Efficient Covariance-based Algorithms for Appearance Modeling in Computer Vision

A Project Report

Submitted in partial fulfilment of the

requirements for the Degree of

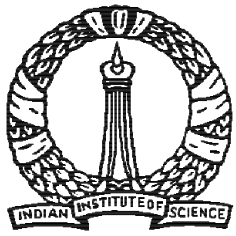
Master of Engineering

in

Faculty of Engineering

by

ADWAY MITRA



Department of Computer Science and Automation

Indian Institute of Science

Bangalore – 560 012

June 2010

Acknowledgements

For this thesis I am deeply indebted to my supervisor Dr. Chiranjib Bhattacharyya, who introduced to me the idea of Joint Diagonalization, and all through the project period helped me in developing mathematically sound models. I would also want to thank Prof. KR Ramakrishnan of EE Department, IISc for his help on the Computer Vision aspects of the work.

Anoop and Ujwal of Computer Vision Lab,EE Department of IISc deserve special thanks. They worked with me all through, and in the initial stages helped me a lot with their codes, and datasets. It was a great pleasure to work with them, and the technical discussions with them were very much fruitful for me.

I am also grateful to my friends both inside IISc (especially Hima, Mayuresh, Mangesh, Sudha and Shweta) and outside IISc (especially Swagato, Sayan and Arijita) who made life enjoyable for me, and provided encouragement all through.

On top of everything I thank my parents for all the love and support they have provided to me all through.

Abstract

In this project, we have explored the Covariance Descriptor for images, and efficient ways of using it for different computer vision tasks like face and object recognition, and visual tracking of objects in videos. The focus of the thesis is in modelling the similarity in structure of covariance descriptors corresponding to similar images. We have proposed two different techniques based on covariance descriptors of similar images. The first one is based on *Joint Diagonalization* of Covariance Descriptors of similar images, and tries to fit a set of common bases to them. It has been used for face and object recognition. We have used it on the ORL datset for faces, and Caltech5,Caltech101 and ETH80 datasets for objects, with impressive accuracies. The second one is that of *Significant Shared Basis* which is a storage-efficient method of handling covariance descriptors of very similar images. This model constrains such a set of images by assuming a shared eigenstructure. It has been shown to be extremely effective for tracking of objects in videos. This method of tracking outperforms the state-of-the-art methods on several publicly available datasets, and is specially suited to handle abrupt illumination changes. As an aside, we have also explored Action Recognition using non-parametric classification of Time-Series.

Contents

List of Tables

List of Figures

Chapter 1

Introduction and Background

1.0.1 Introduction

Computer Vision is the branch of science whose final goal is to make computers see. For this, it makes use of Machine Learning as well as Image Processing techniques. Some of the major problems of Computer Vision include object detection in images, face recognition and object recognition in still images or videos, action recognition in videos, tracking of objects in videos, video segmentation and summarization, and many more. We briefly explain some of these problems.

Face and object recognition are basically classification problem. There are several classes- which represent different persons in case of face recognition, and different objects in case of object recognition. Each class has several instances of the same face or object, with some variations among them. These variations can be due to difference in orientation, illumination condition, viewpoint, scale, noise, partial occlusion and similar factors. In case of objects, the variations are more pronounced, as different objects of the same category can have significant differences in terms of shape, colour and other factors. Action Recognition too is a classification problem, and in this case the different classes represent different actions like walking or running. While in case of face and object recognition the individual instances are static images, in case of action recognition these instances are videos. Once again there is likely to be variation among the instances

within the same class.

In these problems of recognition, the usual technique is to build models for the individual classes of entities (face,object or action). These models are learnt in the training phase. In the testing phase, given a test entity, the task is to find which of the class models it fits the best. Sometimes instead of building class models it is also possible to build models for the individual instances, so that nearest-neighbor classification can be done. This approach is, however, slow.

While considerable progress has been made in face recognition, research in object and action recognition are still in their early stages.

Visual Tracking of objects in videos is another very important task in computer vision. This deals with locating one (or more) object(s) in the successive frames of a video. Trivially this can be done by detecting the object(s) separately in each of the frames, but such an approach is inefficient as it fails to make use of the fact that in successive frames of a video the locations of the target(s) will be close to each other. So, tracking algorithms proceed by searching for the target in a frame in the neighborhood of its location in the previous frames. In this case also we need to build models of the target. Since tracking is a real-time task, these models should be easy to compute and compare. Moreover, the target may undergo appearance changes through the length of the video, and hence the model should be built adaptively. This is known as Adaptive Visual Tracking and is currently a topic of active research.

Object detection is the task of finding the location(s) of one or more instances of a particular object in a static image. It usually involves searching the whole image at different locations and scales. Once again models for the partcular object being searched is required. While a lot of progress has been made in Face Detection, detection of other objects still remains a challenge in the computer vision community.

1.1 Covariance Descriptors for Appearance Modeling

It is evident from the discussions above that main challenge is in building models for different objects, which can be efficiently computed. This is sometimes known as *Appearance Modelling*. But the greatest challenge to all existing techniques of computer visions is posed by the huge variations in appearances that exist in real-world images, even for the same object. Such variations can occur due to different illumination conditions, viewpoints, scales and numerous other factors. Hence one of the main priorities of Computer Vision is to build descriptors for objects which would be robust to such issues.

For such representation, there are many features described in image processing literature. These features include colour/pixel intensity, image gradients, filter responses, wavelet transforms, SIFT and lots of others. These features are normally pixel-descriptors. Different features have different properties, and different features are suitable for different computer vision tasks. For example, some features may be fast to compute, while others might be robust to noise and other issues. However since visual entities consist of a region or continuous group of pixels, representation of such entities require *region descriptors*.

A region descriptor can be theoretically considered to be a joint distribution of pixel features within the region. Since histogram is a discretized non-parametric representation of a distribution, some of the earliest forms of region descriptors were histograms of features(pixel descriptors) as described above, calculated at the pixels within the region. In their seminal paper [4] Tuzel et al propose a *covariance region descriptor*. Here the region is represented by the **covariance** of the features of the pixels lying within the region. This method of representation has the advantage that it allows fusion of multiple features. Hence such a descriptor is quite robust to noise in the pixel features, and is partially invariant to rotation and scale of the region. They can also be computed quickly, using Integral Images, as mentioned in the above-mentioned paper.

1.2 Geodesic Distance

After choosing a descriptor for entities, in most computer vision tasks it is necessary to compare entities using their descriptors. Hence for every descriptor we need a similarity/distance measure. Since the Covariance Matrices do not lie in the Euclidean Space, we cannot use the standard Euclidean distance. Instead, the authors of [4] proposed a measure called *Geodesic Distance* which was originally proposed in [5] using a Lie Algebric structure. The Geodesic Distance between a pair of covariance matrices C_1 and C_2 is given by

$$\rho(C_1, C_2) = \sqrt{\sum ln(\lambda_i(C_1, C_2)^2} \tag{1.1}$$

where the $\lambda_i(C_1, C_2)$ is given by the generalized eigenvalues of C_1 and C_2. In other words, λ_i is the i-th root of the generalized eigenproblem

$$\lambda_i C_1 X_i = C_2 X_i \tag{1.2}$$

This measure was used in [4] for computer vision problems like object detection and texture classification. It can also be used in classification-based problems like face and object recognition.

1.3 Focus of the Thesis

The thesis makes extensive use of Covariance Descriptors. The focus of the thesis is to handle sets of images having some similarity. For example, in case of face and object recognition as discussed earlier, each class consists of images corresponding to the same person (in case of faces) or same category of objects. Also in case of videos, an object will look almost similar in the successive frames. Given that a covariance descriptor is a good representation, the covariance descriptors for similar images should have some common structure. In this thesis we have tried to model this similarity in two different

ways. These are as follows

i) In the first formulation, as in Chapter 2 We have tried to find a set of common basis vectors for these covariance descriptors. These vectors, in our formulation, need not be orthogonal to each other. We have tried to estimate them by a *Joint Diagonalization* of the covariance descriptors. This method has been used in Face and Object Recognition.

ii) In the second formulation, as in Chapter 3 we have constrained the sets of covariance descriptors to have a shared eigenstructure. In other words, the i-th Principal Components of the individual images are same. We have given an algorithm to estimate this shared eigenstructure, and find a template Covariance Descriptor for the set. This method has been used in Visual Tracking of objects.

As an aside, we have studied the problem of action recognition. We have used a method based on non-parametric classification of time-series of features. We have proposed a heuristic to compare two time-series. We have carried out experiments on standard datasets of action recognition, and have obtained interesting results.

Chapter 2

Face and Object Recognition using Joint Diagonalization

2.1 Introduction

In this chapter, we introduce the first of two method of handling sets of covariance descriptors corresponding to similar images. As said earlier, since Covariance Descriptors are a good representation of images, covariance descriptors of similar images should have some similarity in structure. Here, we attempt to find a set of common basis vectors $[V_1, V_2, V_3, \ldots, V_d]$, where d is the dimensionality of the covariance descriptos, as ascertained by the features being used. To estimate these vectors, we put a constraint that these should be able to *Jointly Diagonalize* the covariance descriptors. To estimate such a set of vectors, we make use of existing algorithms for Joint Diagonalization of matrices.

2.2 Joint Diagonalization of Covariance Matrices

In Signal Processing literature there exists a rich set of algorithms for *Approximate Joint Diagonalization* of covariance matrices. In this work, we have used such joint diagonalization algorithms for handling sets of covariance matrices.

Given a set of covariance matrices $C_1, C_2, \ldots, C_n$, an approximate Joint Diagonalization Algorithm tries to find a matrix V which minimizes an objective function under the constraint

$$F_k = V^T C_k V \forall k \tag{2.1}$$

This objective function is a measure of diagonalization, and it is minimized when the F_k matrices are diagonal. Several notions of diagonality exist in literature. One is the function

$$L_k = log(det(diag(F_k))) - log(det(F_k)) \tag{2.2}$$

The algorithm by Pham as in [3] makes use of this.objective function.

There are also other measures of diagonalization, like the *Frobenius Norm* of the off-diagonal elements of the F_k matrices, which is given by

$$Fro(C) = \sum_{i,j,i \neq j} {C_{ij}}^2 \tag{2.3}$$

This objective function is used by the algorithms of Cardoso [2] and Ziehe [1].

Another family of Joint Diagonalization algorithms uses the *Subspace Fitting Formulation*. Given a set of covariance matrices $C_1, C_2, \ldots, C_k$, this is to estimate another set of diagonal matrices $D_1, D_2, \ldots, D_k$ and a single diagonalizing matrix V so as to minimize

$$min_{V,D_1,D_2,\ldots,D_k} \sum_i ||C_i - V D_i V^T||^2 \tag{2.4}$$

In our work, we have made use of Pham's Algorithm, as it was found to be giving best results for our experiments. We have assumed that for each class (faces or objects) it is possible to find a set of basis vectors to characterize that class, and these vectors are obtained by jointly diagonalization of the Covariance Descriptors of the images in that class. In other words, the V matrices estimated by the Joint Diagonalization algorithm is a model for that particular class. As this algorithm tries to minimize the function 2.2, we use this same function to measure how good a test covariance descriptor matches a particular class model.

2.3 Surveys on Computer Vision Tasks

2.3.1 Face Recognition: A Brief Survey

Face Recognition is one of the most well-studied problems on computer vision. It is a classification problem, where the training set is a set of labelled face images for a number of persons. Given the test image of a face, we have to estimate which person it belongs to. The most well-known paper in this field is [6] which introduced Eigenfaces, which is a PCA-based approach. KPCA, the kernelized version of PCA has been used successfully for face recognition [14]. Another classical approach that has been used for face recognition in [7] is *ICA* (Independant Component Analysis) which tries to fit statistically independant bases for the pixel data. Discriminative techniques such as LDA (Linear Discriminant Analysis) has also been used for face recognition, as in [8]. Other techniques include Active Appearance Models as in [10] and Elastic Bunch Graph Matching, as in [9]. A Bayesian technique has also been studied in [11]. The paper [12] provides an excellent survey of the face recognition literature. Different papers suggest different learning techniques, and different descriptors or feature sets. More recently, the Covariance Descriptor of [4] was used with Gabor features as pixel descriptors in the paper [13]. In such attempts the Geodesic distance measure between covariance matrices are used. This paper shows the efficacy of Gabor features for modeling of faces.

2.3.2 Object Recognition: A Brief Survey

Like Face Recognition, object recognition is also a classification problem. It is a relatively new problem, but lots of techniques have already been proposed. Some of the earlier ones were the *Bag-of-Words* model as in [16]. The paper [17] computes pixel descriptors on *salient regions* in the image, and uses a *LDA-like Topic Model* which assigns these regions to a *latent theme*. It characterizes each object category as a theme distribution, and uses it to classify a test image. An approach that has recently captured a lot of attention in the object recognition community is *Multiple Kernal Learning*, which attempts to combine kernels obtained from several different pixel descriptors. This method is more

successful because it can use a large number of pixel descriptors, which is imperative on large datasets having a lot of inter-class and intra-class variance. Some of the recent papers regarding this technique are [18] and [19]. The paper [20] provides an excellent study of different baseline and involved techniques of combining features.

2.4 Experiments on Joint Diagonalization

We have used the Joint Diagonalization-based method as described above, for the computer vision tasks like Face Recogntion and Object Recognition. These are applications which involve sets of similar/related images, whose covariance descriptors can be assumed to be constrained by the assumption of common basis vectors. As explained above, to estimate these common basis vectors, we use an Approximate Joint Diagonalization algorithm. For all the experiments decribed below, we tried out the algorithms mentioned above, and the one by Pham as in [3] was found to give the best performance. The results described below were obtained using this.

Face and Object Recognition are Classifcation Tasks, and hence involve a training and a testing phase. Thecomplete classification algorithm is shown in figure 2.1.

2.4.1 Object Recognition

For object recognition, we perform experiments on 3 datasets- namely Caltech5,Caltech101 and ETH80. As pixel descriptors, we have used the 4-dimensional Gradients Features (the derivatives and second derivatives in the X and Y directions), the 40-dimensional Gabor Filter outputs and the 44-dimensional combination of these two. We also tried the 128-dimensional SIFT features, but the results were very poor, so we do not report them.

Caltech5 dataset has 5 categories- cars (rear view), airplanes, bikes, leaves and human faces. We made use of the provided annotations to select the object of interest out of the background. The Caltech101 has 101 object categories, with substantial interclass and intraclass variations in terms of appearance. The ETH80 has 8 classes, each having

TRAINING

1. For each training image represent each pixel as the d-dimensional vector, and build 5 dXd covariance matrices for it. (one for the full image, two for the upper and lower halves, two for the left and right halves).
2. Jointly diagonalize the matrices of all the training images belonging to the same class using a suitable joint diagonalization algorithm.
3. Save the 5 V matrices for each class k $(V_{k1}, V_{k2}, V_{k3}, V_{k4}, V_{k5})$

TESTING

1. Take a test image. Represent each pixel as the d-dimensional vector, and build 5 dXd covariance matrices $(C_1, C_2, C_3, C_4, C_5)$ for it.
2. For k=1 to n (number of classes)
   - for i=1 to 5
   - Calculate $F_{ki} = V_{ki}^T C_i V_{ki}$
   - Calculate the diagonality measure d_{ki} for F_{ki}
   - end for
   - Calculate the measure $d_k = \sum_i d_{ki} - \max_i d_{ki}$
3. end for
4. If the measure d_k was least for class k, classify the test image to class k.

Figure 2.1: Classification Algorithm

Training Set	Testing Set	Accuracy
10	40	83.02
20	30	81.62
25	25	78.95

Table 2.1: Caltech5 dataset: using 4-dimensional Gradient Features

Training Set	Testing Set	Accuracy
10	40	90.28
20	30	91.55
25	25	90.35

Table 2.2: Caltech5 dataset: using 40-dimensional Gabor Features

10 instances taken from 41 different viewing directions. Below we report the results on these datasets, with different features and different sizes of training and testing sets. **It is to be noted that the figures reported below are obtained by two sets of training/testing data and taking the average of the two accuracies so obtained for each of the cases. These figures can change if more samples are taken**

In ETH80 our LOO accuracy is 100 per cent. Use of 5 training images per class yields 87 per cent accuracy, while using 2 images per class for training yields 71 per cent. In contrast the LOO accuracy reported in [15] is just 86 per cent.

2.4.2 Face Recognition

After object recognition we focus attention on Face recognition. As pixel descriptors we use 40-dimensional Gabor Filter outputs, since they are known to be very effective for faces. We use the ORL dataset, which has 40 people, with 10 images of each. The results for different sizes of the training/testing set are shown below. These results are

Training Set	Testing Set	Accuracy
15	15	32.46
20	10	35.15
25	5	40.61

Table 2.3: Caltech101 dataset: using 44-dimensional Combined Features

Training Set	Testing Set	Accuracy
2	8	81.72
5	5	94.00
7	3	98.75

Table 2.4: ORL dataset: using 40-dimensional Gabor Features

considerably better than those obtained by other techniques, like those of [13]. **PCA-based face-recognition** gives only **67 per cent** accuracy on the ORL Dataset with 2 Training images per class and **Geodesic Distance**-based method gives 80 per cent.

Chapter 3

Significant Shared Basis

3.1 Introduction

In this part of our work we propose a new concept for appearance modeling of very similar images. This concept helps in efficiently representing a set of very similar image patches, such as those obtained from the successive frames of a single video.

An *image patch* is defined to be a window cropped out of an image, typically containing one single object, like the tracking target or its parts. Consider an image patch I_k , consisting of n_k pixels. At each pixel p_{ik} we define an d-dimensional feature vector X_{ik} . Let us consider that the whole image patch can be modelled as a Multivariate Normal distribution in the feature space.

$$X_{ik} \sim N(X_k, C_k) \tag{3.1}$$

The assumption that the distribution corresponding to different image patches (containing the same object, but at different instants of time) have the same mean is justified but assuming that they have the same co-variance is extremely restrictive. The distributions of the patches may not be exactly same but should be quite close to each other. It is precisely this intuition which we would like to capture through the proposal of Shared Significant Basis(SSB). The covariance matrix of an image patch can be written in terms

of its eigenvectors and eigenvalues as follows:

$$C_k = \sum_{1 \leq j \leq d} \lambda_{jk} \beta_{jk} \beta_{jk}^T \tag{3.2}$$

The eigenvectors of a covariance matrix can be ranked according to the magnitude of the corresponding eigenvectors. The eigenvector associated with the largest eigenvalue is called the *First Principal Component*, the one attached with the second-largest eigenvalue is called the *Second Principal Component* and so on.

This model assumes that, for all the covariance matrices, the j-th Principal Components are equal, for $1 \leq j \leq r$, for some r. Mathematically,

$$C_k = \sum_{1 \leq j \leq r} \lambda_{jk} \beta_j \beta_j^T + \sum_{r+1 \leq j \leq d} \alpha_{jk} \psi_{jk} \psi_{jk}^T \tag{3.3}$$

Moreover, the model assumes that the eigenvalues corresponding to the t-th Principal Components ($r \leq t$) are small enough to significantly affect the covariance matrix, and the first r eigenvectors and their eigenvalues are enough to sufficiently approximate the covariance matrix for computer vision tasks. We call these r eigenvectors as the *Significant Shared Basis* for the set of covariance matrices. So we can write the covariance matrix as:

$$C_k = \sum_{j} \lambda_{jk} \beta_j \beta_j^T \tag{3.4}$$

In the experiments section, we show the validation of the two assumptions of the SSB model, viz the shared eigenvectors and the insignificance of the smaller eigenvalues.

3.2 Algorithm to compute SSB

Obviously, in real-world data, the Principal Components will not be all equal as suggested by the model. However, they are very similar to each other, as we will later show experimentally. The problem of approximating these sets of Principal Components with

a single set of Basis Vectors is an Optimization Problem, which has a closed-form solution as derived below. We start by estimating the first shared eigenvector, and incrementally estimate the others in order. Consider the i-th SSE β_i. The i-th principal component of the j-th example patch is denoted by X_{ij}. Then, the i-th shared eigenbasis vector should be approximately

$$u_i = \frac{\sum_{i,j} X_{ij}}{\sqrt{(\sum_{i,j} X_{ij})^T (\sum_{i,j} X_{ij})}} \tag{3.5}$$

However, β_i should also be normal to the previously computed shared vectors, and its norm should be 1. So we write the Lagrangian equation as

$$L = - < \beta_i, u_i > + \sum_{1 \leq n \leq i-1} \mu_n < \beta_n, \beta_i > + \lambda(1 - < \beta_i, \beta_i >) \tag{3.6}$$

Solving the Lagrangian, we get

$$\mu_n = < u_i, \beta_n > \tag{3.7}$$

and

$$\lambda = \sqrt{\frac{1}{4}(1 - \sum_n {\mu_n}^2)} \tag{3.8}$$

and finally,

$$\beta_i = \frac{1}{2\lambda}(u_i - \sum_n \mu_n \beta_n) \tag{3.9}$$

This way, we can evaluate the Significant Shared Basis vectors.

It should be noted that such an approximated Covariance Matrix will not be Positive Definite. Hence we make the following correction:

$$C_k = \sum_{1 \leq j \leq r} \lambda_{jk} \beta_j \beta_j^T + \sum_{r+1 \leq j \leq d} \alpha \gamma_j \gamma_j^T \tag{3.10}$$

where α is some small value. These pseudo-vectors can be generated from the above formulation to satisfy the constraints of (20). However, the u need not be calculated as in (19), and instead we can use any arbitrary vectors with norm 1 in its place, provided it does not match any of the vectors β_j already computed. We have a set of canonical vectors- the individual colums of the identity matrix I_d, from which we choose u.

3.3 Template Covariance Matrix

The representation we have proposed allows us to do Nearest-Neighbor classification, using the means and covariance matrices of the individual image patches in the classes. However, in a real-time application like tracking, nearest-neighbor classification is not appropriate because it is slow. Hence, we can go one step further and fit a Template Covariance Matrix for each class, once again using the SSBs of the examples in the class. Such a Template Covariance Matrix can be written as:

$$C = \sum_{1 \leq j \leq r} \lambda_j \beta_j \beta_j^T + \sum_{t+1 \leq j \leq d} \alpha \gamma_j \gamma_j^T \tag{3.11}$$

Next, we need to find the variances along these SSBs. We find these values from the original eigenvalues by considering a projection the data on to these vectors. The variance along the j-th common eigenvector is given by

$$\lambda_j = \frac{\sum_{1 \leq i \leq N} \sum_{1 \leq k \leq d} n_i \lambda_{ik} cos\theta_{ik}{}^2}{\sum_{1 \leq i \leq N} n_i} \tag{3.12}$$

where N is the number of example patches in the class, n_i is the number of pixels of the i-th patch, λ_{ik} is the k-th eigenvalue of the covariance matrix for the i-th patch and θ_{ik} is the angle made by the k-th principal component of the i-th patch with β_j.

Theorem 1. *The variance of the data along the SSB vector β_j is well approximated by the above formula*

Proof: Consider all the N_i data points $X_{i1}, X_{i2}, X_{i3}, \ldots, X_{iN_i}$ of the i-th image in the set. The mean of this class is then given by

$$X_i = \frac{\sum_{1 \leq n \leq n_i} X_{in}}{n_i} \tag{3.13}$$

By assumption of the model, $X_1 = X_2 = \cdots = X_K = X$ (the pathches from the different images have the same mean).

Now, let us consider the variance along the vector β_j. Then, we need to consider the projections of all the data points on this vector. The projection of a datapoint X on β_j is given ${\beta_j}^T X$.

Also, the n-th point of the k-th image can be written in terms of the individual principal components of this image as

$$X_{kn} = \sum_{1 \leq i \leq d} ({u_{ik}}^T X_{kn}) u_{ik} \tag{3.14}$$

as u_{ik} is an unit vector.

Similarly,

$$X_k = \sum_{1 \leq i \leq d} ({u_{ik}}^T X_k) u_{ik} \tag{3.15}$$

So,

$$X_{kn} - X_k = \sum_{1 \leq i \leq d} ({u_{ik}}^T X_{kn} - {u_{ik}}^T X_k) u_{ik} \tag{3.16}$$

So, the variance along β_j is given by

$$\lambda_j = \frac{\sum_k \sum_n {\beta_j}^T (X_{kn} - X)^2}{N} \tag{3.17}$$

and hence,

$$\lambda_j = \frac{\sum_k \sum_n {\beta_j}^T (X_{kn} - X_k)^2}{N} \tag{3.18}$$

and hence,

$$\lambda_j = \frac{\sum_k n_k \sum_n \frac{{\beta_j}^T (X_{kn} - X_k)^2}{n_k}}{N} \tag{3.19}$$

Then, substituting with 3.3 ,

$${\beta_j}^T (X_{kn} - X_k) = \sum_{1 \le i \le d} ({u_{ik}}^T X_{kn} - {u_{ik}}^T X_k)({\beta_j}^T u_{ik}) \tag{3.20}$$

Squaring and substituting into 3.3 and ignoring the covariance terms among the orthogonal axes, we finally get

$$\lambda_j \approx \frac{\sum_{1 \le i \le N} \sum_{1 \le k \le d} n_i \lambda_{ik} {cos\theta_{ik}}^2}{\sum_{1 \le i \le N} n_i} \tag{3.21}$$

where

$$cos\theta_{ik} = \beta_j^T u_{ik} \tag{3.22}$$

□

Chapter 4

Visual Tracking using SSB

4.1 Motivation

Having explained the mathematics of SSB in details, we now proceed to the main computer vision task we addressed using SSB: tracking. This is the task of following a target object through the frames of a video, and noting its locations. Tracking of a target in a video is important in lots of real-world applications. A good tracking algorithm should be robust to illumination changes, noise, partial occlusion, background etc.

A main challenge in visual tracking is the change in appearance of the target. It is necessary to adaptively rebuild the target model throughout the tracking video. This approach is called *Adaptive Visual Tracking*, and it is a hot topic of research. The main challenge in adaptively rebuilding the target model is that the model should be computable in real-time. Also, it should require minimum resources like storage so that it can be implemented on low-memory devices like sensor nodes which can be be deployed in real-life scenarios like surveillance, where object tracking is vital.

The advantages of using the method of SSB for tracking are twofold:

- The image patch is viewed as a distribution in the feature space, which allows the use of any features (pixel descriptors) as may be considered appropriate. Use of such features can lead to robustness to the problems mentioned above.

- A model consists of simply a mean feature vector and r SSB vectors and their corresponding eigenvalues. It is not necessary to store the entire covariance matrix. And in practice, r is at most 3. Hence this technique is extremely storage-efficient, compared to other contemporary techniques. Moreover, the model size is completely independent of the target size, and depends only on the feature dimension.

- The above formulation is computationally efficient, and hence suitable for real-time adaptive tracking.

4.2 Visual Tracking: A brief survey

Lots of approaches have been tried for tracking since the 1990s. The most popular tracker is perhaps the Mean-Shift Tracker proposed in 2003 [23] that builds a colour histogram-based model of the target, and searches for the best match in the surroundings by comparing colour histograms. However, one problem with this is that it is too much dependant on colour, and if the colour of the target changes (due to illumination change) it is bound to fail, more so as it is not adaptive. Also, if the object is very small, the colour histogram will be difficult to build. Moeover, this algorithm is not adaptive.

The concept of building an eigenstructure for the target was proposed as early as 1996 by Black and Jepson [21]. But that required a very detailed training of the model, with examples collected from a large number of viewpoints. A recent work on tracking that also attempts to build a subspace representation of the target is IVT [22]. This is an adaptive procedure, which progressively builds the subspace with image patches consisting of the tracked object from the successive frames. It modifies the SKL algorithm [26] for online PCA calculation and includes a forgetting factor to reduce importance of the older frames while updating the model.

Some papers pose tracking as a classification problem, in which both positive and negative examples are used, corresponding to the target and non-targets respectively. Attempts have been made to classify a location as target/non-target from these examples. This paradigm is known as *tracking by detection*. A recent paper on this is [27], which

performs online Boosting along with Multiple-Instance Learning to build a powerful classifier.

Covariance descriptors for images have been shown to be robust to illumination and appearance changes as well as slight deformations, as they allow blending of multiple powerful features, without a blow-up in storage. *Covariance Tracking* was proposed in 2006 in [25] which used Geodesic Distance as the distance measure between covariance matrices. It computes the Geodesic Mean of several example covariance matrices as the representation of the target.

Other recent works on tracking include tracking by *L1-norm minimization*. In this work, several target templates are stored along with trivial templates. They attempt to represent any image patch as a sparse combination of the target and trivial templates, to handle phenomenon like occlusion/clutter/illumination change. This algorithm performs better than Covariance Tracker, but is prohibitively slow and has heavy storage requirements.

4.3 Experiments on Tracking

4.3.1 Tracking Algorithm

Here, we focus on appearance modelling of the target. We build models for the target, and the 8 peripheral regions around it. Given any image patch, we can classify it into one of these models. This way we can find the target in the frames. Also, since we are dealing with videos, we make use of the property that the target will be in the vicinity of its location in the previous frame, and restrict the search space.

We look at every model as a Gaussian Distribution (of pixels) in the feature space, and the example patches in each model as sets of samples drawn from such a distribution. A Gaussian distribution is fully described with its mean and covariance matrix. The mean of each model can be estimated from the sample means of the individual examples, and the template covariance matrix can be computed efficiently by the formula described in the previous section. As the appearance of the object may keep changing throughout

the video, it is necessary to continuously rebuild the models (the template covariance matrix and template mean). Since the SSB model holds for very similar image patches,in our experiments we assume every 5 successive frames to have this property. So at each frame we estimate the template covariance matrix using the past 5 frames. Next, when we are given a patch for classification into one of these models, we again view it as a set of samples drawn from one of these distributions. The standard similarity metric between two distributions is the **Kullback-Leibler Divergence** between them. The K-L divergence between two multivariate Gaussians (μ_f, Σ_f) and (μ_g, Σ_g) is given by

$$\frac{1}{2}(ln|\Sigma_f| - ln|\Sigma_g| - d + tr({\Sigma_f}^{-1}\Sigma_g) + (\mu_g - \mu_f)^T {\Sigma_f}^{-1}(\mu_g - \mu_f)) \tag{4.1}$$

So, given any image patch I, we find its mean and covariance matrix. The mean and covariance of each model are already available. So we simply find the class for which the K-L divergence of I is minimized, and assign to I the label of that class.

At any instant t, the location of the object is given by $X = (x, y)$ which are the 2D coordinates in the image plane. The target is localized using a rectangular box centered at X. The dynamics is modelled by Random Walk model, where

$$p(X_{t+1}|X_t) = N(X_{t+1}; X_t, \Sigma) \tag{4.2}$$

We assume that if the target is at X_t in the t-th frame, the candidate locations in $(t+1)$-th frame are modelled by the above equation. We draw samples from the model and evaluate each sample.

The pseudocode for the Tracking algorithm is shown in figure 4.1

4.3.2 Experiments

In this section we present experimental results. As said earlier, our representation allows efficient use of multidimensional features, which can improve performance. To drive home this point we choose datasets where these can be important. It can be seen that our method achieves better performance than traditional and state-of-the-art trackers.

TRACKING ALGORITHM (WITH TEMPLATE UPDATION)

0 Initialize the locations X_1, X_2, X_3, X_4, X_5 of the target and its size (δ_1, δ_2) in the first 5 frames.
1 Crop out rectangular regions centered at X from these frames.
Calculate features, the mean and first r eigenvectors along with their eigenvalues and store them as examples
Form the mean, SSB-s and corresponding eigenvalues for the template covariance matrix.
These form the model for the target.
2 Also crop out 8 rectangular regions around the target (as shown) from these frames.
Calculate features, the mean and first r eigenvectors along with their eigenvalues and store them as examples.
Form the 8 sets of means, SSB-s and corresponding eigenvalues for the template covariance matrices.
These form models for the 8 peripheral regions.

3 for $i = FirstFrame : LastFrame$
3a. Set mean: the position of the target in the previous frame
3b. Set covariance matrix to appropriately chosen constant value.
3c. Draw N samples from a normal distribution having the mean and covariance as defined above.
These are candidate positions of target in the current frame.
3d for $i = 1 : N$
Crop out 9 rectangular windows around the i-th locations
Evaluate the features, and the mean and covariance matrices for all 9 regions
for each of these 9 regions,
calculate the K-L divergence with the stored models for the 9 regions
classify that region into one of these 9 models, based on the K-L Divergence
end **for**
if the number of correctly classified regions is maximum so far, put $id = i$.
3e end **for**
3f Declare the id-th sampled location as the location of the target in the current frame.
3g Crop out the 9 rectangular regions around this location as above.
Calculate their means and first r eigenvectors and eigenvalues and store them as examples
3h Discard the mean,eigenvectors and eigenvalues for the oldest example currently stored
3i Rebuild the new 9 sets of means, SSB-s and eigenvalues (**TEMPLATE UPDATE STEP**)
4 end **for**

Figure 4.1: SSB-based Tracking Algorithm

Figure 4.2: Image Patches containing target in successive frames

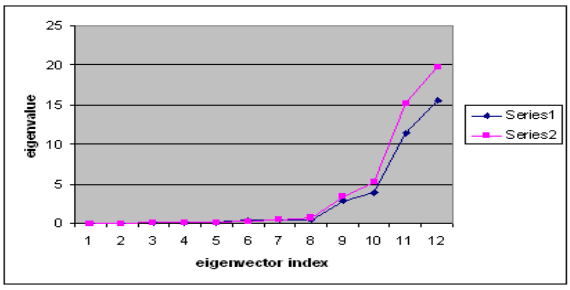


Figure 4.3: Only the first 4 eigenvalues are significant, the rest are negligible

However, before showing the tracking results, we first validate the assumptions of the SSB model.

Validations As mentioned earlier, we validate the assumption that the patches corresponding to the same object in successive video frames have similar leading eigenvectors in the feature space. We consider 2 sets of patches from 5 successive frames, as shown in the figure 4.2. We use 12-dimensional Gabor Features (3scalesX4orientations), and show the **average of the dot-products of pairs of eigenvectors** from these 5 patches. As the eigenvectors have unit norm, the dot-product of two of them is equal to the cosine of the angle between them, and values close to 1 indicate greater similarity.

The other assumption is that, for the feature sets we use, the eigenvalues corresponding to only the 2-3 leading eigenvectors is significant compared to the remaining eigenvalues. We show in the figure 4.3, for two image patches corresponding to tracking targets, how the eigenvalues fall off.

Track	1st PC	2nd PC	3rd PC
1	0.974	0.973	0.984
2	0.975	0.962	0.918

Table 4.1: Average Cosines between pairs of Eigenvectors. PC stands for Principal Components

Camera placed at Height we show the tracking results on several publicly available datasets. We have reported results on two PETS 2000 sequence which shows a person walking across a field on to a road with the camera placed high up, so that the size of the subject is very small (20X70). We use image gradients as features that are easy and fast to compute. Hence in these cases tracking by our method can be implemented even on a cheap resource-constrained camera like a mobile phone. The results have been shown in the Appendix. In the frames of the video, the estimated position of the target are shown with a red rectangle. Greater is the overlap of this rectangle with the target, better is the performance. Our results can be seen to be equivalent to IVT and better than Mean-Shift on these datasets.

Videos with Abrupt Illumination Changes Next we show results on our own videos which involve abrupt illumination changes. We show two such video sequences, when a person is walking through a corridor from a well-illuminated area to an obscure area. In both cases our method goes through while IVT fails. We use Gabor filter outputs as features, since we have observed that these are more robust to illumination change than image gradients. As Mean-Shift Algorithm makes use of colour histograms which are sensitive to Abrupt Illumination changes, we do not compare with it in this case.

Additionally, we also show results on the Dudek dataset available at [28] and find our performance comparable to or better than existing approaches.

Figure 4.4: EE1 Sequence: The results shown are for IVT and Our Method, from top to bottom

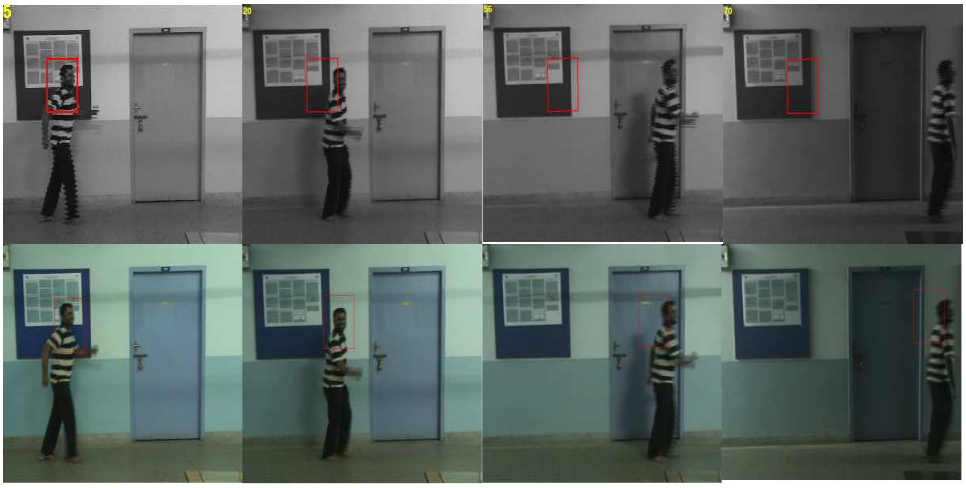


Figure 4.5: CSA Sequence: The results shown are for IVT and Our Method, from top to bottom

Figure 4.6: PETS2000 Sequence 1: The results shown are for IVT, Mean-shift and Our Method, from top to bottom

Figure 4.7: PETS2000 Sequence 2: The results shown are for IVT, Mean-shift and Our Method, from top to bottom

Figure 4.8: Dudek Sequence: The results shown are for IVT, L1 and Our Method, from top to bottom

Chapter 5

Action Recognition

5.1 Introduction

Action Recognition in videos has been an actively studied topic in Computer Vision. Although it was first perceived in the late 1990's, it has become a really hot topic in the vision community around 2006, and especially after the seminal paper [29]. Papers in literature on action recognition use silhouettes [29], [30], [31], optical flows [32], space-time interest points [33], shape and appearance models [34] and many other types of features.In this work we view an action as a time-series of silhouettes of the actor(s). Since it is difficult to extract silhouettes in details, we propose a set of very simple features of the foreground blobs of the actor, and construct a time-series based on each of these features over a sequence of video frames depicting the action. For classification we maintain a dictionary of actions, and for each action in the database we maintain a few prototype time-series of each of the proposed features. Given any test video we can construct such a time-series of the proposed features and classify the action as one of those in the database, by time-series classification techniques. The proposed method is not only simple, intuitive and computationally efficient, but it also outperforms several state-of-the-art methods on standard datasets.Apart from the standard problem of action recognition we extend our analysis to include cases where the actor gets occluded sometimes, so that a time-series with missing values is obtained. Moreover, we also

suggest an approach to detect non-homogenity in actions.

5.2 Background

Recognition of actions from videos is extremely important in Surveillance, Human-Computer Interaction, Sports Video summarization and many other fields. A lot of methods for action recognition have been proposed.Several approaches have been tried, including space-time volumetric matching, building generative models for motions of individual body components, contour graphs, topic models like LDA and many others. The basic problem with many of these methods is that they require extraction and comparison of highly detailed silhouettes, which is very difficult in real-world images due to occlusion and noise. Other methods assume a parametric model like HMM, which may not be appropriate, especially for *short video sequences* where the parameters are difficult to estimate. So in this work we use a nonparametric approach. Our method is meant for surveillance scenarios, where the background model is available, and the density of the people is not very high. This is a supervised learning approach, in which a database is maintained for the different actions, and in the database there are prototype time-series on the different features for each action. We use features of the actor's foreground blob. We first check our method for recognition of standard actions like running, walking and skipping for which standard datasets are publicly available. Then, we extend our analysis to include more realistic situations like occlusion. This, in our formulation, translates to the problem of time-series classification in presence of missing values. The third part of the work includes detecting abnormal behavior, which in our case translates to testing the homogeneity of a time-series.

5.3 Nonparametric Time-Series Classification

For nonparametric time-series classification, Distance Measures are required between two time-series. There are many time-series distance measures in literature. A survey of

these are provided in [39]. Some of these are the Euclidean Distance, Fretchet Distance and Dynamic Time Warp Distance. We prefer to use a raw data-based distance measure [39] in this case. One big problem with the distance measures mentioned above is that the distance between two time-series depends on the absolute values, not the shapes of the curves. Also, Euclidean distance is too rigid and restrictive as it considers element-by-element differences, which requires the two time-series to be exactly aligned to get proper results. Fretchet distance is a measure of the maximum difference between the two series at a single time-point, and so it does not capture the distance between the two series as a whole. In our approach, we try to express one time-series as the linear transformation of the other. Given two timeseries $x(t)$ and $y(t)$ this requires finding numbers a and b such that we have

$$y'(t) = a + b * x(t) \tag{5.1}$$

where $y'(t)$ is the close to $y(t)$ in the absolute sense. This step eliminates the problem of the distance being dependant on the absolute values. Once this has been done, we use the Euclidean Distance after finding the best alignment of the two series. This alignment step removes the problem with Euclidean distance mentioned earlier.

The algorithm to find the dissimilarity between two time-series is as follows:

5.4 Action Recognition by Time-Series Classification

For any learning problem we need feature sets. Lots of different types of feature sets have been proposed in lietrature. We use the *X-coordinate and Y-coordinate of the centroid* of the foreground blob, since the variations of these over time describes the horizontal and vertical motions of the actor's body as a whole. We also note that, other actions can occur due to the movement of the hands and the legs. To capture the orientations of the hands and legs, we cluster the foreground pixels at 4 different heights of the foreground

Algorithm $TimeSeriesCompare(T1, T2)$
Given two Time-Series $T1$ and $T2$ of length $n1$ and $n2$ we wish to find a and b such that $T_1 \approx a * T_2 + b$

- $x[1] = T2[2] - T2[1]$, $x[2] = T2[3] - T2[2]$, ..., $x[n2-1] = T2[n2] - T2[n2-1]$
- set $x = mean(x)$
- $y[1] = T1[2] - T1[1]$, $y[2] = T1[3] - T1[2]$, ..., $y[n1-1] = T1[n1] - T1[n1-1]$
- set $y = mean(y)$
- set $a = \frac{y}{x}$
- set $T3[1] = a * T2[1]$,$T3[2] = a * T2[2]$,...,$T3[n2] = a * T2[n2]$
- set $b = mean(T1[1], T1[2], \ldots, T1[n]) - mean(T3[1], T3[2], \ldots, T3[n])$
- set $T4 = a * T2 + b$
- for $c = 0 : t$ (t is a parameter)

  $T5[1] = T2[c+1] - T4[1], T5[2] = T2[c+2] - T4[2], \ldots, T5[n2-c] = T2[n2] - T4[n2-c]$
  $diff1[c] = mean(T5[1], T5[2], \ldots, T5[n2-c])$
- end
- for $c = 0 : t$ (t is a parameter)

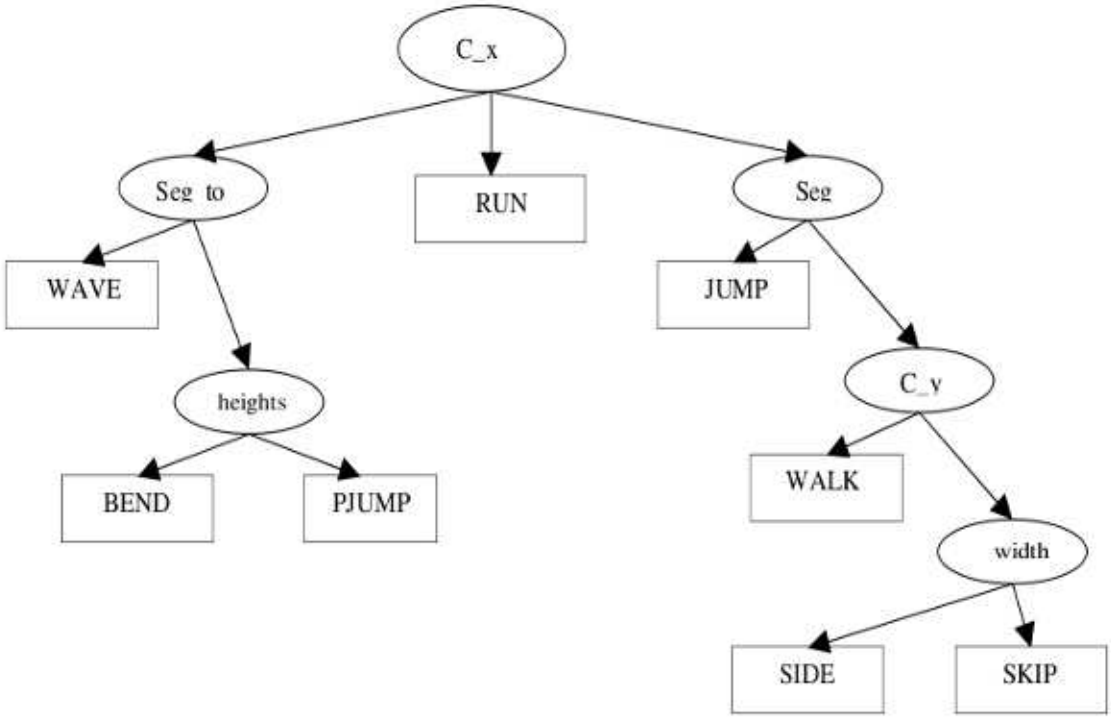
  $T5[1] = T4[c+1] - T2[1], T5[2] = T4[c+2] - T2[2], \ldots, T5[n2-c] = T4[n2] - T2[n2-c]$
  $diff2[c] = mean(T5[1], T5[2], \ldots, T5[n2-c])$
- end
- $diff = min(|diff1[1]|, |diff1[2]|, \ldots, |diff1[t]|, |diff2[1]|, |diff2[2]|, \ldots, |diff2[t]|)$
- return diff

Figure 5.1: Algorithm to compare two time-series

blob. For example, if the actor holds up his hand, there should be two separate clusters of foreground pixel near the top- one cluster corresponding to the head and another corresponding to the raised hand. We call these as the number of *foreground regions* at different heights. Also, we cluster the optical flow vectors (obtained by Lucas-Kanade method) at these same locations, to capture motion of these individual body parts. For this, we again use clustering of the optical flow vectors at different heights, to obtain what we will call *action regions*. We also use the actor's height as a feature. So we have 11 features. As mentioned earlier, we will construct a time-series of each of these features for a video. Now that we have a distance measure between two time-series, we can use a nearest-neighbor classifier to classify time-series. We have a dictionary of known actions, and for each we have a set of prototype time-series for the different features. So, by using nearest-neighbor classification, we can classify the test action video as one of the actions in our dictionary. But now we have 11 different time-series for each video. We weigh the features hierarchically, and build a decision tree accordingly. This weighing of features is on the basis of prominence to the human eye. For example, we give maximum weight to the $x - coordinate$ of the centroid, which is a measure of the motion of the person as a whole. The time-series of this feature allows us to classify between running, walking and staying at the same place. Since all actions in our dictionary are characterized by the features we have defined, we can do successful action recognition using this approach. The decision tree for classification, using different features at different depths, has been shown in 5.2. This tree has been constructed for the 9 actions listed in the Weizzmann Dataset.

5.5 Handling of Occlusions

Occlusions are very common in real world situatons. In such situations, the actor may not be fully visible at all frames, because other people will cross him. In our case, because of occlusions, we may get a time-series with missing values. The challenge is to handle such missing values in the time-series and still perform successful classification. So, we



C_x: X-coordinates of centroid
C_y: Y-coordinates of centroid
Heights: Height of the silhouette
Width: Width of the silhouette at about 80% of height (in the knee region)
Seg: Number of body segments in the knee region (measure of togetherness of legs)
Seg_top: Number of body segments on the scanline of the silhouette's top

Figure 5.2: Decision Tree using the described Features for Weizzmann Dataset

need to estimate the missing values from the remaining data. In time-series literature, the most standard method to do this is to use Kalman Filters [**?**]. But once again, this assumes Gaussian noise, and involves estimation of its parameters. But we choose to keep our methods nonparametric, and so we avoid using models. In this case also, we use the distance measure above. We assume that the time-series is homogeneous, and the distance between overlapping subsequences are very small. So, if x_t is missing, we would want to estimate it with a value x such that the distance between the overlapping subsequences x_{t-n},x_{t-n+1},...,x_{t-1} and x_{t-n+1},x_{t-n+2},...,x_{t-1},x is minimized. Also, the distance between x,x_{t+1},...,x_{t+n-1} and x_{t+1},x_{t+2},...,x_{t+n} should be minimized. To satisfy these two simultaneously, we estimate x to minimize the sum of these two distances. Once we have the estimate, we can use it in the time-series classification task. In case two simultaneous data points are missing, for each of them we look to minimize only one of the two distances mentioned above.

5.6 Detection of Nonhomogeneity

An interesting problem in computer vision is to detect abnormal behavior or event in videos. This problem is extremely important in surveillance. In this work we aim to solve the problem of detection of abnormalities in actions, in which the action is not homogeneous throughout the video. A typical example might be that a person walks for sometime and then suddenly starts running. In case of such actions the nature of the timeseries changes over time. Such actions, referred to as *Nonstationary Actions* have been modelled in [41] using Mixed-State Models, using Viterbi Algorithm to distinguish the different parts. In this work, we use the same idea of comparing subsequences from different parts of the time-series, and for comparsion we once again use our distance measure described earlier.So, we divide the time-series into non-overlapping subsequences. The lengths of these subsequences are equal. We compare pairs of them using our distance measure. For comparison we select pairs of subsequences which are not adjacent to each other, as adjacent subsequences are expected to be close enough by absolute

values even if they have different shapes. So, for each test time-series, we have a vector of distances between different pairs of subsequences. Based on this distance vector, the action can be classified as normal or abnormal. Intuitively, in a time series, if these values are small, the action is homogeneous.

5.7 Results

We have experimented on two datasets that have been widely used for action recognition problems. These are the Weizzmann and KTH datasets. The Weizzmann datasets have 9 different actions performed by 10 different actors. The actions are run,walk,jump,sidewise jump,skip,bend,jump in place,jack,one-hand-wave and two-hand-wave. We could classify the actions with 100 per cent accuracy in almost all cases, except for running which was confused with walking in about 10 per cent cases. The KTH dataset has 6 actions, and we achieved over 90 per cent accuracy in all cases. These results are at par with the best quoted results on these datasets.

Chapter 6

Conclusion and Future Work

In this thesis we have studied Covariance Descriptors for sets of similar images, and attempted to model their structural similarities. We have proposed two models- one using Joint Diagonalization and the other using the concept of Significant Shared Basis. These two models have been used for two different tasks of computer vision, namely face/object recognition and visual tracking of objects. As an aside we have also proposed a heuristic for comparing pairs of time-series, and used it for action recognition.

Some future directions of work on these lines can be

- The Joint Diagonalization-based method can be used for modeling video clips corresponding to a particular person or object, since such clips are also collections of similar images. These video clips can then be clustered using this method.

- The appearance model proposed for objects in the tracking framework, using Central and Peripheral Regions, potentially opens up detection of pose/orientation changes of the target in the track. This can be an interesting addition on the tracking problem.

- A bag-of-words-like representation of images is possible using covariance descriptors. This will involve building multiple covariances for a single image, for different salient regions, as is done in BOW-based algorithms. Handling such a bag-of-covariances can be an exciting research problem.

- New algorithms for estimating common basis vectors for sets of covariance matrices need to be explored as an alternative to Joint Diagonalization. One possibility is to use concepts from Bayesian Matrix Factorization.

Bibliography

[1] A.Ziehe,P.Laskov,G.Nolte,K.Muller (2004) *A Fast Algorithm for Joint Diagonalization with Non-orthogonal Transformations and its Application to Blind Source Separation*, Journal of Machine Learning Reesearch 5, pp 777-800.

[2] D. Pham and J-F Cardoso (2001). *Blind Separation of Instantaneous Mixtures of Nonstationary Sources*, IEEE Transactions on Signal Processing, Vol 49, No. 9, pp 1837-1848.

[3] D. Pham (2001). *Joint Approximate Diagonalization of Positive Definite Hermitian Matrices*, SIAM Journal of Matrix Analysis and Applications, No. 22.

[4] O.Tuzel, F. Porikli, P.Meer (2006) *Region Covariance: A Fast Descriptor for Detection and Classification*, ECCV 2006, pp 589-600

[5] W. Forstner, B. Moonen (1999) A metric for Covariance Matrices, TR, Stuttgart University

[6] M. Turk and A. Pentland (1991). *Face recognition using eigenfaces*. Proc. IEEE Conference on Computer Vision and Pattern Recognition 1991, pp. 586-591.

[7] M.S. Bartlett, J.R. Movellan, T.J. Sejnowski (2002). *Face Recognition by Independent Component Analysis*. IEEE Trans. on Neural Networks, Vol. 13, No. 6, November 2002, pp. 1450-1464

[8] P.N. Belhumeur, J.P. Hespanha, D.J. Kriegman (1996). *Eigenfaces vs. Fisherfaces:*

Recognition using Class Specific Linear Projection. Proc. of the 4th European Conference on Computer Vision, ECCV'96, 15-18 April 1996, Cambridge, UK, pp. 45-58

[9] L. Wiskott, J.-M. Fellous, N. Krueuger, C. von der Malsburg (1997), *Face Recognition by Elastic Bunch Graph Matching*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 19, No. 7, 1997, pp. 775-779

[10] T.F. Cootes, K. Walker, C.J. Taylor (2000), *View-Based Active Appearance Models*, Proc. of the IEEE International Conference on Automatic Face and Gesture Recognition, 26-30 March 2000, Grenoble, France, pp. 227-232

[11] B. Moghaddam, T. Jebara, A. Pentland (2000), *Bayesian Face Recognition*, Pattern Recognition, Vol. 33, Issue 11, November 2000, pp. 1771-1782

[12] W. Zhao, R. Chellappa, A. Rosenfeld, P.J. Phillips (2003) *Face Recognition: A Literature Survey*. ACM Computing Survey, (2003) pp. 399-458

[13] Y. Pang, Y. Yuang, X. Li(2008). *Gabor-Based Region Covariance Matrices for Face Recognition*. IEEE Transactions on Circuits and Systems for Video Technology, Vol. 18 Issue 7 (July 2008), pp 989-993

[14] Li-Hong Zhao,Xi-Li Zhang,Xin-He Xu(2007). *Face recognition base on KPCA with polynomial kernels*. Proc. IEEE Conference on Wavelet Analysis and Pattern Recognition 2007, pp. 1213-1216

[15] B.Leibe and B.Schiele (2003) *Analyzing Appearance and Contour-based Methods for Object Categorization* Proc. CVPR 2003, pp 409-415

[16] G. Csurka, C.R. Dance, L. Fan, J. Willamowski, and Bray C. *Visual categorization with bags of keypoints*. ECCV, 2004.

[17] G. Wang, Y.Zhang, Li Fei-Fei. *Using Dependant Regions for Object Categorization in a Generative Framework*, CVPR 2006.

[18] Manik Varma and D.Roy, *Learning the Discriminative Power-Variance Trade-Off*, ICCV 2007.

[19] J.Saketha Nath, G.Dinesh, S.Raman, C.Bhattacharyya,KR Ramakrishnan, A Ben-Tal, *On the Algorithmics and Applications of a Mixed-norm based Kernel Learning Formulation*, NIPS 2009.

[20] P.Gehler and S.Nowozin, *On Feature Combination for Multiclass Object Classification*, ICCV 2009.

[21] M.Black and A. Jepson *Eigentracking:Robust Matching and Tracking of Articulated Objects using a view-based Representation*, IJCV, 26 (1996), pp 63-84.

[22] D.Ross, J.Lim, M.H. Yang *Incremental Learning for Robust Visual Tracking*, IJCV 77(2008) pp 125-141

[23] D.Comaniciu, V.Ramesh and P.Meer *Kernel-based Object Tracking*, PAMI 25(2003) pp 564-575.

[24] X.Mei and H.Li, *Robust Visual Tracking with L1 minimization*, ICCV 2009.

[25] F.Porikli,O.Tuzel and P.Meer, *Covariance Tracking using Model Update using Lie Algebra*, CVPR 2006

[26] A.Levy and M.Lindenbaum, *Sequential Karhunen-Loeve Basis Extraction and its Application to Images*, IEEE Transactions on Image Processing 9(2000), pp 1371-1374

[27] B.Babenko M-H Yang and S.Belongie, *Visual Tracking with Online Multiple Instance Learning*, CVPR 2009.

[28] *http://www.cs.toronto.edu/ dross/ivt/*

[29] ,L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri *Actions as Space-Time Shapes*, PAMI, December 2007.

[30] Y.Ke, R. Sukhthankar, M.Hebert *Efficient Visual Event Detection using Volumetric Features*, IEEE Computer Society,Volume 1, 2005.

[31] A.Yilmaz and M.Shah, *Action Sketch: A novel action representation*, CVPR, 2005.

[32] A.Efros,A.Berg,G.Mori and J.Malik, *Recognizing Action at a Distance*, CVPR 2003.

[33] J.Niebles and L. Fei-Fei, *A Hierarchical Model of Shape and Appearance for Human Action Classification*, CVPR 2007..

[34] M. Bregonzio, S. Gong and T. Xiang, *Recognising Action as Clouds of Space-Time Interest Points*, CVPR, 2009.

[35] F.Lv and Ram Nevatia, *Single-View Human Action Recognition using Key Pose Matching and Viterbi Path Searching*, CVPR 2007.

[36] P.Turaga,R.Chellappa,VS Subrahmanian, O.Udrea, *Machine Recognition of Human Activities-A Survey*, IEEE TCSVT, Nov 2008.

[37] ,Y.Wang and G.Mori, *Max-Margin Hidden Conditional Random Fields for Human Action Recognition*, CVPR 2009.

[38] F.Lv and Ram Nevatia, *Single-View Human Action Recognition using Key Pose Matching and Viterbi Path Searching*, CVPR 2007.

[39] W.Liao, *Clustering of time series dataa survey*, Pattern Recognition,Elsevier, Volume 38,2005.

[40] H.Isliker,J.Kurths, *A Test for Stationerity: Finding Parts in Time-Series for Correlation Dimension Estimate*, International Journal for Bifurcation and Chaos,1993

[41] N.Cuntoor,R.Chellappa, *Mixed-State Models for Nonstationary Multiobject Activities*, EURASIP Journal for Advances in Signal Processing,Volume 2007

[42] G.Das,K.Lin,H.Mannila,G.Renganathan,P.Smyth, *Rule Discovery from Time Series*, KDD 1998.