^2 G}{\partial c \beta} = 2\sum_{i=1}^{M} \alpha^{(i)} \tag{4.51}$$

$$\frac{\partial^2 G}{\partial c w_{jk}} = 2\sum_{i=1}^{M} \alpha_j^{(i)} \alpha_k^{(i)} \tag{4.52}$$

$$\frac{\partial^2 G}{\partial \beta_j c} = 2\sum_{i=1}^{M} \alpha_j^{(i)} \tag{4.53}$$

$$\frac{\partial^2 G}{\partial \beta^2} = 2\left[\sum_{i=1}^{M} \alpha^{(i)} \alpha^{(i)T}\right] + 2\lambda I \tag{4.54}$$

$$\frac{\partial^2 G}{\partial \beta_m w_{jk}} = 2 + 2\sum_{i=1}^{M} \alpha_j^{(i)} \alpha_k^{(i)} \alpha_m^{(i)} \tag{4.55}$$

$$\frac{\partial^2 G}{\partial w_{jk} c} = 2\sum_{i=1}^{M} \alpha_j^{(i)} \alpha_k^{(i)} \tag{4.56}$$

$$\frac{\partial^2 G}{\partial w_{jk} \beta} = 2\left[\sum_{i=1}^{M} \alpha_j^{(i)} \alpha_k^{(i)} \alpha^{(i)}\right] + 2\lambda \mathbf{d}(j) \tag{4.57}$$

$$\frac{\partial^2 G}{\partial w_{jk} w_{ln}} = 2\left[\sum_{i=1}^{M} \alpha_l^{(i)} \alpha_n^{(i)} \alpha_j^{(i)} \alpha_k^{(i)}\right] + 2\lambda\left[\frac{1}{3}\delta(j-l, k-n) + \delta(j-l, 0)\right] \tag{4.58}$$

where $\mathbf{d}(j)$ is a N_a-element vector of all 0 except the j-th element being 1. And $\delta(x, y)$ is the Dirac function whose value is zero across the entire space except $\delta(0, 0) = 1$.

4.3.3 Performance of Regularized Autoassociators

To examine the performance of the regularization networks, we use two problems (the Promoter recognition and the Sonar Target recognition) to be introduced later as test beds. In each problem, 3 experiments using 5-fold cross-validation techniques were conducted and results were averaged. A range of values for λ were tested to examine its effect on both the empirical reconstruction error and the roughness of the backward mapping function. The reconstruction error, the roughness and the recognition performance of the regularized networks were obtained and are plotted in Figure 4.2 and Figure 4.3 as functions with respect to λ.

The results indicate that our regularization method effectively controlled the roughness of the kernel networks. Although the smoother networks yielded larger error for

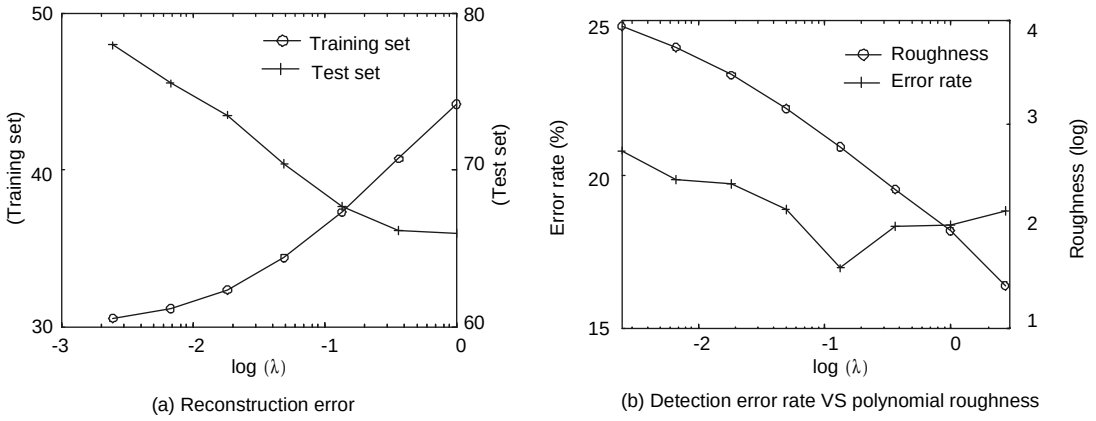


Figure 4.2: Regularized networks in the Promoter recognition problem.

Panel (a) shows the mean reconstruction errors for samples respectively from the training set and the test set, and the errors are drawn as the functions over the regularization parameter λ which controls the smoothness of the regularized networks. Panel (b) shows the roughness measure of the regularized networks and their performances in terms of recognition/detection error rates.

the training set, they produced better results for the test set and thus demonstrated better generalization performance. Furthermore, with λ in an appropriate range, the classification performance of kernel autoassociators can also be enhanced as shown in Figure 4.2 and Figure 4.3. In practice, the appropriate value for λ could be estimated empirically by naive searching with cross-validations on the training set. A possible negative consequence is much more computational complexities. It will be an interesting topic to automatically determine λ from the training set.

4.4 Nonlinear Learning with Autoassociators

This section aims to evaluate the capability of the proposed models for nonlinear concept learning. We generated an artificial dataset which consists of a few samples on a 2D spiral, as plotted in the upper left panel of Fig. 4.4. We constructed a kernel autoassociator with linear F_{back} and another one with polynomial F_{back} (hereafter referred to as KAA-1, KAA-2 respectively). Besides, an autoencoder with sigmoid transfer function was applied for comparison. The respective reproduction error surfaces are displayed as images in Fig. 4.4, where the error value is denoted by gray level (i.e. bright

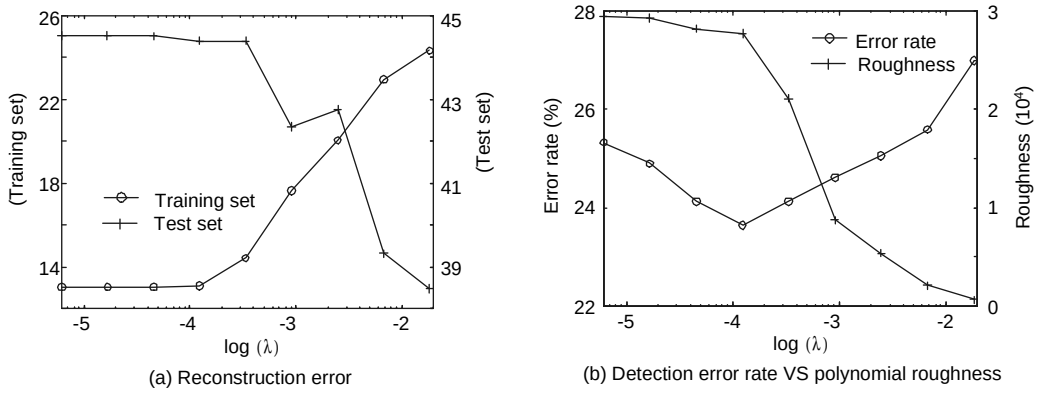


Figure 4.3: Regularized networks in the Sonar Target Recognition domain. *Please refer to Figure 4.2 for the explanation.*

means small). Particularly for the autoencoder we tested a various number (4 to 40) of hidden neurons, but obtained similar results. The configuration (including the kernel bandwidth and the number of kernel principal components) of kernel autoassociators is automatically set (see Section 4.5.4).

The experimental results show that, in spite of the complex structure of spiral patterns, the kernel autoassociators were able to produce reconstruction error surfaces that correctly reflected the data structure. On the other hand, the autoencoder produced a unimodal-Gaussian-alike error surface, akin to linear machines.

We also conducted a comparison between kernel density estimate [Scott, 1992] (a non-parametric technique for density estimation) and kernel autoassociators, by using a set of examples generated from a 3-mode 2D random distribution. The samples and the experimental results are plotted in Figure 4.5.

The results show that every method successfully captured the three modes by the estimated density surface or reconstruction error surface. In particular, with the same kernel bandwidth (σ), kernel autoassociators appear to produce smoother surfaces than kernel density estimate. Thus, they tend to have better generalization capability while maintaining good specialization capability.

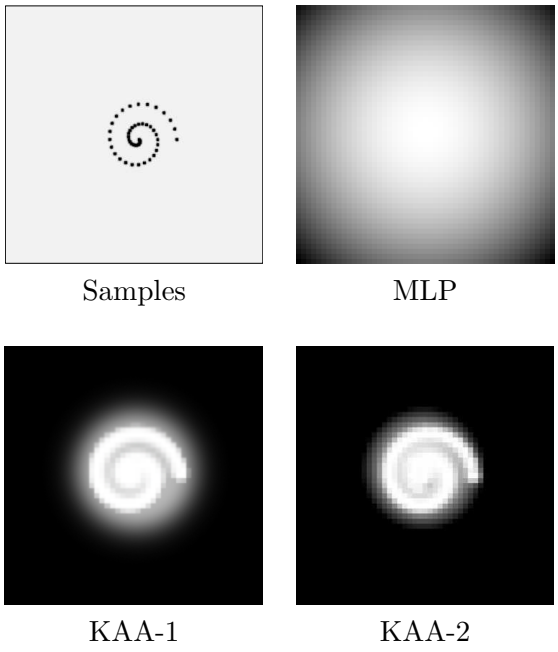


Figure 4.4: Concept learning on spiral pattern.

Results are shown as reconstruction error surfaces. KAA-1 or KAA-2 represents kernel autoassociator with linear or polynomial reconstruction functions. Note the results with KAA-1 and KAA-2 were thresholded because some reconstruction errors (outsides the region of interest) can be extremely large.

4.5 Applications to Novelty Detection

As mentioned earlier, novelty detection is the identification of novel patterns of which the learning system is given few samples in the training stage. This problem happens when novel or abnormal examples are expensive or difficult to obtain. Here we consider two specific problems, i.e. promoter detection and sonar target recognition.

The ***Promoter*** problem takes as input segments of DNA, some subset of which represent promoters. A promoter is a sequence that signals to the chemical processes acting on the DNA where a gene begins. The goal of the problem is to train a classifier that is able to detect promoters – the novel patterns. The ***Sonar Target*** recognition problem takes as input the signals returned by a sonar system in the cases where mines and rocks were used as targets. We choose mine patterns as novelty.

We acquired both the Promoter database and the sonar target database from *UCI Machine Learning Repository*. In particular, the promoter database consists of 106 samples, 53 for promoters while the others for non-promoters. Each sample is a 57-unit

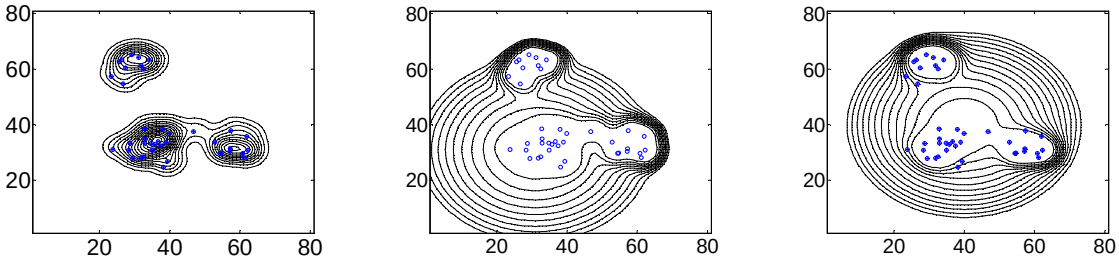


Figure 4.5: Results of concept learning on multimodal pattern.

The results are shown as probability density surface (by kernel density estimate) or reconstruction error surface (by kernel autoassociators).

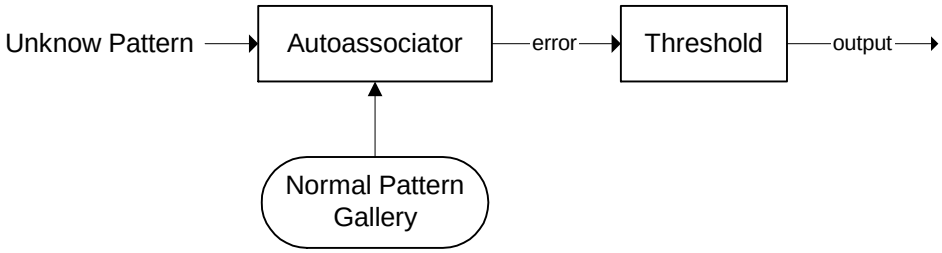


Figure 4.6: Novelty Detection Scheme.

An autoassociator learns normal examples; when an unknown pattern is presented, the reconstruction error by the autoassociator will be compared with a threshold to signal whether it is a novel pattern (with larger error) or a normal pattern (with smaller error).

long string composed of four chars $\{a, c, g, t\}$, which we convert to $\{1, 2, 3, 4\}$ in the experiment. The sonar target database comprises 111 positive and 97 negative patterns (60-unit long).

The autoassociator detection system relies on the fact that an autoassociator is designed to learn normal patterns, thus the network would tend to produce relatively larger reproduction errors for novel patterns. The errors can be thresholded to signal novelty. Figure 4.6 plots the detection scheme.

The specific threshold (ξ) is crucial for the system performance, and should be chosen carefully. In respect of the threshold setting, there are two different cases in novelty detection. In one case, a small number of novel patterns are available for training the detection system; in the other case, one can obtain merely normal patterns for the training. For the two cases, we propose and study two different approaches.

4.5.1 Novelty detection with novel examples

With samples from both novel and normal patterns, novelty detection can be viewed as a usual binary classification problem. The goal is to find a rule that best separates the positive and negative classes.

Given an unknown object $\mathbf{x}$, the system will produce a reconstruction error $e(\mathbf{x})$ and the probability of $\mathbf{x}$ to be identified as novelty is given by $P(e(\mathbf{x}) > \xi)$. Let the novel class be ω_1 and the normal class be ω_0. The probability of misclassification is

$$\begin{aligned} P_e &= P(e(\mathbf{x}) < \xi, \omega_1) + P(e(\mathbf{x}) > \xi, \omega_0) \\ &= \int \left[P(e(\mathbf{x}) < \xi) p(\mathbf{x}, \omega_1) + P(e(\mathbf{x}) < \xi) p(\mathbf{x}, \omega_0) \right] d(\mathbf{x}) \end{aligned} \tag{4.59}$$

Here $P(e(\mathbf{x}) < \xi, \omega_1)(P(e(\mathbf{x}) > \xi, \omega_0))$ is the probability of a novel (normal) pattern classified as normal (novel). The empirical value of the above error is given by

$$P_e = \frac{1}{N_i} \sum_{\mathbf{x}_i \in \omega_1} P(e(\mathbf{x}_i) < \xi) + \frac{1}{N_j} \sum_{\mathbf{x}_j \in \omega_0} P(e(\mathbf{x}_j) > \xi) \tag{4.60}$$

where $\mathbf{x}_i$ ($\mathbf{x}_j$) is a random sample generated from the distribution $p(\mathbf{x}, \omega_1)(p(\mathbf{x}, \omega_0))$, and N_i or N_j denotes the number.

For the given samples and an autoassociator, $e(\mathbf{x}_i)$ is fixed and $P(e(\mathbf{x}_i) < \xi)$ takes binary value (0 or 1) that depends on the threshold ξ. Thus, setting the threshold becomes equivalent to seeking a ξ that minimizes P_e in Eq. 4.60. Since ξ is a one dimensional variable, it can be easily determined with a simple searching program.

We set up an autoassociator system with the above threshold setting method, and tested it over the aforementioned two detection problems. Parzen-window novelty detectors (ParzenND) [Yeung and Chow, 2002] – a kernel density estimation technique for novelty detection, and autoencoders, are compared using the same threshold setting method.

In the experiment, each method was evaluated using 5-fold cross-validation (see [Haykin, 1999] p.213): the whole data set was randomly partitioned into 5 subsets, and at every fold one subset was picked out for testing while the remainder was used for training. To enhance the accuracy of evaluation, the classification error rates reported here are averaged over 10 tests.

Table 4.2 lists the detection error rates (See the table for the explanation of the symbols). It shows that in both domain studies, KAA-2^P, KAA-2^G and KAA-1^G outperformed the Parzen-window method and the autoencoder method. The KAA-1^P system, on the other hand, achieved moderate results similar to the Parzen-window method. Note that all the compared methods were empirically tuned to their best performance in the experiment.

Table 4.2: Novelty detection accuracy with novel examples

	Novelty Detection Approach					
	KAA-1^G	KAA-2^G	KAA-1^P	KAA-2^P	Parzen-window	Autoencoder
Promoter	20.4 ± 7.4	18.1 ± 9.9	26.1 ± 9.1	19.4 ± 8.9	22.7 ± 8.4	30.3 ± 13.2
Sonar	27.0 ± 5.5	26.2 ± 7.4	33.9 ± 6.1	27.5 ± 6.7	28.5 ± 6.5	32.6 ± 5.3

The superscript P denotes polynomial kernel function, while G denotes Gaussian kernel function. Numbers after each "±" are standard deviations of the detection accuracy.

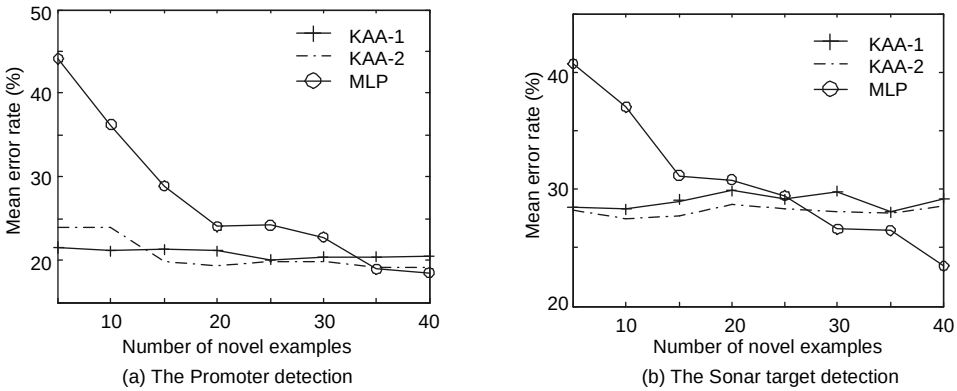


Figure 4.7: Recognition error rates over the number of novel examples in the two novelty detection problems.

The number of novel examples used for training ranges from 5 to 40.

In addition, we examined the systems' sensitivities in the two domains to small numbers of novel examples. A typical discriminant machine – a MLP classifier – is also compared. Fig. 4.7 shows the results, which indicate that neither KAA-1 nor KAA-2 is sensitive to the small number of novel examples, but the performance of MLP classifier deteriorated along with the decreasing number of novel examples.

4.5.2 Novelty detection without novel examples

In the case without novel samples for training, the above method to determine the novelty detection threshold ξ is not applicable. Instead, we adopt a method from [Yeung and Chow, 2002] that is designed to achieve a given *false detection rate* (FDR):

$$\upsilon = P(e(\mathbf{x}) > \xi | \mathbf{x} \in \omega_0) \tag{4.61}$$

Similar to Eq. 4.60, the equation can also be expressed in terms of samples, and υ will become a function with respect only to ξ. Therefore, for a given FDR, ξ can be easily determined with a one-dimensional search procedure.

The same threshold setting method is used to compare kernel autoassociators with autoencoders, SOM-ND [Ypma and Duin, 1998] and a Parzen-window novelty detector ([Yeung and Chow, 2002]). The Parzen-window detector is a non-parametric density estimation technique for novelty detection, while SOM-ND is a novelty detection technique based on self-organizing maps.

We conducted experiments with five-fold cross-validation, which is the same as in the previous subsection. The results are summarized in Table 4.3. Here $\upsilon = 0.1$.

Table 4.3: Novelty detection accuracy without novel examples

	KAA-1^G	KAA-2^G	KAA-1^P	KAA-2^P
Promoter	24.2 ± 7.5	20.7 ± 8.6	29.0 ± 11.0	27.9 ± 5.5
Sonar	31.6 ± 10.7	28.2 ± 6.6	34.9 ± 7.6	29.3 ± 7.9

	Parzen-window	Autoencoder	SOM-ND
Promoter	23.9 ± 9.1	27.7 ± 10.2	26.5 ± 9.4
Sonar	29.5 ± 5.8	37.0 ± 8.2	32.5 ± 7.03

The table shows that KAA-2^G achieved the best detection results. Other techniques are comparable in terms of detection accuracy.

4.5.3 Autoassociator-based novelty detection against noise

To examine the robustness of the proposed method, we conduct the experiment where the Gaussian noise is added to the test data in the Promoter database, independently

distributed over each dimension with standard deviation from 0 to 1.5. Figure 4.5.3 shows the detection results.

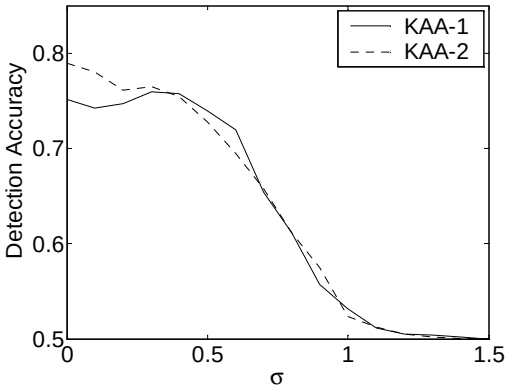


Figure 4.8: Kernel autoassociators against noise for the Promoter detection. *Here shows the resulting detection accuracies (without novel examples) as functions over the level (standard deviation) of additive zero-mean Gaussian noise.*

It needs to be mentioned that the standard deviation of the Promoter patterns is about 1.1 on each dimension, while the detection accuracy remains larger than 70% under the noise up to $\sigma = 0.6$. The results demonstrate that the kernel machines are not very sensitive to additive noise.

4.5.4 Discussions on Novelty Detection

This section has studied the autoassociator-based novelty detection system in various situations. The results indicate that either with or without novel examples, kernel autoassociators (especially the KAA-2^G) achieved slightly better results than Parzen-window detectors, and significantly outperformed autoencoders or SOM-ND detectors. Furthermore, the KAA systems demonstrated consistent performance against a varying number of novel examples, in contrast to the MLP classifier that requires a large number of novel examples for good performance.

Comparing the autoassociators in the two detection domains, those using Gaussian kernels seem to outperform others using polynomial kernels. Furthermore, the experimental results also attest to the robustness of Gaussian kernel autoassociators against additive noise.

The architectures of kernel autoassociators are determined by only a few parameters. For kernel autoassociators with linear backward mapping f_{back}, the user needs only to choose between different types of kernels (Gaussian, polynomial, etc.). For kernel autoassociators with polynomial f_{back}, the user may need to choose an additional parameter - the number (N_a) of kernel principal components. But in practice, the number N_a can be automatically selected to account for a high percentage of variance of the samples (e.g., 90% in the tests). Our empirical study also suggests that the ultimate performance of kernel autoassociators is not very sensitive to N_a.

However, as Markou and Singh put it [Markou and Singh, 2003a], there is no single best model for novelty detection and the success depends not only on the type of method used but also statistical properties of data handled. Thus, a specific parameter-setting method can not be always suitable for different problems. And in some cases we may resort to empirical methods such as cross-validation for parameter setting ([Haykin, 1999], page 213).

It's worthwhile to mention that Markou and Singh have summarized important principles related to novelty detection [Markou and Singh, 2003a]. The principle of parameter minimization states that a novelty detection method should aim to minimize the number of parameters that are user set. Kernel autoassociators inherit an advantage of neural networks for novelty detection in that during network training, *a priori* information is not very critical on data distribution [Markou and Singh, 2004]. Furthermore, to define a kernel autoassociator the user only needs to select one essential parameter: the kernel type, since the other parameters such as the number of principal components can be learned from the training samples. Hence, the proposed model adheres more closely to this principle than conventional autoencoders which need to define a number of parameters including the number and the dimension of layers, and the transfer functions. Besides, the principle of independence asserts that the novelty detection method should show reasonable performance in the context of imbalanced data set, small number of samples, and noise. Our experiments in Section 5.A and 5.C indicate that the proposed method with kernel autoassociators offers satisfactory performance in such a context. In addition, since the training/running of kernel autoassociators is of low com-

putational complexity, the networks enable online adaptation which is in accordance with the principle of adaptability and the principle of computational complexity.

4.6 Applications to Multi-Class Classification

As one-class learning machines, kernel autoassociators can be used for multi-class classification tasks, if each autoassociator is associated with an individual category to learn its concept. The system would use a competitive classification scheme shown in Figure 4.9 : when a test pattern is presented, it will be reproduced by each autoassociator, the respective reproduction results will be collected and compared, with the best one indicating the corresponding class. In the subsections to follow we examine this classification scheme on various recognition problems.

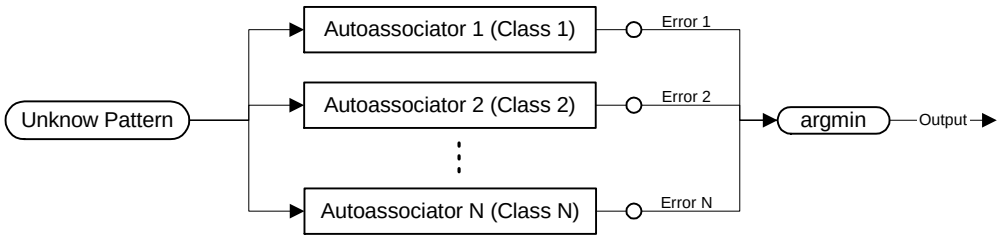


Figure 4.9: Multi-Class Classification Scheme based on Autoassociators.
Each autoassociator is trained with the examples from its associated class. When an unknown pattern is presented, it will be processed by all the networks respectively. The system will collect and compare the reconstruction errors, and pick out the network with the minimal error.

4.6.1 Wine and Glass Recognition

The *Wine Recognition* data [Aeberhard et al., 1992] is acquired from UCL Repository of Machine Learning Databases. The data set contains 3 types of wines, each type has 59, 71 or 48 instances. The analysis of the wines determines the quantities of 13 constituents.

The *Glass Recognition* data is also acquired from UCL Repository of Machine Learning Databases. The data set contains instances of 6 types of glasses. Each type has 70, 17, 76, 13, 9 or 27 instances. The goal is to determine the glass type from 9 attributes.

We used a 2-fold cross-validation technique to test the autoassociator-based classi-

fication scheme in these two recognition domains. Typical classification machines such as multilayer perceptrons and support vector machine were tested for comparison. Each network was tuned to achieve the best performance. The comparative recognition results are summarized in Table 4.4. Besides, since neural networks are usually sensitive to the pattern scale, in the experiment we use a pre-processing program to normalize the Wine and Glass data to the value range [-1, 1]. (In the other experiments the acquired data were already in normalized form.)

	KAA-1	KAA-2	Autoencoder	MLP	SVM
Wine	95.8 ± 2.7	96.9 ± 1.5	90.5 ± 3.37	92.6 ± 5.5	98.3 ± 1.3
Glass	62.1 ± 4.2	62.6 ± 5.4	59.3 ± 4.5	43.1 ± 6.1	57.8 ± 5.9

Table 4.4: Comparative results of wine and glass classification

The results show that kernel autoassociators are comparable to support vector machines in terms of recognition accuracy, and clearly outperforms both autoencoders and multilayer perceptrons.

4.6.2 Handwritten Digit Recognition

Here we consider the handwritten digit recognition problem with the US-Postal Service (USPS) handwritten digit database that consists of 7291 training images and 2007 test images (16×16 pixels).

To illustrate the classification process with autoassociators, some examples from the test set are shown in Figure 4.10. The leftmost column displays the probe patterns, to the right their respective reproductions by the KAA-2 networks corresponding to '0' to '9'. The first three rows in the figure show that three patterns of '0'/'2'/'3' get best reproduction from the correct networks (i.e. the 1st/3rd/4th network). There exists only a few cases in which the correct network among all networks cannot reproduce the best reproduction for a new pattern. An example is given in the bottom row where a pattern of class '7' is misclassified as '9'.

Table 4.5 summarizes the classification results in comparison with the published scores of other technologies. In addition, a newly proposed kernel Fisher discriminant method [Juwei et al., 2003] called KDDA, as well as KPCA-NN – a 1-Nearest-Neighbour

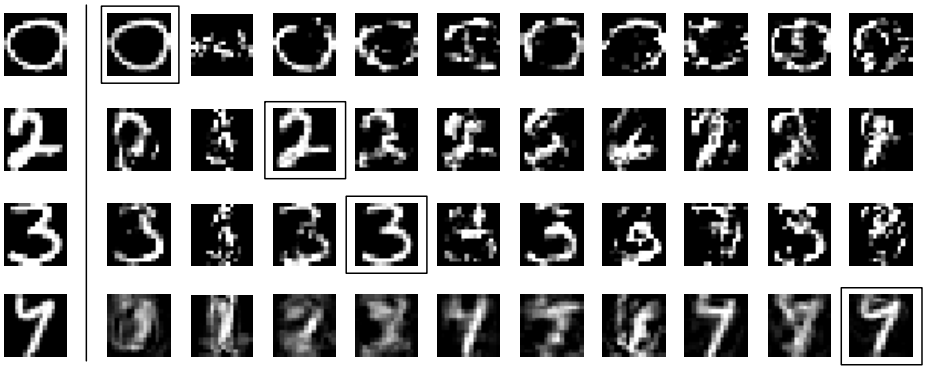


Figure 4.10: Examples of handwritten digit recognition with kernel-autoassociator classifier on the USPS database.
The leftmost column displays the test patterns, to the right their reconstructions by each kernel autoassociator, with the best results being singled out by a frame.

technique with KPCA features (see [Juwei et al., 2003]), has been implemented for comparison. The results in the table demonstrate that KAA-1 and KAA-2 could match LeNet and SVM in terms of performance while significantly outperforming autoencoders or kernel density baseline (a kernel density estimate).

In the comparison, the tangent distance approach yielded the best result. This can be explained by the fact that it is the only one here that dedicatedly explores domain knowledge about the invariance of image patterns. Similarly, there is a variant of SVM referred to as virtual SVM [Scholkopf, 1997] that, by incorporating prior knowledge about image invariance, gained considerable improvement and produced a good classification error rate of 3.0% on the USPS database. Similarity, incorporating domain knowledge may also yield a promising extension of kernel autoassociators.

Table 4.5: Recognition error rates on USPS.

Method	Error Rate
KAA-1	4.38%
KAA-2	4.68%
Autoencoder	7.42%
Kernel Density baseline [Keysers et al., 2000]	5.5%
Nearest Neighbor [Keysers et al., 2000]	5.7%
KDDA	10.0%
KPCA-NN	6.15%
LeNet1[LeCun et al., 1999]	4.2%
SVM[Scholkopf et al., 1995]	4.0%
Tangent Distance[Simard et al., 1998]	2.5%

4.7 Summary

In this chapter we have proposed a novel nonlinear autoassociator model. By making use of kernel feature space, the model resorts to relatively simpler functions (linear and polynomials) for autoassociation learning, in contrast to conventional nonlinear machines using a complex class of functions.

The kernel autoassociator model has been applied to novelty detection with or without novel examples. Experimental results show evidence of the robustness of the system in the context of imbalanced data set, small number of samples, and noise. Furthermore, the model has an intrinsic advantage over conventional nonlinear autoassociators in terms of fast training and easy tuning, as well as much less user-set parameters.

The multi-class recognition scheme based on autoasociators is a detection-based approach [Japkowicz et al., 1995]. Thus it has an inherent advantage over discriminant-based methods in terms of adaptability. Particularly, when a new subject is added to the system, the detection-based approach needs only to create a new network for the subject without re-training the other networks. On the other hand, discriminant-based methods usually need to retrain all the classifiers in the system.

Besides, the simulations and the extensive experiments show that the proposed method can capture complex nonlinear features and clearly outperform autoencoders. Compared with other systems, kernel autoassociators offer better or comparable performance, though they are generic one-class learning machines. In conclusion, the proposed method provides an alternative approach to nonlinear autoassociation modeling, and is promising in various nonlinear concept learning and recognition applications.

Chapter 5

Kernel Autoassociator Model for View-based Face Recognition

5.1 Introduction

The thesis aims to develop efficient view-based models not only for capturing the movement, but also for identifying the objects of interest despite large variations in the images. The tracking problem about affine transformation has been addressed in Chapter 2 and 3. We have also presented in Chapter 4 a generic pattern recognition scheme. It is of importance to extend the scheme to specific recognition issues, especially face recognition which is recognized as a typical visual recognition problem and has been a very active research topic in pattern recognition and computer vision communities in the last decade. It has a wide range of applications such as content-based indexing, identity authentication, access control and surveillance. Two excellent reviews can be found in [Chellappa et al., 1995, Zhao et al., 2000]. This chapter focuses on how to incorporate domain knowledge about face images into kernel autoassociators to produce a high-performance face recognition system.

Face recognition generally remains very challenging in terms of 1) large intra-class variance – complex and large variations in face images from changes in image condition, facial expression or pose; 2) small inter-class variance – close similarity between many individual faces; 3) practical limitation in face examples – only one or several examples per subject for a large population.

Two major issues are critical for addressing the face recognition problem. The first one is how to create efficient representations for face images; the second one is how to

classify a face using the selected representation and few examples per subject.

Selecting an appropriate image representation is crucial to the success of face recognition. Psychophysical studies suggest that the visual perception tasks such as similarity judgment tend to operate on a low-dimensional representation of the sensory data [Edelman, 1999]. One natural approach is to resort to low-dimensional geometric representations. For example, [Kanada, 1973] proposed to use geometric features based on the relative positions of eyes, nose and mouth [Kanada, 1973]. The pre-requisite for the success of this approach is an accurate facial feature detection scheme, which still remains a very difficult problem.

A widely used alternative methodology is by holistic or local image decomposition with some special 2D signals (so-called image kernels). A typical example is Eigenfaces which apply the technique of principal component analysis (PCA) to the pictorial domain, and represent faces by linear combinations of appropriately selected principal components [Turk and Pentland, 1991]. But Eigenfaces are only capable of capturing 2nd order pairwise dependency between pixels, though it is widely recognized that much of the discriminating information is contained in higher order statistics of face images [Bartlett and Sejnowski, 1997].

Some extensions of Eigenfaces have been proposed for face recognition, e.g. probabilistic principal component analysis (PPCA) [Moghaddam and Pentland, 1997] and local feature analysis (LFA) [Penev and Atick, 1996]. The PPCA method uses eigenspace decomposition for probability density estimation on high-dimensional data, and applies the densities to formulate a maximum-likelihood estimation framework for object detection and recognition. The LFA method extracts, from the holistic PCA bases, local topographic representations with local features. But it has been reported that the LFA method does not necessarily outperform PCA in terms of recognition performance [Donato et al., 1999].

Independent Component Analysis (ICA) [Bartlett and Sejnowski, 1997, Comon, 1994b] can also be viewed as an extension of PCA. It projects a raw image linearly onto some statistically independent components, accounting for higher order statistics. It has demonstrated superior performance over PCA for face recognition[Liu and Wechsler, 1999].

In order to accurately capture the local features in face images, a strict spatial-frequency analysis is often desirable. Wavelet analysis is particularly useful for this purpose since it has the unique characteristics of space-frequency localization. In computer vision, the multi-resolution scheme in wavelet analysis has been justified by psycho visual research as human visual system processes images in a multiple-scale way. Among various type of wavelet functions, Gabor functions provide the best possible trade off between spatial resolution and frequency resolution [Gabor, 1946]. There is also a strong biological relevance of processing images by Gabor wavelets as they have similar shapes as the receptive fields of simple cells in the primary visual cortex (V1) [Daugman, 1988].

In the last several years, a number of Gabor wavelets based methods have been proposed for face recognition, such as Gabor-Fisher classifier [Liu and Wechsler, 2002] and elastic bunch graph (EBG) [Wiskott et al., 1997]. Liu et al. employs traditional Gabor filtering together with an enhanced Fisher linear discriminant model (EFM) to effectively reduce the dimension of feature vectors produced by Gabor filtering. On the other hand, a EBG extracts certain sets of Gabor coefficients called Jets at predefined positions, which can substantially represent the local spatial and local frequency features on a face image.

Unlike EBG with manually determined feature positions and Gabor functions, a recently proposed Gabor wavelet network (GWM) [Krueger, 2001] represents an image by a set of weighted Gabor wavelets that vary in continuous state space. GWN has been applied to face recognition [Krueger, 2001], where one GWN is created for each face example. Given a probe image, the system will encode it with each GWN in the gallery, and comparing the encoded probe image against examples in the gallery. Because a GWN was designed to represent a single static image rather than a set of images, what it captures may be more likely to be transient features in a particular image than essential and consistent features that identify the subject.

For the second major issue in face recognition, many discrimination techniques have been proposed, for example, Fisher linear discriminant(FLD) and Neural Networks classifiers. FLD separates two classes by simultaneously maximizing between-class scatter and minimizing within-class scatter. A major drawback of FLD in face recognition is

the critical demand for a large training set for good generalization, which may be very difficult to meet. Examples of applying neural networks in face recognition include (1) the convolutional neural network (CNN) [Lawrence et al., 1997], which is a hybrid approach by combining self-organizing map(SOM) and a convolutional neural network; and (2) the probabilistic decision-based neural networks [Lin et al., 1997].

Another kind of neural networks that has been actively researched in face recognition is the associative memory (AM) [Kohonen, 1980]. Essentially, AM-based classification learns how to do recognition by categorizing positive examples of a subject. There has been a long history of associative memory research and the continuous interest is due to a number of attractive features of these networks, such as content addressable memory, collective computation capabilities etc. Kohonen is the first to illustrate some useful properties of auto-associative memory with faces as stimuli [Kohonen, 1977].

In this chapter, we study how to extend kernel autoassociator models (introduced in last chapter) for face recognition. First, we examine the direct application scheme which uses kernel autoassociators to learn face images. Next, we propose a novel face representation model called Gabor Wavelet Associative Memory (GWAM) by incorporating domain knowledge with a subject dependent Gabor wavelet network. The domain knowledge here used is that an individual face has a certain configuration of local and global image features such that we can develop a set of special image kernels to represent them. Finally, we carry out extensive experiments to evaluate a GWAM-based face recognition system. Three publicly available benchmark face databases, including the FERET, the ORL and the AR face databases, are used in the experimental study.

The rest of the chapter is organized as follows. Section 2 studies the direct applications of kernel autoassociators to face recognition. Section 3 starts by proposing a subject dependent Gabor wavelet network (SDGWN) for face representation, followed by the presentation of the Gabor wavelet associative memory (GWAM) model. In Section 4, we describe the experimental results. A summary is given in Section 5.

5.2 Direct Application and Performance

In the direct applications of kernel autoassociators to face recognition, the classification system essentially inherits the architecture of the multi-class classification scheme illustrated earlier in Figure 4.9. In other words, the system comprises a collection of autoassociators, each one is trained with face examples from its associated subject. When an unknown face is presented, it will be processed by all the networks respectively. The system will collect and compare the reconstruction errors, and pick out the one that produces the minimal error. Thus, no domain knowledge about faces is applied in this system. The performance of the recognition scheme is studied in the following.

Experiments on UMIST Database

The UMIST database [Graham and Allinson, 1998] consists of 575 gray-scale images for 20 subjects, each covering a wide range of poses from profile to frontal views. Some UMIST faces are shown in Figure 5.1, where large variations can be clearly observed.

Figure 5.1: Complex patterns present in multiview face recognition (examples from the UMIST database)

We divide the data into a training set and a test set. The training set consists of 6 images per person, whilst the remainder images in the database constituting the test set. In the experiment KPCA-NN (a 1-Nearest-Neighbour technique with KPCA features) and KAA-2 (kernel autoassociators with polynomial reverse mapping) both use 40 principal components. In addition, two non-kernel techniques including 1-Nearest-Neighbour (NN) and autoencoders are also compared. We have tuned each system and picked out their best results in the comparison. Final results in terms of recognition error rate over σ (the bandwidth of Gaussian kernel k) are plotted in Fig. 5.2.

The results indicate that the kernel autoassociator methods can produce better recognition than the others except KDDA. Note that they outperformed KDDA earlier in the OCR experiment (see the last chapter). And, the kernel autoassociator approach ap-

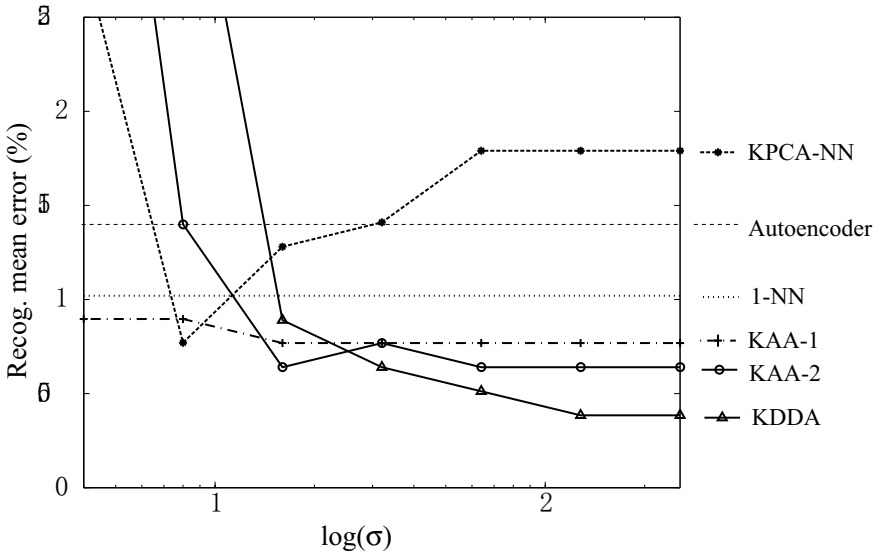


Figure 5.2: Comparative face recognition results on the UMIST database. *Here shown the recognition error rates as functions of the bandwidth σ of Gaussian kernels used by the kernel machines. Note 1-NN represents 1-Nearest-Neighbor technique, and it serves as a baseline together with another non-kernel machine, i.e. an autoencoder.*

pears to be overall more robust across various classification domains.

Experiments on the ORL database

The Olivetti-Oracle Research Lab (ORL) database consists of 40 subjects, and each subject has 10 different facial views representing various expressions, small occlusion (by glasses), different scale and orientations (Figure 5.3). Thus, the database contains totally 400 face images. The resolution of the images is 112×92. Note the ORL database has been used in many previous works such as[Lawrence et al., 1997, Li and Lu, 1999] (a face recognition scheme that combines the Self-organizing Map with convolutional neural networks) and [Turk and Pentland, 1991].

From the ORL database, we randomly selected a small number (3 or 5) of faces out of 10 for each subject to set up a GWAM model, and then counted the recognition accuracy on the remaining faces. Because there are only a few of training examples available, the deformation variances of the faces are difficult to capture. One efficient approach for tackling the issue is to augment the training set with some synthetically-generated face images. In this experiment, we synthesized images by some simple geometrical

Figure 5.3: Examples from the ORL database. Here shown 4 persons, each with two face images.

transformations, particularly rotation and scaling. Such an approach has also been used in some previous face recognition studies, and generally improves performance. In the present work, we generated 10 synthetic images from each raw training image by making small, random perturbations to the original image: rotation (up to +5 and -5) and scaling (by a factor between 95% and 105%). The final recognition results are listed in Table 5.4.

Table 5.1: Comparative recognition accuracy on ORL database

n	Eigenfaces	AA-MLP	SOM+CN	KAA-1	KAA-2
3	81.8	86.5	88.2	89.3	90.4
5	89.5	92.0	96.5	94.5	95.5

The results show that kernel autoassociators outperformed Eigenfaces and AA-MLP techniques, while achieved comparable performance to SOM+CN.

5.3 Spatial-Frequency Feature Learning and Face Recognition

The last section has introduced kernel autoassociators to directly classifying face images, without exploiting the special domain knowledge about faces. It is known that domain knowledge about face images is critical for face processing systems. Hence, in this section we present a novel autoassociator model to take advantages of both domain knowledge and the kernel autoassociation model. The domain knowledge here used is that an individual face has a certain configuration of local and global image features such that we can develop a set of special image kernels to represent them.

5.3.1 Subject Dependent Gabor Wavelet Networks

Wavelet networks were first introduced in [Zhang and Benveniste, 1992] as a combination of feed-forward neural networks and the continuous wavelet decomposition. The principle of a wavelet network consists in choosing a set of wavelets as activation functions for the second layer, adaptively according to a specific function f to be represented, such that an approximation $\hat{f}$ would be a linear combination of this wavelet set. Let the raw function be $f(x)$ and the wavelets be $\{\psi(a_i \cdot x + b_i)\}$, the approximation $\hat{f}(x)$ is given by

$$\hat{f}(x) = \sum_{i=1}^{M} w_i \psi(a_i \cdot x + b_i) \tag{5.1}$$

And the objective of tuning the network is to minimize the approximation error

$$\epsilon = \left\| f - \hat{f} \right\|^2 \tag{5.2}$$

where w_i is a weight and ψ the wavelet function, and a_i, b_i are the parameters in terms of dilation and translation of the function. M is the number of wavelets being used, defining the dimension of the second layer of the network.

Krueger extended the Wavelet Networks to 2D image representation by using Gabor functions [Krueger, 2001]. A Gabor wavelet network (GWN) thus adopts the following form

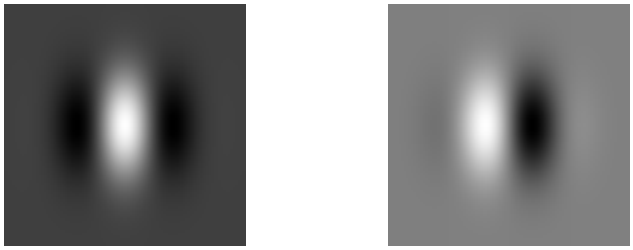
$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{M} w_i \psi_i(\mathbf{x}) \tag{5.3}$$

Here ψ_i, a particular 2D Gabor function, is defined by

$$\psi_i(\mathbf{x}) = \frac{k_i^2}{\sigma_i^2} \exp(-k_i^2 \Theta_i^T (\mathbf{x} - \mathbf{x}_{i0}) / 2\sigma_i^2) \exp(-j \cdot k_i \Theta_i^T (\mathbf{x} - \mathbf{x}_{i0})) \tag{5.4}$$

where k_i,Θ_i,σ_i,$\mathbf{x}_{i0}$ define the frequency, the phase, the spatial bandwidth and the centre of a Gabor kernel, respectively.

The Gabor wavelet Eq. 5.4 is a complex function, therefore has a real part and a imaginary part. The two parts differ by $\pi/2$ in phase. An example is given in Figure 5.4 that shows the two parts emphasize different types of image features. In particular, the real part tends to capture the features with relatively smooth structure at the center. On the other hand, the imaginary part would capture the transitional features at the center. In other words, the real part is suited for ridge but the imaginary part is suited for edge.



(a) Real Part (b) Imaginary Part

Figure 5.4: Real and Imaginary Parts of a Gabor kernel

It must be pointed out that the choice of values for σ and Θ may also considerably influence the wavelet image.

Because real world images are usually of real value, we may need to choose between the real and imaginary part of the kernel to represent the image. In [Krueger, 2001], Krueger suggests to manually select the desired part in accordance with the given image features. For example, he claims that if the edge of the object is of more importance than smooth features, we should choose imaginary parts of Gabor kernels because they are suitable for edge representation. However, manually selection method is neither convenient nor suitable because many objects would better be identified by both edge and ridge features.

The present work introduces an alternative, simple way to solve the problem. Instead of choosing between imaginary and real parts, we just add in a phase-shift variable ρ_s to the real part, yielding

$$\psi_i(\mathbf{x}) = \frac{k_i^2}{\sigma_i^2}\exp(-k_i^2\Theta_i^T(\mathbf{x}-\mathbf{x}_{i0})/2\sigma_i^2)\cos(k_i\Theta_i^T(\mathbf{x}-\mathbf{x}_{i0})+\rho_s) \tag{5.5}$$

With ρ_s being 0, the result will be equivalent to the real part of an kernel; with ρ_s being $\pi/2$, the function will be identical to the imaginary part. When the value ρ_s varies, the image of the kernel function Eq. 5.5 would reflect the change, and such an example is given in Figure 5.5.

A GWN is optimized for the representation of a single static image. But in face recognition, we have to take into account the variations in each subject. In other words, what GWN captures may be more likely to be transient features in a particular face image than essential and consistent features that identify the subject.

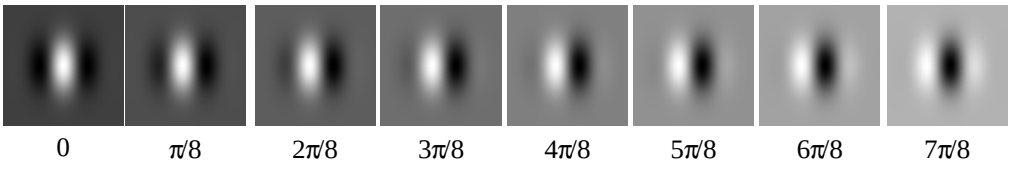


Figure 5.5: A Gabor kernel with shifting phase

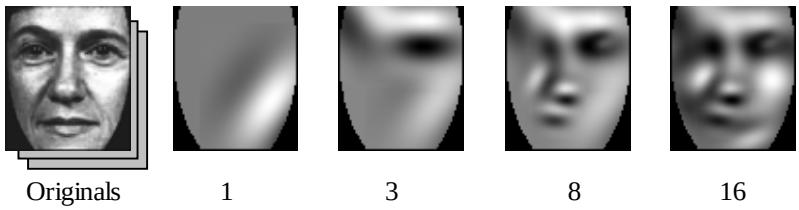


Figure 5.6: Progressive representation of faces with Gabor wavelets. *(Original images from FERET2 [Phillips et al., 1998]). The leftmost column gives a set of training samples; to the right are representations of a image by SDGWN with 1,3,8 and 16 Gabor wavelets, respectively.*

To overcome this drawback, we extend the GWN model to a subject dependent Gabor wavelet network (SDGWN) model. For a given subject and its image set $\{f_j\}$, a SDGWN aims to approximate the images by a common set of Gabor wavelets. (It must be pointed out that the common wavelet set differs from subject to subject.)

$$f_j(\mathbf{x}) = \sum_{i=1}^{M} w_{ji}\psi_i, j = 1 \cdots N \tag{5.6}$$

And the objective of network training is

$$E = \sum_j \left\| f_j - \sum_{i=1}^{M} w_{ji}\psi_i \right\|^2 = \sum_j \{ \int \left[f_j(\mathbf{x}) - \sum_{i=1}^{M} w_{ji}\psi_i(\mathbf{x}) \right]^2 d\mathbf{x} \} \tag{5.7}$$

Here $\{\psi_i\}$ is the common set of Gabor wavelets, $W_j = \{w_{j1} \ldots w_{jM}\}$ a weight vector specific for an image f_j, and can be considered as the projection of f_j into the subspace spanned by $\{\psi_i\}$.

Subject dependent Gabor wavelet network (Eq. 5.6) describes the variable appearances of a given subject rather than a single image. With a common set of bases, it is possible to grasp the concept of a subject by studying its examples in forms of weight vectors $\{W_j\}$ in the common subspace spanned by $\{\psi_i\}$.

In Figure 5.6, we demonstrate how a SDGWN uses a common set of Gabor wavelets to progressively represent a subject. The wavelet set is specified to the images in the

Figure 5.7: Subject representation with different number of Gabor wavelets. *From left to right are shown 3 instances of a person represented by an identical set of Gabor wavelets. From top row to bottom row are shown the original faces, the approximations with 16 Gabor wavelets and the approximations with 80 Gabor wavelets, respectively.*

left column. Note that only the representation results for the first image are displayed. It is evident that particular wavelets may capture some local spatial and frequency features across the images. For instance, here the most significant (largest energy) Gabor wavelet captures the feature on the cheek; the subsequent wavelet captures the feature around eyebrow and forehead. It is also obvious that the approximation accuracy largely depends upon the number of Gabor wavelets in the representation network.

With the weight vector trained for different images, a SDGWN produces different approximations. Figure 5.7 gives such an example of SDGWN representation using a common set of wavelets. The top row shows three face images of a subject, for which a representation network was created. The next two rows draw the representations respectively with 16 Gabor wavelets and with 80 Gabor wavelets.

The configuration of Gabor wavelets $\{\psi_i\}$ is crucial for accurate representation. Hence, the optimization of Eq. (5.2) that determines the configuration plays a vital role in the system. In this work, we use Levenberg-Marquardt algorithm and multi-scale strategy that was proposed by Krueger [Krueger, 2001]. In particular, we construct two layers of Gabor functions in each representation network. The top layer is for capturing large scale features by using Gabor functions with larger spatial frequency k, while the

Figure 5.8: Comparative performance for representing a new face. *The leftmost one is an input image to be represented by different GWN/SDGWN models which have been trained in advance by a set of 3 images. 3 GWNs, each adapted to a gallery image, produced 3 individual representations shown successively to the right, while the rightmost one is the representation by a SDPGWN model that has been adapted to the whole training set.*

bottom layer is for refining and captures local, small-scale features. The optimization scheme offers efficient computation. For an image at size 150×130, it will cost approximately 20 seconds to optimize 16 top-level Gabor functions, and 1 minute or so for 64 bottom-level functions, by non-optimized Matlab codes running on a PentiumIII 733MHz system.

It should be pointed out that because Gabor wavelets are non-orthogonal, it is impossible to calculate a weight w_i by directly projecting an image f onto the Gabor wavelet $\{\psi_i\}$. In practice, we resort to linear algebra to find the weight vector that produces a minimal approximation error. Thus, the weight vector can be efficiently computed by

$$W_j = \tilde{\Psi} f_j \; with \; \tilde{\Psi} = \Psi^+ = (\Psi^T \Psi)^{-1} \Psi^T \tag{5.8}$$

Figure (5.8) shows a simple comparison between GWN and SDGWN for face representation. We used three samples for training and another image for test. A SDGWN model was adapted to the three images, and three GWN models were adapted to the three images respectively. Representations of the test image are drawn successively to the right of original one in the figure. It can be seen that the SDGWN demonstrates better capability, since it favorably describes the expression appearance and details more precise facial features.

Table 5.2: Performance of GWN and SDGWN as a Function of approximation accuracy for new images

	mean	var
GWN	0.843	0.028
SDGWN	0.893	0.018

SDGWN vs. GWN

To compare SDGWN with GWN on their approximation performance, we generated a test-bed – a randomly selected subset from the FERET database. The data set contains 117 images for 10 subjects, and 3 images per subject were randomly selected for training, while the remainder composes the testing set.

When a new pattern is presented, both SDGWNs and GWNs produce their representations. The representation efficiency is evaluated by measuring the difference between the output and the input pattern, according to the similarity measurement in Eq. (5.9). Note we used 80 Gabor wavelets for each SDGWN/GWN model; moreover, to take advantages of multiple GWNs for each subject, we only selected the best output from them. Table 5.2 shows the final results that suggest the SDGWN model produced clearly better representations.

5.3.2 The Gabor wavelet associative memory model

The SDGWM model offers an effective approach to face representation. As kernel autoassociators with Gaussian kernels have proven in the last chapter to be an efficient approach for concept learning and recognition, we use it as the basis to develop a Gabor wavelet associative memory (GWAM) (Fig. 5.9) which consists of five layers:

1. The first layer receives the input image, say f;

2. The second layer serves as an encoder that transforms the input pattern f into the particular SDGWN subspace and produces a weight vector $\mathbf{x}$ according to Eq. (5.8);

3. The third layer compares the pattern $\mathbf{x}$ (weight vector) with prototypes in kernels, and produces a kernel product vector k;

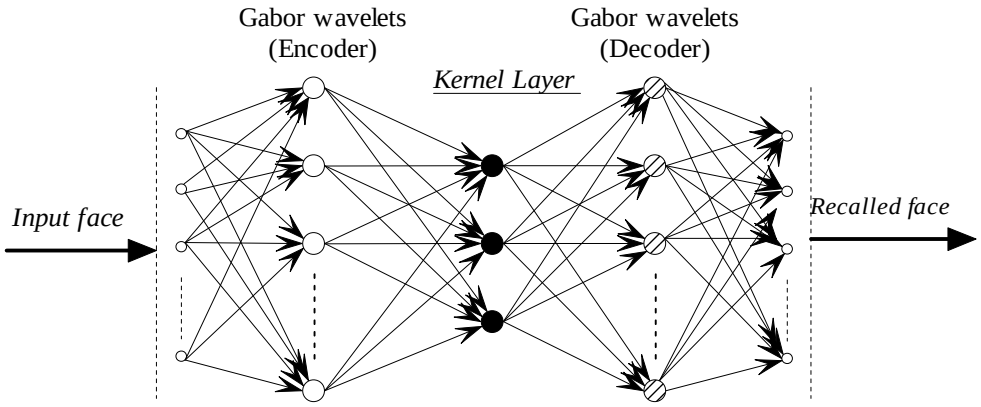


Figure 5.9: Architecture of Gabor wavelet associative memory. *Gabor wavelet associative memory is a combination of a Gabor wavelet network and a kernel associative memory. The 2nd layer and the 4th layer correspond to a particular set of Gabor wavelets, and are used to encode/decode the image patterns to/from the SDGWN subspace. A kernel autoassociator model is embedded as the center part between the encoder and the decoder layers.*

4. The fourth layer uses the vector $\mathbf{k}$ to recall a weight vector pattern $\hat{\mathbf{k}}$ from the kernel autoassociative memory, according to Eq. (4.12) (here we consider kernel autoassociators with linear backward mapping functions F_b);

5. The last layer produces the final approximation by decoding the weight vector $\hat{\mathbf{k}}$ into image domain, according to (Eq. (5.3));

Therefore, the GWAM model inherits the advantages of both SDGWN and kernel autoassociators in face representation and concept learning. Similar to the classification scheme in the last chapter, we set up a modular face recognition system (Fig. (5.10).

In this scheme, each subject is associated with a particular GWAM model. At the recognition stage, a probe image f is input to all GWAM models to produce respective reproductions. The similarity measurements between the probe image and recalled images are taken to determine which class the image f is from. Given the probe image f and a recalled image $\hat{f}$, we adopted a widely used similarity measurement given by

$$\cos(\hat{f}, f) = \frac{\hat{f}^T f}{\|\hat{f}\| \cdot \|f\|} \tag{5.9}$$

To better illustrate this scheme, in Figure 5.11 we show an example of the recognition process by GWAMs. The top image is the probe image to be recognized. It is encoded

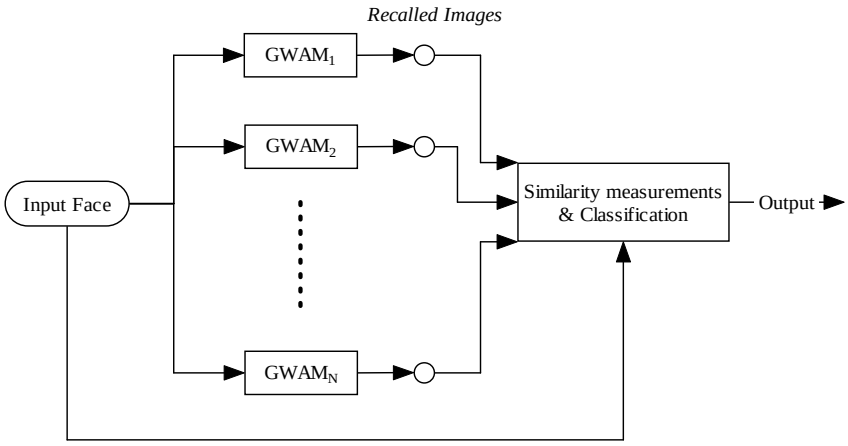


Figure 5.10: Face recognition scheme.
The probe image is first input to individual GWAMs to yield respective reconstructions, which are compared with original image by a similarity measurement. And the system classifies the image to the class with the maximal similarity.

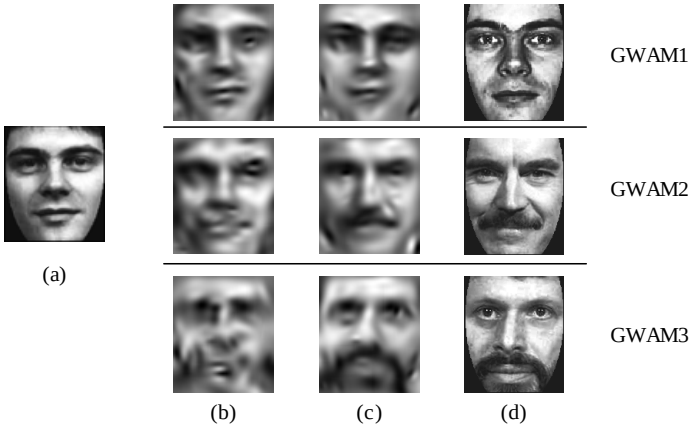


Figure 5.11: Illustration of face recognition process by GWAM.
(a) A probe image to be recognized; (b) SDGWN representations being used as the keys for the kernel autoassociators; (c) The reconstructions from recalled weight vectors by the autoassociators; (d) The typical faces of the three subjects. In this picture we show only the top three reproductions, with the best one (left) singled out.

by various GWAM models to produce different representations that are shown in the second row. The representations are in turn transferred, in forms of weight vectors, to kernel associative memories to recall weight vector patterns. The recalled weight vector patterns are then mapped backwards to image domain, producing the final reconstructions. In the third row we show only the top three reconstructions in terms of minimal reconstruction error. The bottom row denotes each subject with a typical image from the training gallery. It is evidence that the true subject's network tends to produce the best reproduction.

5.4 Performance of GWAM-based Face Recognition System

We have conducted extensive experiments to examine the proposed face recognition scheme and to compare it with other well-known methods, e.g. the Eigenfaces, SOM+CN [Lawrence et al., 1997], Line Edge Map [Gao and Leung, 2002], and ARENA [Sim et al., 1999], using some publicly available benchmark face databases, i.e. the FERET standard facial database (Release2) [Phillips et al., 1998], the Olivetti-Oracle Research Lab (ORL) database [Samaria and Harter, 1994] and the AR face database from Purdue University [Martinez and Benavente, 1998].

Experiments on FERET Database

FERET2, the second release of the FERET, consists of 14,051 8-bit gray-scale face images with views ranging from frontal to left and right profiles. The database design took into account various image variations from different expressions, different hairstyles and different illuminations.

FERET2 provides explicit coordinate information for 3816 images. From these images, we selected the subjects with more than 5 frontal or near-frontal (Pose angle $\leq 15^{\circ}$) images. Finally, we have a dataset of 119 persons with 927 images, which were pre-processed by a normalization program based on the given eye positions, yielding images at 130×150 pixels (see Figure 5.12).

We carried out a few experiments with the data set. In each experiment, the training set P was selected from the whole data set, and the remainder constituted the testing

Figure 5.12: Samples from FERET face database.
Top row: samples from FERET database. Bottom row: the corresponding normalized images.

set G. When the number of samples per subject is $m = 3$, there were totally 357 images for training and 570 images for testing; when $m = 4$, there were 476 training images and 451 testing images.

Here we compare the proposed scheme with a variant of Eigenfaces, called PCA-nearest-neighbour [Sim et al., 1999]. Basic Eigenfaces compute the centroid of weight vectors for each person in the training set, by assuming that each person's face images will be clustered in the eigenface space. While in PCA-nearest-neighbor, every weight vector is stored for richer representation. When a probe image is presented, it first transforms into the eigen-space and the weight vector will be compared with memorized patterns, then a nearest-neighbor (NN) algorithm will be employed to locate the closest pattern class. From the face data set, we first computed the covariance and then chose the first n eigenvectors to construct an eigen-space. We tried a few n from 20 to 40 but did not perceive remarkable variance in the performance. Without otherwise specified, $n = 25$ in the following.

Another face recognition system under comparison is a recently proposed simple nearest-neighbor based template matching algorithm, termed ARENA [Sim et al., 1999], which employs reduced-resolution images and a simple similarity measure defined as

$$L_0^*(\vec{x} - \vec{y}) = \sum_{|x_i - y_i| > \delta}^{n} 1 \tag{5.10}$$

where δ is a user–defined constant for which we took $\delta = 10$. Similar to the above PCA

technique, every training pattern was memorized. The distance from the query image to each of the stored images in the database is computed and the label of the best match is returned.

The experiment also investigates and compares the performance of another face recognition system [Zhang et al., 2004a] that combines discrete wavelet transform (DWT) features and kernel associative memory (with linear backward mapping functions F_b). The system is referred to as KAA-WT hereafter.

To evaluate our proposed GWAM system using the FERET dataset, we first set up one GWAM model for each subject, which is determined by the Gabor set $\{\psi_n\}$, sample weights $\{\mathbf{x}_n\}$, kernel autoassociator weight matrix W and variance σ. When a probe image is given at test stage, a GWAM system recognizes the face by picking up the optimal reconstruction.

Table 5.3: Recognition accuracy for FERET dataset

n	PCA	ARENA	KAA-WT	GWAM	
				M=80	M=16
3	54.3	55	84.7	99.3	95.8
4	55.2	55.2	91.6	99.6	99.1

As elaborated in Section 1, the number of Gabor wavelets, i.e. the GWAM' dimension M, has remarkable influence on the reconstruction precision. To examine its influence on the system performance, we conducted experiments with M =80 and 16 respectively. The experimental results are summarized in Table 5.3. Obviously, GWAM dramatically outperformed other systems.

We adopted an evaluation methodology proposed by the developers of FERET [Phillips et al., 1998]. In this evaluation, the recognition system will answer a question like "is the correct answer in the top n matches?" rather than "is the top match correct?". The performance statistics are reported as cumulative match scores. In this case, an identification is regarded as correct if the true object is in the top n matches. For example, if $n = 5$ and 80 identifications out of 100 satisfy the condition (have their true identities in top 5 matches respectively), the cumulative match score for R_5 is $80/100 = 0.8$.

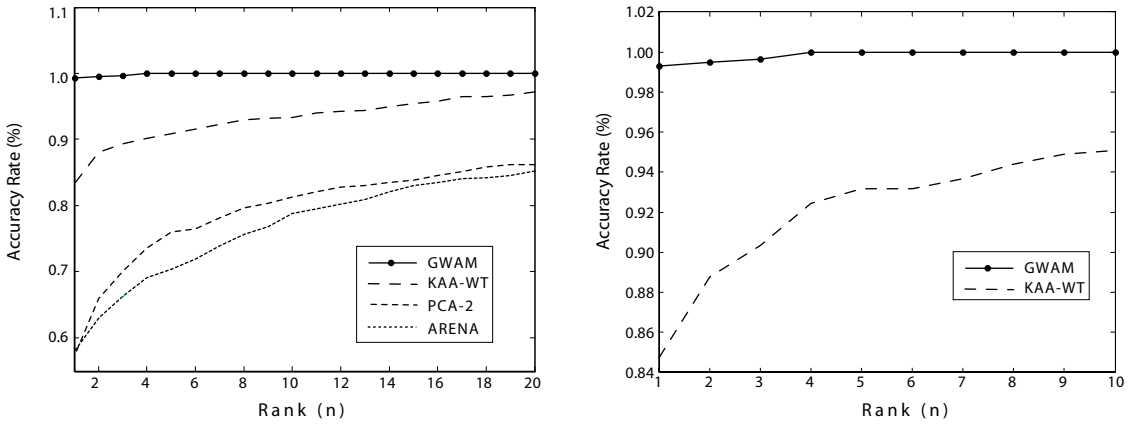


Figure 5.13: Comparison of accumulated accuracy on FERET.
The left one illustrates accumulated accuracy from GWAM, KAA-WT, PCA-2, ARENA respectively. The right details the left figure in a particular region for comparing GWAM and KAM models.

Figure 5.13 illustrates the cumulative match scores with different algorithms. The rank is plotted along the horizontal axis, and the vertical axis denotes the percentage of correct matches. Again, GWAM exhibits clear superiority over other methods. It is also interesting to notice that with GWAM, the true class of an image consistently lies in top 4 matches, i.e. $R_4 = 100\%$.

Experiments on ORL database

ORL database contains 40 subjects with 10 images per subject. The images are acquired under variable lighting condition, facial expressions and viewpoint. The image resolution is 92×112. Based on this database, we are able to compare our system with some others, i.e. [Lawrence et al., 1997, Li and Lu, 1999].

Some examples of ORL face images have been given in Figure 5.3 earlier. From the ORL database, we randomly selected a limited number (3 or 5) of faces out of 10 for each subject to set up a GWAM model, and then counted the recognition accuracy on the remaining faces. The results are given in Table 5.4 where the GWAM system (with 16 Garbor wavelets in each network) again shows superior performance.

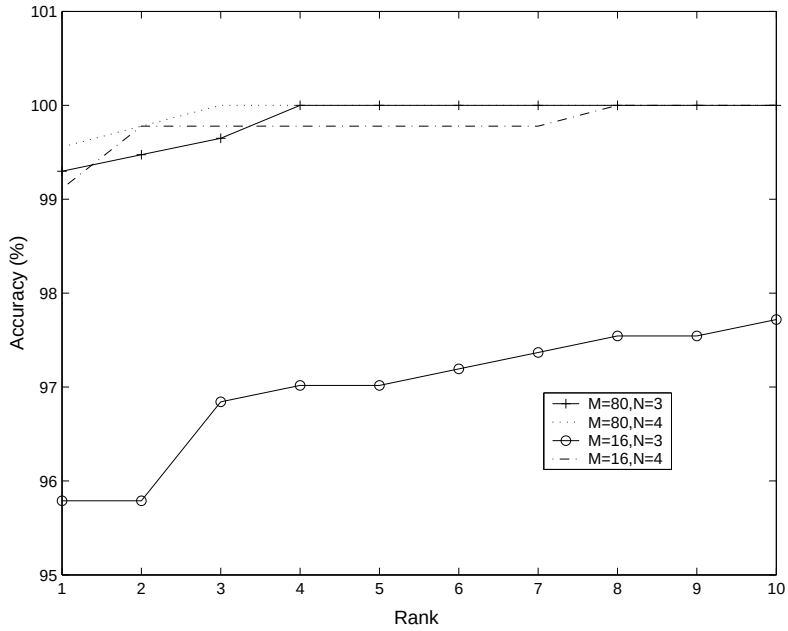


Figure 5.14: Accumulated accuracy on FERET by GWAM.
M: the number of Gabor wavelets, N: the number of training samples for each subject.

Table 5.4: Recognition accuracy for the ORL database

n	PCA	SOM+CN	ARENA	KAA-WT	GWAM	
					M=80	M=9
3	81.8	88.2	92.2	94.3	100	97.9
5	89.5	96.5	97.1	98.2	100	98.8

Experiments on AR face database

The AR face database from Purdue University consists of over 3000 color images of the frontal view faces of 126 people, with roughly 26 different images per person, recorded in two different sessions separated by two weeks and each session consisting of 13 images per person[Martinez and Benavente, 1998]. We randomly selected 42 males and 44 females to create the experimental dataset. We pre-processed all the images by a normalization process by first turning them into gray scale images, followed by an image normalization based on eye positions.

AR face images show dramatically varying lighting conditions. Though theoretical analysis suggested that a function invariant to illumination does not exist in general case [Moses and Ullman, 1992], an object representation robust to lighting variations is

Figure 5.15: Samples from AR face database.
Sample images of training (2 leftmost) and test faces (under varying lighting conditions). The top row shows the original images from one session. The bottom row shows the normalized images from both sessions.

Table 5.5: Recognition accuracy for AR database

L. Cond.	Eigenface	Edge map	LEM	GWAM
Left .	26.8%	82.1%	92.9%	96.5%
Right .	49.1%	73.2%	91.1%	90.9%
Both .	64.3%	54.5%	74.1%	89.0%

still critical in real world applications. Here we would like to compare our method (with 80 Gabor wavelets for each GWAM model) with some recent techniques such as the Line Edge Map (LEM) [Gao and Leung, 2002]. LEM uses polygonal line segments of edges to represent a face, and then a kind of Hausdorff distance measure was proposed to compare the LEMs for recognition.

In this experiment, we selected 2 normal views from each person to set up the training set, while the testing set includes other 6 images, i.e. 2 "Left light on", 2 "Right light on" and 2 "Both lights on" (Fig.5.15).

Experimental results are summarized in Table 5.5, in which the performances of other techniques were duplicated from [Gao and Leung, 2002]).

Remarks:

1. The GWAM significantly outperforms other techniques in general situations. But in "right light on" case it achieved a similar rate to that of LEM.

2. Our system shows robust performance under varying lighting conditions. By simple calculation, the maximum variation of the accuracy is only 7%, comparing to 18.8% with LEM, 27.6% with Edge Map and 37.5% with Eigenfaces.

3. The performance of LEM or "Edge map" degrades by "worse" illumination conditions, while the Eigenfaces improves. This suggests that "single side" illumination

has some adverse effects on the methods using "holistic texture" information than on those using "edge information". On the other hand, our GWAM system seems to inherit both advantages of "edge method" and "texture method".

4. Apparently, the recognition rates with left light on are higher than that with right light on. This is because the illumination from the right side is generally stronger than that from the left side, as has been pointed out in [Gao and Leung, 2002].

5.5 Summary

In this chapter we have applied kernel autoassociators to the challenging problem of face recognition. We tentatively examined the direct application of kernel autoassociator models to recognizing face images. More importantly, we proposed a high performance face recognition system by taking advantages of both Gabor wavelet networks for face representation and kernel autoassociators for concept learning and recognition. Here Gabor wavelet networks employ the domain knowledge about face images that an individual face has a certain configuration of local and global image features such that we can develop a set of special image kernels to represent them.

We have carried out extensive experiments to evaluate the GWAM-based face recognition scheme, in comparison with other state-of-the-art face recognition systems. Our scheme provided excellent performance on three popular databases, namely, the FERET (Release 2), the ORL and the AR face database. The results suggest that we have successfully incorporated domain knowledge about face images into kernel autoassociators to develop a high performance face recognition system.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

The fundamental objective of the thesis is to develop efficient view-based models for determining the states and the identities of (moving and transforming) known objects in images.

We developed a tracking method on the basis of a kernel-based model for representing objects under affine transformation. The model allows a robust, spatially-and-spectrally smooth similarity measure with respect to affine transformation parameters. The method can distinguish similar objects in cluttered scenes; and it also leads to better motion understanding than non-posture-estimation methods. It has demonstrated excellent performance for determining the affine transformation of visual objects in various synthetic and real-world tasks. In addition, since it depends on merely one sample image, it is quite easy to implement and is suited to many real tracking tasks that have only few samples for each target object.

For generic object classification, the thesis presented a learning and classification model called kernel autoassociators. The model takes advantage of kernel feature space to learn the nonlinear dependencies among multiple samples. It is more accessible and easier to implement than conventional autoassociative networks, while providing better performance. The model has demonstrated excellent performance in novelty detection and multi-class classification. For a special type of object recognition – face recognition, a Gabor wavelet associative memory model was presented that combines Gabor wavelet networks for face representation and kernel autoassociators for nonlinearity learning.

It has been shown that the model can dramatically improve the capability of kernel autoassociators in learning faces, yielding a high-performance face recognition system.

In conclusion, the thesis has proposed a few efficient view-based models that can be applied to visual tracking and recognition.

6.2 Future Work

The theme throughout the thesis was the use of computational approaches to model one or multiple images of an object, for either tracking or recognition. It is interesting to expect the future extension of these approaches in the field of computer vision and pattern recognition. In particular, we may expect improvements in at least three areas.

Multi-Sample Spatial-Spectral Representation. The current tracking method depends on a given sample image represented by the kernel-based spatial-spectral model. However, in some cases a single image may not be sufficient for reference, especially when the target object has largely different images resulting from changing views. As many tracking tasks may offer multiple samples for each target object, it is promising to extend the representation model as well as the tracking method by taking advantage of multiple samples. We suggest that statistical representation models based on the samples should be so considered as to account for more complex image transformations induced by e.g. non-ridge 3D object deformations.

Filtering Mechanism. Visual tracking can be viewed as a fusion process that combines the dynamics of target objects and the observation. This thesis emphasizes not on dynamics but on observation models and the related object-searching method. As the presented method was essentially a searching mechanism that seeks a target object around a predicted position/state, it is possible to incorporate a special particle filtering scheme to enhance the method. Such a scheme will bring two potential benefits: it will improve the efficiency of particle-filtering-based tracking by particle-optimization (the importance of the optimization can be seen in [Deutscher et al., 2000][Zhang et al., 2004e]); it will increase the robustness of the current tracking method especially when the target objects are subject to fast, non-continuous movements.

Discriminant Autoassociation Learning. An autoassociator is essentially used to model a particular class by describing the dependency among the patterns. Thus, it avoids directly analyzing the difference between classes, and mostly caters to applications like one-class learning such as novelty detection in which few or no examples of the negative class are available. What follows is an interesting question: is it possible to better account for discriminating features in training autoassociators. In fact, the current mechanism of autoassociators does not necessarily preserve all the essential discriminating information. Therefore, we may in future work consider how to design a special autoassociation scheme which not only learns the dependencies among the patterns in the positive class, but also preserves the important features that distinguish it from other classes. We refer to this problem as discriminant autoassociation learning.

In any case, it appears that the proposed tracking method, the nonlinear autoassociation model and the face recognition scheme will continue to serve as important bases of more advanced and sophisticated algorithms that can make computers better see and understand the visual world.

List of Publications

Journal

1. Haihong Zhang, Bailing Zhang, Weimin Huang, Qi Tian, Gabor Wavelet Associative Memory for Face Recognition. Vol 16, No.1, Jan 2005, pp.275-278.

2. Bailing Zhang, Haihong Zhang, Shuzhi Sam Ge, Face Recognition by Applying Wavelet Subband Representation and Kernel Associative Memory. IEEE Transactions on Neural Networks, 1(15), 166-177, Jan. 2004.

3. Haihong Zhang, Weimin Huang, Zhiyong Huang, Bailing Zhang, A Kernel Autoassociator Approach to Pattern Recognition, to appear in IEEE Transactions on Systems, Man and Cybernetics - Part B, 2005

Conference

1. Haihong Zhang, Weimin Huang, Zhiyong Huang and Liyuan Li, Affine Object Tracking with Kernel-based Spatial-Color Representation, International conference on Computer Vision and Pattern Recognition 2005, June 20-26.

2. Haihong Zhang, Weimin Huang, Zhiyong Huang and Liyuan Li, Kernel-Based Method for Tracking Objects with Rotation and Translation, the 17th International Conference on Pattern Recognition, Volume 2, 2004, Page(s):728 - 731

3. Haihong Zhang, Weimin Huang, Zhiyong Huang and Bailing Zhang, Kernel Autoassociator with Applications to Visual Classification, the 17th International Conference on Pattern Recognition, Volume 2, 2004, Page(s):443 - 446

4. Haihong Zhang, Yan Guo, Facial Expression Recognition using Continuous Dynamic Programming, IEEE ICCV Workshop on Recognition, Analysis, and Track-

ing of Faces and Gestures in Real-Time Systems, July, 2001, pp. 163-167.

5. Haihong Zhang, Yan Guo, Personalized Facial Expression Recognition for Next Generation Man-Machine Interaction, the 3th International Converence on Information, Communications, and Signal Processing, Oct. 2001, Singapore.

6. Yan Guo, Haihong Zhang, Avatar Control by Facial Expressions, the 10th IEEE International Conference on Fuzzy Systems ,Volume: 3 , 2-5 Dec. 2001, pp. 1351 - 1354

7. Haihong Zhang, Weiming Huang, Zhiyong Huang, Bailing Zhang, A Particle Filtering Framework for Visual Tracking using Indirect Measurements, accepted by the International Conference on Control, Automation, Robotics and Vision, 2004.

Bibliography

[Aeberhard et al., 1992] Aeberhard, S., Coomans, D., and de Vel, O. (1992). Comparison of classifiers in high dimensional settings. Technical Report 92-02, Dept. of Computer Science and Dept. of Mathematics and Statistics, James Cook University of North Queensland, Australia.

[Aggarwal and Cai, 1999] Aggarwal, J. K. and Cai, Q. (1999). Human motion analysis: A review. *Computer Vision and Image Understanding*, 73(3):428–440.

[Aloimonos and Rosenfeld, 1991] Aloimonos, Y. and Rosenfeld, A. (1991). Computer vision. *Science*, 253(5025):1249–1254.

[Aronszajn, 1950] Aronszajn, N. (1950). Theory of reproducing kernels. *Transaction of American Mathematical Society, volume=*.

[Avidan, 2004] Avidan, S. (2004). Support vector tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8).

[Baldi and Hornik, 1989] Baldi, P. and Hornik, K. (1989). 'neural networks and principal component analysis: learning from examples without local minima. *Neural Networks*, 2:53–58.

[Bar-Shalom and Fortmann, 1988] Bar-Shalom, Y. and Fortmann, T. (1988). *Tracking and Data Association*. Academic Press.

[Bartlett and Sejnowski, 1997] Bartlett, M. and Sejnowski, T. (1997). Independent components of face images: A representation for face recognition. In *Proc. the 4th Annual Joint Symposium on Neural Computation*.

[Bascle and Deriche, 1995] Bascle, B. and Deriche, R. (1995). Region tracking through image sequences. In *IEEE International Conference on Computer Vision*, pages 302–307.

[Baudat and Anouar, 2000] Baudat, G. and Anouar, F. (2000). Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12.

[Beck et al., 1983] Beck, J., Prazdny, K., and Rosenfeld, A. (1983). A theory of textural segmentation. In Beck, J., Hope, B., and Rosenfeld, A., editors, *Human and Machine Vision*, pages 1–38. Academic Press, New York.

[Birchfield, 1998] Birchfield, S. (1998). Elliptical head tracking using intensity gradients and color histograms. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 232–237.

[Blake and Isard, 1998] Blake, A. and Isard, M. (1998). *Active Contours*. Springer-Verlag, London.

[Boykov and Huttenlocher, 2000] Boykov, Y. and Huttenlocher, D. (2000). Adaptive bayesian recognition in tracking rigid objects. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 697–704.

[Bressan et al., 2003] Bressan, M., Guillamet, D., and Vitria, J. (2003). Using an ica representation of local color histograms for object recognition. *Pattern Recognition*, 36(3):691–701.

[Cascia et al., 2000] Cascia, M. L., Sclaroff, S., and Athitsos, V. (2000). Fast, reliable head tracking under varying illumination: An approach based on registration of texture-mapped 3d models. 22(4):322–336.

[Chellappa et al., 1995] Chellappa, R., Wilson, C., and Sirohey, S. (1995). Human and machine recognition of faces: A survey. *Proceedings of the IEEE*, 83(5):705–740.

[Cheng, 1995] Cheng, Y. (1995). Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:790–799.

[Cho et al., 2001] Cho, K., Jang, J., and Hong, K. (2001). Adaptive skin-color filter. *Pattern Recognition*, 34(5):1067–1073.

[Collins and Liu, 2003] Collins, R. and Liu, Y. (2003). On-line selection of discriminative tracking features. In *IEEE International Conference on Computer Vision*, volume 1, pages 346–C352.

[Collins, 2003] Collins, R. T. (2003). Mean-shift blob tracking through scale space. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Collins et al., 2000] Collins, R. T., Lipton, A. J., and Kanade, T. (2000). Introduction to the special section on video surveillance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:745–746.

[Comaniciu et al., 2003] Comaniciu, D., Ramesh, V., and Meer, P. (2003). Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577.

[Comanicui et al., 2000] Comanicui, D., Ramesh, V., and Peer, P. (2000). Real-time tracking of non-rigid objects using mean shift. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 142–149.

[Comon, 1994a] Comon, P. (1994a). Independent component analysis - a new concept? *Signal Processing*, 36:287–C314.

[Comon, 1994b] Comon, P. (1994b). Independent component analysis - a new concept? *Signal Processing*, 36:287–314.

[Cootes et al., 2001] Cootes, T., Edwards, G., and Taylor, C. (2001). Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685.

[Cootes et al., 1993] Cootes, T., Taylor, C., Lanitis, A., Cooper, D., and Graham, J. (1993). Building and using flexible models incorporating grey-level information. In *IEEE International Conference on Computer Vision*, pages 242–246.

[Cottrell et al., 1987] Cottrell, G. W., Munro, P., and Zipser, D. (1987). Learning internal representations of gray scale images: An example of extensional programming. In *Proc. 9th Annu. Cognitive Sci. Soc. Conf.*, pages 462–473.

[Cover, 1965] Cover, T. M. (1965). Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, EC-14:326–334.

[Dai and Nakano, 1996] Dai, Y. and Nakano, Y. (1996). Face-texture model based on sgld and its application. *Pattern Recognition*, 29:1007–1017.

[Daugman, 1988] Daugman, J. (1988). Complete discrete 2d gabor transform by neural networks for image analysis and com-pression. *IEEE Trans. Acoustics, Speech, and Signal Processing*, 36:1169–1179.

[Daunicht, 1991] Daunicht, W. J. (1991). Autoassociation and novelty detection by neuromechanics. *Science*, 253(5025):1289–1291.

[DeMers and Cottrel, 1993] DeMers, D. and Cottrel, G. W. (1993). Nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems*, pages 580–587.

[Deutscher et al., 2000] Deutscher, J., Blake, A., and Reid, I. (2000). Articulated body motion capture by annealed particle filtering. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 126–133.

[Devroye and Lugosi, 2000] Devroye, L. and Lugosi, G. (2000). Variable kernel estimates: On the impossibility of tunning the parameters. In Gine, E., Mason, D., and Wellner, J., editors, *High-Dimensional Probability II*, pages 405–424. Springer, New York.

[Dimitrijevic et al., 2004] Dimitrijevic, M., Ilic, S., and Fua, P. (2004). Accurate face models from uncalibrated and ill-lit video sequences. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Donato et al., 1999] Donato, G., Bartlett, M., Hager, J., Ekman, P., and Sejnowski, T. (1999). Classifying facial actions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):974–989.

[Edelman, 1999] Edelman, S. (1999). *Representation and Recognition in Vision*. MIT Press.

[Elgammal et al., 2001] Elgammal, A., Duraiswami, R., and Davis, L. (2001). Efficient nonparametric adaptive color modeling using fast gauss transform. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 563–570.

[Elgammal et al., 2003a] Elgammal, A., Duraiswami, R., and Davis, L. (2003a). Efficient kernel density estimation using the fast gauss transform with applications to color modeling and tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(11):1499–1504.

[Elgammal et al., 2003b] Elgammal, A., Duraiswami, R., and Davis, L. (2003b). Probabilistic tracking in joint feature-spatial spaces. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Féraud et al., 2001] Féraud, R., Bernier, O. J., Viallet, J. E., and Collobert, M. (2001). A fast and accurate face detector based on neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:42–53.

[Ferrari et al., 2001] Ferrari, V., Tuytelaars, T., and van Gool, L. (2001). Real-time affine region tracking and coplanar grouping. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 226–233.

[Funt and Finlayson, 1995] Funt, B. and Finlayson, G. (1995). Color constant color indexing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:522–529.

[Gabor, 1946] Gabor, D. (1946). Theory of communications. *J. Inst. of Electronical Engineering*, 93:429–557.

[Gao and Leung, 2002] Gao, Y. and Leung, K. (2002). Face recognition using line edge map. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(6):764–778.

[Girolami and He, 2003] Girolami, M. and He, C. (2003). Probability density estimation from optimally condensed data samples. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25.

[Graham and Allinson, 1998] Graham, D. B. and Allinson, N. M. (1998). Characterizing virtual eigensignatures for general purpose face recognition. In *Face Recognition: From Theory to Applications*, pages 446–456. NATO ASI Series F, Computer and Systems Sciences.

[Grimson, 1990] Grimson, E. (1990). *Object Recognition by Computer*. MIT Press, Cambride, MA.

[Hadjidemetriou et al., 2001] Hadjidemetriou, E., Grossberg, M., and Nayar, S. K. (2001). Histogram preserving image transformations. *International Journal of Computer Vision*, 45:5–23.

[Hager and Belhumeur, 1996] Hager, G. and Belhumeur, P. (1996). Real-time tracking of image regions with changes in geometry and illumination. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 403–410.

[Hanson and Gluck, 2000] Hanson, N. J. S. J. and Gluck, M. A. (2000). Nonlinear autoassociation is not equivalent to pca. *Neural Computation*, 12.

[Hanson and Kegl, 1987] Hanson, S. J. and Kegl, J. (1987). Parsnip: a connectionist network that learns natural language grammar from exposure to natural language sentences. In *Proc. the Ninth Annual Conference on Cognitive Science*, pages 106–119.

[Haykin, 1999] Haykin, S. (1999). *An Introduction to Neural Networks – A Comprehensive Foundation, 2nd Edition*. New Jersey, Prentice-Hall.

[Hertz et al., 1991] Hertz, J., Krogh, A., and Palmer, R. G. (1991). *Introduction to the Theory of Neural Computation*. Addison-Wesley, Redwodd City, CA.

[Hubel and Wiesel, 1994] Hubel, D. and Wiesel, T. (1994). Brain mechanisms of vision. In Madan M. Gupta, G. K. K., editor, *Neuro-vision systems : principles and applications*, pages 167–173. New York, IEEE Press.

[Isard and Blake, 1996] Isard, M. and Blake, A. (1996). Visual tracking by stochastic propagation of conditional density. In *European Conference of Computer Vision*, pages 343–356.

[James, 1950] James, W. (1950). *The princples of psychology*. New York: Henry Holt &Co.

[Japkowicz et al., 1995] Japkowicz, N., Mayers, C., and Gluck, M. A. (1995). A novelty detection approach to classification. In *Proc. the Fourteenth Joint Conf. Artificial Intelligence*, pages 518–523.

[Juwei et al., 2003] Juwei, L., Plataniotis, K., and Venetsanopoulos, A. (2003). Face recognition using kernel direct discriminant analysis algorithms. *IEEE Transaction on Neural Networks*, 14(1):117–126.

[Kanada, 1973] Kanada, T. (1973). Picutre processing by computer complex and recognition of human faces. Technical report, Kyoto University, Department of Information Science.

[Keysers et al., 2000] Keysers, D., Dahmen, J., Theiner, T., and Ney, H. (2000). Experiments with an extended tangent distance. In *International Conference on Pattern Recognition*, volume 2, pages 38–42.

[Kohonen, 1977] Kohonen, T. (1977). *Associative Memory: A system theoretic approach*. Springer-Verlag, Berlin.

[Kohonen, 1980] Kohonen, T. (1980). *Content-Addressable Memories*. Springer, Berlin.

[Kramer, 1991] Kramer, K. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *Journal of AIChE*, 37:233–243.

[Krueger, 2001] Krueger, V. (2001). *Gabor Wavelet Networks for Object Representation.* PhD thesis, Christian-Albrecht University, Germany.

[Landy, 1996] Landy, M. (1996). Texture perception. In Adelman, G., editor, *Encyclopedia of Neuroscience*. Amsterdam: Elsevier.

[Lawrence et al., 1997] Lawrence, S., Giles, C. L., Tsoi, A., and Back, A. (1997). Face recognition: A convolutional neural network approach. *IEEE Transaction on Neural Networks*, 8:98–113.

[LeCun et al., 1999] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. J. (1999). Backpropatation applied to andwritten zip code recognition. *Neural Computation*, 1:541–551.

[Lee et al., 1996] Lee, C. H., Kim, J. S., and Park, K. H. (1996). Automatic human face location in a complex background. *Pattern Recognition*, 29:1877–1889.

[Lee et al., 2003] Lee, J., Lee, W., and Jeong, D. (2003). Object tracking method using back-projection of multiple color histogram models. In *International Symposium on Circuits and Systems*, volume 2, pages 668–671.

[Li and Lu, 1999] Li, S. and Lu, J. (1999). Face recognition using the nearest feature line method. *IEEE Trans. Neural Networks*, 10:439–443.

[Lin et al., 1997] Lin, S., Kung, S., and Lin, L. (1997). Face recognition/detection by probabilistic decision-based neural network. *IEEE Trans. Neural Networks*, 8:114–132.

[Liu and Wechsler, 1999] Liu, C. and Wechsler, H. (1999). Comparative assessment of independent component analysis (ica) for face recognition. In *The 2nd Int. Conf. on Audio- and Video-based Biometric Person Authentication.*

[Liu and Wechsler, 2002] Liu, C. and Wechsler, H. (2002). Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition. *IEEE Trans. Image Processing*, 11(4):467–576.

[Lowitz, 1983] Lowitz, G. (1983). Can a local histogram really map texture information. *Pattern Recognition*, 16:141–147.

[Malthouse, 1998] Malthouse, E. C. (1998). Limitations of nonlinear pca as performed with generic neural networks. *IEEE Transaction on Neural Networks*, 9(1):165–173.

[Markou and Singh, 2003a] Markou, M. and Singh, S. (2003a). Novelty detection: a review - part 1: statistical approaches. *Signal Processing*, pages 2481–2497.

[Markou and Singh, 2003b] Markou, M. and Singh, S. (2003b). Novelty detection: a review - part 2: neural network based appraoches. *Signal Processing*, pages 2499–2521.

[Markou and Singh, 2004] Markou, M. and Singh, S. (2004). An approach to novelty detection applied to the classification of image regions. *IEEE Transactions on Knowledge And Data Engineering*, 16(4).

[Martin et al., 1998] Martin, J., Devin, V., and Crowley, J. (1998). Active hand tracking. In *IEEE International Conference on Automatic Face and Gesture Recognition*.

[Martinez and Benavente, 1998] Martinez, A. and Benavente, R. (1998). The ar face database. Technical report, CVC Technical Report 24.

[McLeod et al., 1998] McLeod, P., Plunkett, K., and Rolls, E. T. (1998). *An Introduction to Connectionist Modelling of Cognitive Processes*. New York, Oxford University Press.

[Medin and Coley, 1998] Medin, D. L. and Coley, J. D. (1998). *Perception and Cognition at Century's End, 2nd Edition*, chapter Concepts and categorization, pages 403–440. Academic Press, San Diego.

[Micchelli, 1986] Micchelli, C. A. (1986). Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2:11–22.

[Moghaddam, 1999] Moghaddam, B. (1999). Principal manifolds and bayesian subspaces for visual recognition. In *IEEE International Conference on Computer Vision*, pages 1131–1136.

[Moghaddam and Pentland, 1997] Moghaddam, B. and Pentland, A. (1997). Probabilistic visual learning for object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:696–710.

[Mohan et al., 2001] Mohan, A., Papageorgiou, C., and Poggio, T. (2001). Example-based object detection in images by components. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:349–361.

[Morozov, 1984] Morozov, V. A. (1984). *Method for Solving Incorrectly Posed Problems*. Springer, New York.

[Moses and Ullman, 1992] Moses, Y. and Ullman, S. (1992). Limitation of non-model-based recognition schemes. In *European Conference of Computer Vision*, pages 820–828.

[Nguyen and Smeulders, 2004] Nguyen, H. and Smeulders, A. (2004). Fast occluded object tracking by a robust appearance filter. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1099–1104.

[Oliver et al., 2000] Oliver, N., Pentland, A., and Berard, F. (2000). Lafter: A real-time face and lips tracker with facial expression recognition. *Pattern Recognition*, 33:1369–1382.

[Parke and Waters, 1996] Parke, F. I. and Waters, K. (1996). *Computer Facial Animation*. Wellesley, Mass.

[Parzen, 1979] Parzen, E. (1979). Nonparametric statistical data modeling. *Journal of American Statistical Association*, 74:105–131.

[Pei and Tseng, 2002] Pei, S.-C. and Tseng, C.-L. (2002). Robust face detection for different chromatic illuminations. In *International Conference on Image Processing*.

[Penev and Atick, 1996] Penev, P. and Atick, J. (1996). Local feature analysis: A general statistical theory for object representation. *Neural Systems*, 7:477–500.

[Petsche et al., 1996] Petsche, T., Marcantonio, A., Darken, C., Hanson, S. J., Kuhn, G. M., and Santoso, I. (1996). A neural network autoassociator for inductionmotor failure prediction. In *Advances in Neural Information Processing Systems*, volume 8, pages 924–930.

[Phillips et al., 1998] Phillips, P., Moon, H., Rizvi, H., and Rauss, P. (1998). The feret evaluation methodology for face-recognition algorithms. Technical report, NISTIR 6264.

[Plankers and Fua, 2003] Plankers, R. and Fua, P. (2003). Articulated soft objects for multi-view shape and motion capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Qian et al., 1998] Qian, R. J., Sezan, M. I., and Matthews, K. E. (1998). A robust real-time face tracking algorithm. In *International Conference on Image Processing*, volume 1, pages 131–135.

[Raja et al., 1998a] Raja, Y., Mckenna, S. J., and Gong, S. (1998a). Colour model selection and adaptation in dynamic scenes. In *European Conference of Computer Vision*.

[Raja et al., 1998b] Raja, Y., Mckenna, S. J., and Gong, S. (1998b). Tracking colour objects using adaptive mixture models. *Image Vision Computing*, 17(17):225–231.

[Rumerlhart et al., 1986] Rumerlhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by back-propatation errors. *Nature*, 323:533–536.

[Saitoh, 1988] Saitoh, S. (1988). *Theory of Reproducing Kerels and its Applications*. Longman, Harlow, U.K.

[Samaria and Harter, 1994] Samaria, F. and Harter, A. (1994). Parametrisation of a stochastic model for human face identification. In *Proc. IEEE Workshop on Applications on Computer Vision*.

[Scholkopf, 1997] Scholkopf, B. (1997). *Support Vector Learning*. Oldenbourg Verlag, Munich.

[Scholkopf et al., 1995] Scholkopf, B., Burges, C. J. C., and Vapnik, V. N. (1995). Extracting support data for a given task. In *Int. Conf. on Knowledge Discovery and Data Mining*.

[Scholkopf et al., 1999] Scholkopf, B., Mika, S., and Burges, C. J. C. (1999). Input space versus feature space in kernel-based methods. *IEEE Transaction on Neural Networks*, 10(5):1000–1017.

[Scholkopf and Smola, 2002] Scholkopf, B. and Smola, A. (2002). *Learning with Kernels*. MIT Press, Cambridge, MA.

[Scholkopf et al., 1998] Scholkopf, B., Smola, A., and Muller, K. R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319.

[Schwenk and Milgram, 1995] Schwenk, H. and Milgram, M. (1995). Transformation invariant autoassociatioon with application to handwritten character recognition. In *Advances in Neural Information Processing Systems*, pages 991–998.

[Scott, 1992] Scott, D. (1992). *Multivariate Density Estimation*. Wiley, New York.

[Sim et al., 1999] Sim, T., Sukthankar, R., Mullin, M., and Baluja, S. (1999). High-performance memory-based face recognition for visitor identification. Technical report, JPRC-TR-1999-001-01, Carnegie Mellon University.

[Simard et al., 1998] Simard, P., LeCun, Y., Denker, J., and Victorri, B. (1998). Transformation invariance in pattern recognition – tangent distance and tangent propagation. In Orr, G. and Muller, K., editors, *Neural Networks: Tricks of the Trade, Lecture Notes in Computer Science*, pages 239–274. Springer, Heidelberg.

[Suen and Healey, 2000] Suen, P.-H. and Healey, G. (2000). The analysis and recognition of real-world textures in three dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(5):491–503.

[Terrell and Scott, 1992] Terrell, G. and Scott, D. (1992). Variable kernel density estimation. *The Annals of Statistics*, (20):1236–1265.

[Terrillon et al., 2000] Terrillon, J.-C., Shirazi, M., Fukamachi, H., and Akamatsu, S. (2000). Comparative performance of different skin chrominance models and chrominance spaces for the automatic detection of human faces in color images. In *IEEE International Conference on Automatic Face and Gesture Recognition*.

[Terzopoulos and Waters, 1993] Terzopoulos, D. and Waters, K. (1993). Analysis and synthesis of facial image sequences using physical and anatomical models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):569–579.

[Todd and Akerstrom, 1987] Todd, J. and Akerstrom, R. (1987). Perception of three-dimensional form from patterns of optical texture. *Journal of Experimental Psychology: Human Perception and Performance*, 13:242–255.

[Toyama, 1998] Toyama, K. (1998). Prolegomena for robust face tracking. Technical report, Vision Technology Group, Microsoft Research.

[Turk and Pentland, 1991] Turk, M. and Pentland, A. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3:71–86.

[Usui et al., 1991] Usui, S., Nakauchi, S., and Nakano, M. (1991). Internal color representation acquired by a five-layer neural network. pages 867–872. Elsevier Science, New York.

[Vacchetti et al., 2004] Vacchetti, L., Lepetit, V., and Fua, P. (2004). Stable real-time 3d tracking using online and offline information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Valentin et al., 1994] Valentin, D., Abdi, H., O'Toole, A. J., and Cotterell, G. W. (1994). Connectionist models of face processing: A survey. *Pattern Recognition*, 27:120–1230.

[Vapnik, 1995] Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag.

[Vapnik, 1998] Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley, New York.

[Viola and Jones, 2004] Viola, P. and Jones, M. J. (2004). Robust real-time face detection. *International Journal of Computer Vision*, 57:137–154.

[Wern et al., 1997] Wern, C., Azarbayejani, A., Darrell, T., and Pentland, A. P. (1997). Pfinder: Real-time tracking of human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Wiles et al., 2001] Wiles, C., Maki, A., and Matsuda, N. (2001). Hyperpatches for 3d model acquisition and tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1391–1403.

[Winn and Blake, 2004] Winn, J. and Blake, A. (2004). Generative affine localisation and tracking. In *Advances in Neural Information Processing Systems*.

[Wiskott et al., 1997] Wiskott, L., Fellous, J. M., Kruger, N., and Malsburg, C. (1997). Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:775–779.

[Withagen et al., 2002] Withagen, P., Schutte, K., and Groen, F. (2002). Likelihood-based object detection and object tracking using color histograms and em. In *International Conference on Image Processing*, volume 1, pages 589–592.

[Yan et al., 2004] Yan, S., He, X., Hu, Y., Zhang, H., Li, M., and Cheng, Q. (2004). Bayesian shape localization for face recognition using global and local textures. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(1).

[Yang and Waibel, 1996] Yang, J. and Waibel, A. (1996). A real-time face tracker. In *IEEE Proc. of the 3rd Workshop on Applications of Computer Vision*.

[Yeung and Chow, 2002] Yeung, D. Y. and Chow, C. (2002). Parzen window network intrusion detectors. In *International Conference on Pattern Recognition*.

[Ypma and Duin, 1998] Ypma, A. and Duin, R. P. W. (1998). Novelty detection using self-organising maps. In *Progre. Connectionist Based Information Systems 2*, pages 1322–1325.

[Zhang, 2001] Zhang, B. (2001). Face recognition by auto-associative radial basis function network. In *The 3rd International Conference on Audio-and Video-based Biometric Person Authentication (AVBPA)*.

[Zhang et al., 2004a] Zhang, B., Zhang, H. H., and Ge, S. (2004a). Face recognition by applying wavelet subband representation and kernel associative memory. *IEEE Transaction on Neural Network*, 1:166–177.

[Zhang et al., 2004b] Zhang, H., Huang, W., Huang, Z., and Li, L. (2004b). Kernel-based method for tracking objects with rotation and translation. In *International Conference on Pattern Recognition*. Regular papar.

[Zhang et al., 2004c] Zhang, H., Huang, W., Huang, Z., and Zhang, B. (2004c). A kernel autoassociator approach to pattern recognition. *under review by IEEE Transactions on Systems, Man and Cybernetics - Part B*.

[Zhang et al., 2004d] Zhang, H., Huang, W., Huang, Z., and Zhang, B. (2004d). Kernel autoassociator with applications to visual classification. In *International Conference on Pattern Recognition*. Regular papar.

[Zhang et al., 2004e] Zhang, H., Huang, W., Huang, Z., and Zhang, B. (2004e). A particle filtering framework for visual tracking using indirect measurements. In *International Conference on Control, Automation, Robotics and Vision*.

[Zhang et al., 2005] Zhang, H., Zhang, B., Huang, W., and Tian, Q. (2005). Gabor wavelet associative memory for face recognition. *IEEE Transactions on Neural Networks, to appear*.

[Zhang and Benveniste, 1992] Zhang, Q. and Benveniste, A. (1992). Wavelets. *IEEE Trans. Neural Networks*, 3:889–898.

[Zhao et al., 2000] Zhao, W. Y., Chellappa, R., Rosenfeld, A., and Phillips, P. J. (2000). Face recognition: A literature survey. Technical report, UMD CfAR Technical Report CAR-TR-948.

[Zhu et al., 2001] Zhu, Y., Tan, T., and Wang, Y. (2001). Font recognition based on global texture analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:1192–1200.