This is to certify that the
thesis entitled

LOBE COMPONENT ANALYSIS IN COGNITIVE DEVELOPMENT FOR SPATIO-TEMPORAL SENSORY STREAMS

presented by

RAJA SEKHAR GANJIKUNTA

has been accepted towards fulfillment
of the requirements for the

Master of Science degree in Computer Science and Engineering

Juyang Weng

Major Professor's Signature

2003-8-22

Date

DATE DUE	DATE DUE	DATE DUE

Lobe Component Analysis in Cognitive Development for Spatio-temporal Sensory Streams

By

Raja Sekhar Ganjikunta

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Computer Science and Engineering

2003

Abstract

Lobe Component Analysis in Cognitive Development for Spatio-temporal Sensory Streams

By

Raja Sekhar Ganjikunta

This thesis addresses feature analysis in autonomous mental development. An attempt at general object recognition is made. A need for feature analysis is felt. Fast incremental independent component analysis algorithms are developed. A new domain of feature analysis called *lobe component analysis* (LCA) is introduced. A *collective best match* (CBM) criterion, biologically motivated by sparse coding, is proposed for LCA. An algorithm for LCA is developed and its properties are analyzed. Comparative analysis of LCA, principal component analysis (PCA) and independent component analysis (ICA) is made. The LCA algorithm derived is the fastest among existing incremental ICA algorithms to extract super-Gaussian components. Using LCA, spatial and spatio-temporal filters are derived for natural images and natural image video respectively. The applicability of LCA/ICA for motion segmentation is demonstrated. An architecture for sensory mapping based on the techniques used in LCA is proposed and results are obtained.

To my mom, dad and sister.

ACKNOWLEDGMENTS

I would like to express my sincere thanks to my advisor Dr.Juyang Weng who has given me a new outlook to research through his insight and inspiration. He helped me get a good feel of what research is about and how to do it well. My interactions with him have given me good experience in solving research problems. I am grateful for the time and effort he spent in discussions which were very helpful for me, my mind set and my research orientation. I am very thankful to Dr.George Stockman for his course on computer vision which was both practical and interesting. Also, I would like to thank him for sharing his research books with me. My sincere thanks to Dr.Juyang Weng, Dr.George Stockman, and Dr.Fathi Salem for serving on my committee. I would like to thank Dr.Anil K. Jain for providing me with a strong basis of pattern recognition concepts through his course.

I would like to thank my lab mates Nan Zhang, Yi Chen, Xiao Hwang, David, Shuqing Zeng, Yilu, Ameet and Micky for many open and useful discussions and also for the fun we had. I consider myself very lucky to have good friends - Mahesh, Narsimhan, Chandan, Aravindan, Loga, Prasanna and Shankar who were there to share my joy and provide comfort and support during hard times. I would also like

to thank my lab mates in this regard. I wish to thank some of the PRIPies - Arun Ross, Umut Uludag, Anoop for the discussions we had.

I would like to thank the CSE department and Dr.Jon Sticklen for having provided me with financial aid to continue my research while enjoying teaching. It was fun interacting with the CSE131 kids. I am thankful to my professors Dr.Anthony Wojcik, Dr.Sandeep Kulkarni, Dr.Betty Cheng and Dr.Abdol Esfahanian for their interesting courses which gave me good insight into various fields of computer science.

There are no words that could say how much my dearest parents and sister have helped me in all that I have become. Their constant words of encouragement, their belief in me and their prayers served as an endless source of motivation for everything that I had undertaken in my journey through life. I am eternally indebted for having such great parents and a caring sister.

Table of Contents

LIST OF FIGURES ix

LIST OF TABLES xi

1 Introduction 1
1.1 What is general object recognition? . . . 2
1.2 Challenges in general object recognition . . . 3
1.3 SAIL robot project . . . 4
1.4 My research path . . . 5
1.5 Thesis outline . . . 6

2 Background 8
2.1 Autonomous mental development . . . 8
2.1.1 Development versus learning . . . 9
2.1.2 Autonomous mental development paradigm . . . 10
2.1.3 Architecture of an autonomous mental development system . . . 12
2.1.4 Sensory mapping . . . 13
2.1.5 Cognitive mapping . . . 14
2.2 Feature extraction . . . 16
2.2.1 Principal component analysis . . . 17
2.2.2 Independent component analysis . . . 18
2.2.3 Linear discriminant analysis . . . 18
2.2.4 Spatial and spatio-temporal features . . . 19
2.3 Object recognition . . . 19
2.3.1 Eigen faces and Fisher faces . . . 20
2.3.2 Developmental methods for object recognition . . . 21
2.4 Optimization methods . . . 21
2.4.1 Stochastic gradient methods . . . 21
2.4.2 Newton-Raphson method . . . 22
2.4.3 Fixed-point iteration . . . 23
2.4.4 Constrained optimization . . . 24

3 Some Issues Faced in Computer Vision 25
3.1 Threshold finding problems . . . 25
3.2 Space-time constrained algorithms . . . 26
3.3 Single modality for autonomous mental developmental ? . . . 27

4 General Object Recognition: Missing Label Problem **29**
4.1 Introduction 30
4.2 The basic model of an agent 32
4.3 Architecture for the missing label problem 34
4.4 Dense label problem 36
4.4.1 Experimental setup 37
4.4.2 Results 37
4.4.3 Analysis 38
4.5 Sparse label problem 40
4.5.1 Experimental setup 41
4.5.2 Results 42
4.5.3 Analysis 43
4.6 Conclusion 44

5 New Incremental Independent Component Analysis **45**
5.1 Independent component analysis 47
5.2 Existing methods for independent component analysis 55
5.2.1 FastICA 55
5.2.2 Extended Bell-Sejnowski algorithm 57
5.2.3 Nonlinear PCA (NPCA) 58
5.2.4 Gradient algorithms 59
5.3 Incremental methods developed 60
5.3.1 Amnesic gradient ascent of the square of kurtosis 60
5.3.2 Incremental fixed point on kurtosis 64
5.3.3 Incremental fixed point on negentropy 65
5.3.4 ccNegIICA 66
5.4 Results and analysis 67
5.4.1 Measuring convergence error 68
5.4.2 Uniform distribution - sub-Gaussian 69
5.4.3 Laplacian distribution - super-Gaussian 71
5.5 Conclusion 72

6 Lobe Component Analysis **75**
6.1 Introduction 76
6.2 Lobe Component Analysis (LCA) 81
6.2.1 Illustration 82
6.2.2 Measure of expressiveness: collective best match criterion 86
6.2.3 LCA algorithm 87
6.2.4 Discussion of the LCA algorithm 88
6.2.5 Intuitive proof of expressive power of the algorithm 89
6.3 Properties of LCA 92
6.4 PCA and ICA - special cases of LCA ? 97
6.5 Results with PCA 98
6.5.1 Dimensionality reduction - PCA vs LCA 99
6.5.2 Clustering experiment on faces 103

6.5.3 Data Compression 106
6.6 Results with ICA 108
6.6.1 LCA can extract super-Gaussian independent components 108
6.6.2 LCA filters for whitened and non-whitened natural images 110
6.6.3 Over-complete basis estimation 113
6.7 Conclusion 114

7 Motion Features and Feature Maps **116**
7.1 Applicability of LCA for motion segmentation 117
7.2 Motion filters in natural video 118
7.2.1 Variation of eigen values in natural video block 122
7.3 Self organizing feature map 124
7.3.1 Spatio-temporal SHM 126

8 Conclusion **131**
8.1 Summary and Contributions 131
8.2 Future work 133

List of Figures

2.1 The internal architecture of an agent consists of sensory mapping, cognitive mapping, motor mapping and a value system 13
2.2 General architecture of a recognition system. 19

4.1 Basic model of an agent. 33
4.2 Basic architecture of the SAIL robot. (Adapted from [53]) 35
4.3 Double clustering in both input and output spaces. (Adapted from [53]) 36

5.1 Comparison of analytical distributions. 51
5.2 Amnesic gradient ascent of $kurt^2$ for sub-Gaussians. (a) 5-D b) 10-D. . . 70
5.3 Fixed point on kurtosis for sub-Gaussians. (a) 5-D b) 10-D. 70
5.4 Fixed point on negentropy for sub-Gaussians. (a) 5-D b) 10-D. 70
5.5 ccNegIICA for sub-Gaussians. (a) 5-D b) 10-D 71
5.6 NPCA for sub-Gaussians. (a) 5-D b) 10-D. 71
5.7 Amnesic gradient ascent of $kurt^2$ for super-Gaussians. (a) 5-D b) 10-D. . 72
5.8 Fixed point on kurtosis for super-Gaussians. (a) 5-D b) 10-D. 73
5.9 Fixed point on negentropy for super-Gaussians. (a) 5-D b) 10-D. 73
5.10 ccNegIICA for super-Gaussians (a) 5-D b) 10-D. 73
5.11 NPCA for super-Gaussians (a) 5-D b) 10-D. 74

6.1 Recognition under occlusion. 77
6.2 Illustration of LCA versus PCA and ICA. 82
6.3 Illustrating the expressiveness of LCA derived features. 84
6.4 Mean of ϕ and its variance for LCA and PCA for dimensionality reduction. 100
6.5 Mean-square error (MSE) results for PCA and LCA. 102
6.6 Clustering Error for PCA and LCA on FERET Data set. 104
6.7 Eigen faces and LCA faces 105
6.8 Error in reconstruction for PCA and LCA for data compression. 107
6.9 Convergence speed of LCA versus existing adaptive ICA algorithms. . . . 110
6.10 LCA derived features for natural images with pre-whitening. 111
6.11 Number of hits for the LCA features for natural images. 112
6.12 Distribution of Eigen values for a set of natural images. 112
6.13 LCA derived features for natural images without pre-whitening. 113

7.1 Motion segmentation using LCA and ICA. 119
7.2 LCA filters for frame 1 of the video block. 121
7.3 LCA filters for frame 2 of the video block. 121

7.4 Motion filters for the video block. 122
7.5 Number of Eigen values to be considered for 95% variation. 123
7.6 Eigen variation for 512 dimension natural video blocks. 123
7.7 Filters for the first neural layer in sensory mapping architecture. 125
7.8 Sensory mapping filters for 16 neurons per RF of the retina. 127
7.9 Motion Filters for spatio-temporal SHM 128
7.10 Filters for frame 1 of spatio-temporal SHM 129
7.11 Filters for frame 2 of spatio-temporal SHM 130

List of Tables

2.1 Differences between learning and development. (Adapted from [9]) 9

4.1 Recognition results for the dense label problem 38
4.2 Percentage error for the dense label problem. 38
4.3 Recognition results for the sparse label problem 42
4.4 Percentage error for the sparse label problem. 43

7.1 Sensory mapping parameters using 1 neuron per RF of the retina 125
7.2 Sensory mapping parameters using 16 neurons per RF of the retina . . . 126
7.3 Spatio-temporal SHM parameters for 16x16 neurons. 129

8.1 Contributions of the thesis. 133

Chapter 1

Introduction

There has been an enormous advancement in the field of computing over the past 50 years. Computing in a general sense has moved towards making life more elegant but simple. The way humans interact with computers is growing more natural day after day. The day is approaching when science fiction is no longer in our imagination, when robots interact with humans just like humans interact among themselves. Towards this dream, one of the important problems that needs to be addressed is the problem of *general object recognition* that deals with how to make a computer or robot recognize any general object. More generally, we would like to have a robot that can learn new objects and develop its skills based on the experience it gains from the environment. The domain of autonomous mental development addresses this issue of development and provides a promising base upon which general object recognition can be realized. Initial attempts to use this *developmental base*, provided by autonomous mental development, to address the problem of general object recognition resulted in a need for attention selection in unconstrained environments. For attention selection,

we needed good feature analysis techniques for the sensory mapping, which is a part of the developmental architecture [49].

More specifically, this thesis is composed of three important and interlinked parts - one, an attempt to address the problem of general object recognition, two, developing better feature analysis techniques for use by the *developmental base* for general object recognition, and three, application of the new feature analysis technique to built an architecture for the sensory mapping.

The next few sections discuss what general object recognition is, the challenges in the field, some information on the SAIL robot project on which the developmental base is being developed and tested, my research path and the outline of the thesis.

1.1 What is general object recognition?

Computer vision is a field that tries to extract meaning from an image or understand an image or a sequence of images. Object recognition is an important part of computer vision that focuses on recognizing objects from images. The methods that have been proposed for object recognition can be broadly classified into many categories such as - appearance based, feature based, template based, view based and texture based methods to name a few. Among these, appearance based methods have become popular because of their capability to address general recognition problems, and because of the development of automatic feature extraction techniques such as principal component analysis (PCA), independent component analysis (ICA), etc. The commonalty between the various recognition approaches is that they use classifiers.

Classifiers are algorithms that map input to output classes. The output classes are representative of the objects that are to be recognized. Some of the sub-domains of object recognition are face recognition, gesture recognition, etc. Each of these sub-domains has restricted a input space, which is a set of faces for face recognition and a set of gestures for gesture recognition. Each of these problems is considered to be hard because of various reasons like - high dimensionality of input space, segmentation problem, etc. A more general problem is how to make a computer recognize general objects. This problem is called *general object recognition.*

1.2 Challenges in general object recognition

The problem of general object recognition is much harder than the specific problems of face or gesture recognition. Some of the challenges in general object recognition are:

1. We cannot model each and every kind of object that is present in the world. So, we need a model-less approach to solve the problem of general object recognition.

2. As we cannot model objects by hand, we cannot use or extract programmer-defined features in the classification task.

3. We have to resort to using the raw image data or the pixel level information or any function of it, that minimally loses raw information, as input to the classifier. Appearance based methods are used to address this problem.

4. If we consider each pixel as a feature, we enter a high-dimensional space of 10,000 features for a simple 100x100 gray scale image set.

5. With such a large number of features comes a lot of issues that need to be addressed. The curse of dimensionality comes into play with such a large number of features.

6. As we cannot tell the system the list of all objects it is going to see before-hand, we should have a method that lets a teacher indicate to the system the various objects as part of the training. This brings us to an interactive system such as a robot, which can learn from the interactions of a teacher. The SAIL robot, at the EI-lab of Michigan State University, serves as a platform to carry out experiments on general object recognition using a developmental architecture.

1.3 SAIL robot project

SAIL stands for Self-organizing Autonomous Incremental Learner. SAIL is the name given to the robot, which was built in the Embodied Intelligence Lab under the guidance of Dr.Juyang Weng at Michigan State University. The robot is of a new genre in the sense that it is motivated by a developmental paradigm. It is intended to develop like a human child. The line of work on the developmental robot is funded in part by NSF, DARPA, Microsoft Research, Siemens Corporate Research, and Zyvex. The first step is to give it some functionality like a human child, viz., the freedom of motion, vision, audition and touch. The robot was built with an arm, which is

capable of motion, two eyes in the head that can pan and tilt, and a head that rotates around a vertical axis. The robot has a motor base with four wheels for locomotion. A computer attached to SAIL's body is used to deploy the developmental algorithms. A microphone is used to obtain speech input from the environment. Apart from the computer program, a joystick can also be used to control the motion of SAIL. There are many touch sensors on the arm of the robot, which can be read by a computer program. Developmental work on the robot is motivated by the autonomous mental development methodology.

1.4 My research path

The problem of general object recognition in a developmental framework was one of the main long-term goals of the SAIL robot project. Along this direction, a literature survey of autonomous mental development was done. Attempts were made to make the robot recognize general objects using a developmental framework. Experiments were performed to address general object recognition through two experiments on the dense label problem and the sparse label problem. From the experiments, a need for attention selection mechanism was realized. It was also realized that to perform attention selection in the autonomous mental development framework, spatial and spatio-temporal features had to be extracted. So, feature extraction techniques were analyzed and developed. New incremental independent component analysis methods were derived and tested. Later, a new feature analysis domain called lobe component analysis (LCA) was proposed by us. An adaptive method for the new

feature analysis domain was derived based on a biologically motivated criterion. The properties of the new technique were compared to those of PCA and ICA. Many of the technique's desirable features were analyzed. Using this new technique, spatial and spatio-temporal features were extracted from natural images and natural image sequences respectively. A self organizing hierarchical feature map using the LCA techniques for sensory mapping based on the existing staggered hierarchical mapping (SHM) architecture is proposed.

1.5 Thesis outline

Chapter 2 presents the background concepts and related work for the thesis. Chapter 3 discusses some of the problems that exist in the current methods of research in computer vision. Chapter 4 presents my attempts and results for general object recognition using the developmental architecture. The experiments conducted presented a need for attention selection using spatial and spatio-temporal feature extraction. So we shift our attention to feature analysis techniques in Chapter 5. There, we derive faster incremental algorithms for independent component analysis. In Chapter 6 we introduce a new domain of feature analysis techniques called *lobe component analysis* (LCA) which is biologically motivated. We also present results comparing LCA with PCA and ICA. In Chapter 7 we develop a hierarchical self-organizing feature map as an architecture for sensory mapping and derive features for staggered receptive fields so that when attention signals for a region are presented to the sensory mapping, it can deliver the feature representation for that region. Also, we demonstrate the use

of traditional ICA for motion tracking of objects moving in a changing background. Chapter 8 concludes the thesis with a discussion of contributions made and future work.

Chapter 2

Background

This chapter gives a review of the concepts that served as a background for the work carried out in the thesis. We start with an overview of the concept of Autonomous Mental Development, the overall architecture and a short introduction to the major components of the architecture in Section 2.1. We present some of the popular methods for feature extraction in Section 2.2, followed by a discussion of the existing approaches for object recognition in Section 2.3. Section 2.4 reviews of some of the optimization techniques used in the thesis.

2.1 Autonomous mental development

In his words Weng [9] describes development as:

> "With time, a brain-like natural or an artificial embodied system, under the control of its intrinsic developmental program (coded in the genes or artificially designed) develops mental capabilities through autonomous

Table 2.1: Differences between learning and development. (Adapted from [9])

Property	Learning	Development
Task specific	Yes	No
Tasks are unknown	No	Yes
Generates a representation of an unknown task	No	Yes
Animal-like online learning	No	Yes
Open-ended learning	No	Yes

real-time interactions with its environments (including its own internal environment and components) by using its own sensors and effectors."

Autonomous Mental Development is a new methodology in the field of artificial intelligence in which an artificially embodied system develops like a child using a developmental program that it is "born" with. This developmental program can be thought of as the "developmental algorithm" in human genes. The robot develops just like a human child would develop - from its interactions with the environment.

2.1.1 Development versus learning

Development is distinguished from traditional learning. In learning, the programmer programs the computer to perform a particular task or in a particular domain as in a expert system. The system developed is limited by the amount of knowledge it is built with and the domain it can operate in. It is not extensible to other domains and cannot be used, without reprogramming, to do other tasks for which it is not designed for.

In the case of development, as opposed to learning, an *embodied system* is programmed with a developmental algorithm to learn a broad range of tasks which are unknown at the time of designing the system. Moreover, the system should not be

built with any task-specific information. Some of the characteristics that are innate to the system like pain, hunger and good/bad rewards can be built into the system, as they are thought to be innate characteristics that a human being is born with. The embodied-system or the robot can be trained by a teacher who does not have access to the internal representation of the system after the system's "birth." The robot learns from its interactions with the environment (including the teacher) guided by a developmental program. Table 2.1 [9] distinguishes development and learning.

2.1.2 Autonomous mental development paradigm

For dealing with muddy tasks that are easy for humans but are tough for today's computers, we need a new methodology for systems to learn and develop like humans do. This methodology requires a fundamental change in the way of thinking and, hence, is a new paradigm. This paradigm is called the Autonomous Mental Development (AMD) paradigm [47] motivated by biological mental development. The AMD paradigm enables an embodied system to develop its "mind" autonomously. The paradigm's main idea is as follows [49]:

1. **Designing a robot body:** We need a robot that looks like a human and can interact with its environment using its sensors and effectors. We need to provide as much freedom to the robot as a human has in order to have minimal limitations in its freedom of motion and interaction with the environment.

2. **Designing a developmental program:** The developmental program can be thought of as all that is innate in human beings at the time of birth, and

that which enables the humans to develop their mind by interacting with the environment. Development of such a program is tough and is the central part of the paradigm. This developmental program should be able to learn new tasks, which it has not been designed for, through interactions with the environment.

3. **Birth:** This is when the developmental program starts to run on the robot body, thereby enabling it to develop a "mind" by interacting with its environment.

4. **Developing mind autonomously:** This part of the paradigm refers to the online interactions that enable the developmental program to learn new tasks and to gain experience driven by the developmental program. A human operator can give rewards to the robot through the sensors and teach it commands or sensations and their corresponding actions. The robot should be able to associate sensation with action and learn autonomously without requiring a change in its developmental program once it is "born."

We adopt the above AMD paradigm to address the problem of general object recognition.

Challenges in autonomous mental development paradigm

Some of the technical challenges that make designing and implementing algorithms conforming to the AMD paradigm tough are as follows:

- The ability to deal with high dimensionality input from the sensors.
- Real-time interaction with the environment needs real-time processing in high-dimensions. So, algorithms with high space or time complexity are not feasible

to be incorporated into the developmental programs.

- The need for real-time processing enforces another constraint on the algorithms, which is that they must be incremental. Batch processing is time consuming and is infeasible for processing large amounts of high dimensional data in real-time for even the computer speeds that could be conceivable in the next few decades.

2.1.3 Architecture of an autonomous mental development system

Towards realizing developmental algorithms, we need to understand the major parts of a system. The system as a whole can be thought of as an agent in an external environment. The agent senses the environment using its external sensors; senses its internal representation using its internal sensors and performs internal or external actions using its internal or external effectors. The internal brain or the neurological system is responsible for all the internal processing. The conversion of the external signals by the sensors to the neurological signals is done by the sensory mapping. The cognitive mapping is the neurological center which is responsible for higher-level processing of the signals which it gets from the sensory mapping. The motor mapping converts the neurological signals to signals for the effectors. A diagram of the internal architecture of an agent is shown in Figure 2.1.

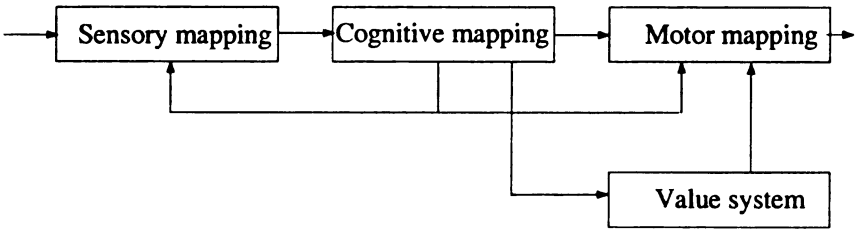


Figure 2.1: The internal architecture of an agent consists of sensory mapping, cognitive mapping, motor mapping and a value system

2.1.4 Sensory mapping

Definition The *sensory mapping* is an architecture that develops a high level representation of the sensory signals received from the external world.

The sensory mapping for vision can be thought of as an architecture which derives representation for a group of receptive fields of various sizes at various positions on the retina. Sensory mapping for vision takes, as input, the raw image and gives, as output, the signals from the receptive fields. This output goes to the cognitive layer which processes these signals.

The motivation for sensory mapping comes from the work of David H. Hubel and Torsten N. Wiesel on receptive fields in a cat's visual cortex [16] that led to their Nobel Prize. They reported that visual cortex contains specialized cells, analogous to feature detectors that extract features in their *receptive field.* The sensory mapping extracts features from different receptive fields and converts the external real-world signals to internal *neurological signals.*

What are the major functions of the sensory mapping?

The major functions of the sensory mapping include the following.

1. Grouping input signals from different sensory modalities like vision, audition, touch in space and time.

2. Automatically derive features in the combined space of different modalities as a new representation (*neurological signals*).

3. Be able to selectively deliver neurological signals for inter-modal or intra-modal attention signals received from higher layers. This suggests that the sensory mapping should be able to derive *staggered* [51] representation for receptive fields of various sizes at various positions on the retina.

The features derived would be in reduced dimensionality so as to mitigate the curse of dimensionality. The sensory mapping's input is raw high-dimensional signals. Assuming an image size of 100×100, the spatial dimensionality becomes $10,000$, and to operate in such high-dimensions in real-time is a very hard task without reducing the dimensionality. Also, spatio-temporal features have to be estimated for the robot to perform attention selection. Assuming that we keep track of the latest 10 frames, the spatio-temporal dimensionality becomes $100,000$.

2.1.5 Cognitive mapping

The sensory mapping transforms the raw data into what we call *neurological signals* for processing by higher layers. We group the higher layers under *Cognitive Mapping.* This is the actual *"brain"* of the robot that serves to learn autonomously a mapping between input (images, sound, etc.) and output (actions, speech, etc.).

Definition The *goal of cognitive mapping* is to establish a mapping $f : X \rightarrow Y$, from the input space X to the output space Y. Input from the sensory mapping arrives incrementally in real-time and the corresponding action for the input sample can either be *imposed* by a teacher or *primed* by an agent.

The cognitive mapping realizes the basic functions of memory, recall, and performs the function of development. It automatically generates *discriminating features* derived incrementally as and when the input comes from the sensory mapping and from the robot's effectors as feedback.

Symbols are not allowed

When we talk about discriminating features, the existence of an output space (like class labels) is assumed. Having class labels for output denotes the existence of symbolic information. Presence of symbolic information denotes that the output space is defined at the time of designing the robot and this implies task-specificity. But, we need a task non-specific architecture, which implies that symbolic information is not allowed. So, we cannot use classifiers that operate on symbols. Instead, we can use regression approaches and convert the classification problem to a regression problem, converting the class labels into actions which the robot can perform. The output space is, thus, represented by an action vector which can be interpreted as having class label information.

To build an architecture for the cognitive mapping we need to have a mechanism for memory storage and recall. The experience of the robot is saved in the memory storage in the architecture. When new sensory input comes in, the robot recalls its

experience in a similar *context* it had seen earlier and delivers an action based on its previous experience. To make the robot run in real-time, we need a fast recall mechanism. So, we resort to using a tree data structure to speed up the search. Moreover as we have a unified classification and regression problem at hand, we have to use a discriminating regression tree. The *Hierarchical Discriminant Regression (HDR) tree* has been proposed [17] as a solution for the cognitive mapping. The HDR tree forms the core of the cognitive mapping. The cognitive mapping architecture, based on HDR, provides mechanisms for storage, recall and development. Q-learning [42] is one of the popular mechanisms for learning used in this architecture. The reader is referred to [17] [53] for more information on the HDR tree classifier.

2.2 Feature extraction

Feature extraction mechanisms provide a means to extract meaning out of a data set. Most of the feature extraction mechanisms can be considered to be transformations of the data. In other words they find different means of representing the data. For many reasons like ease of use, computational complexity and sufficiency, linear transformations have been popularly used. Some examples of popular linear transformations are Principal Component Analysis (PCA), Independent Component Analysis (ICA), and Linear Discriminant Analysis (LDA). Each of these techniques extracts features based on different criterion functions. PCA and ICA do not use class label information and are used in unsupervised domains, while LDA finds the set of features which best separates the data based on the class information. PCA extracts

the most varying orthogonal set of features, while ICA extracts independent features from a given data set. HDR also extracts features. It extracts *Most Discriminating Features* (MDF) [17]. In this section, we will take a quick look at each of these feature extraction techniques and comment on spatial and spatio-temporal features.

2.2.1 Principal component analysis

The goal of PCA is to derive features that best represent the data in the least square error sense for reduced dimensions. The criterion function used is the least root mean square error (RMS).

$$J = \sum_{i=1}^{n} \|(V'(V')^T x_i - x_i)\|^2 \tag{2.1}$$

where x_i's are the zero mean sample vectors in d dimension and V' is a linear transformation of the sample vectors from d dimensional space to $d' \leq d$ dimensional space. We denote samples x_i by column vectors. Note that the term $(V')^T x_i$ is the new representation of x_i in the reduced dimension d' and $V'(V')^T x_i$ represents a reconstruction of the sample from its representation in the reduced dimension.

Minimizing this criterion, we can derive that V' is a matrix with $d' \leq d$ columns of the eigen matrix V of the covariance matrix of x. Moreover, the columns of the matrix V' correspond to the eigen vectors that have the highest eigen values. The features derived by PCA are given by the columns of the matrix V.

2.2.2 Independent component analysis

The criterion function for independent component analysis (ICA) is *statistical independence of the resulting representation of the samples.* This statistical independence is measured by using non-Gaussianity. The justification of such a measure is due to the central limit theorem. In general, if x is the given data, the features derived by ICA is given by the matrix W

$$W^T X = W^T AS = \text{directions of independent components} \tag{2.2}$$

where S is the underlying set of statistically independent components, and A is the mixing matrix. The reader is referred to Chapter 5 for more details on ICA.

2.2.3 Linear discriminant analysis

Linear Discriminant Analysis or LDA is a supervised feature analysis technique which uses class label information. The criterion used is:

$$J(W) = \frac{W^T S_B W}{W^T S_W W} \tag{2.3}$$

where S_B is the between class scatter and S_W is the within class scatter and W is the set of features which LDA derives upon maximizing the criterion function $J(W)$. LDA maximizes the between-class variation while minimizing the within-class variation. The reader is referred to [8] for the derivation of LDA from the criterion function.

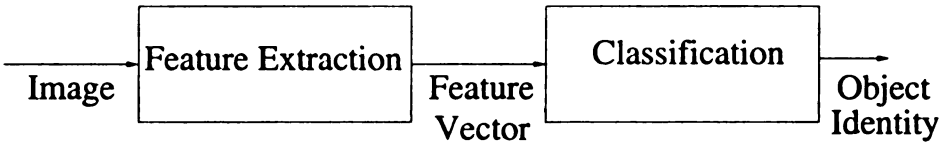


Figure 2.2: General architecture of a recognition system.

2.2.4 Spatial and spatio-temporal features

The feature extraction techniques derived above can be used to extract useful features for numerous applications. The statistics of natural image scenes have been studied due to growing interested in biologically motivated methods to develop feature extractors that derived features similar to those observed in the brain of living animals like cats [16] and monkeys [16, 5]. Spatial features generally refer to the features extracted by a particular feature extraction technique or architecture for images. Spatio-temporal features refer to the features extracted for video blocks, which have time information in each sample. Generally natural images and natural image video are used for analysis involving spatial or spatio-temporal features. We analyze such features later in the thesis in Chapters 6 and 7

2.3 Object recognition

The general architecture of an object recognition system looks like the one shown in Figure 2.2

The features used in general object recognition can be specified manually or automatically. Automatic feature extraction from images is done using the feature extraction methods discussed in Section 2.2. Appearance-based methods are those

that take images as input and extract information from the images. Such methods usually use automatic feature extraction mechanisms to process the data. Most of the methods reduce dimensionality of the data using PCA as the first step of preprocessing. In this section we will not deal with classification methods as it is beyond the scope of this work. Next, we will discuss the types of feature extraction mechanisms used in traditional object recognition systems.

2.3.1 Eigen faces and Fisher faces

Eigen faces uses PCA to extract the top few eigen vectors from a given database of faces. These eigen vectors are also called eigen faces. The face data is projected from high-dimension into the subspace represented by the set of these eigen faces. The resulting vector in this subspace is taken as the feature vector for input into classifiers. One example of a simple classifier is the nearest neighbor classifier.

The new representation used for the face samples is defined by the set of eigen faces. Such a representation will not be useful for other objects such as toys. So, this method is model-specific. In the autonomous mental development paradigm, we cannot use model specific methods.

Fisher faces are the feature vectors derived by Fisher linear discriminant analysis (LDA) discussed in Section 2.2.3. This uses class specific information to compute the between class and within class scatter matrices. Fisher faces overcomes the problem faced by eigen faces as it can represent many kinds of objects in a recognition experiment. But, all the class information must be known before hand for such a

computation of Fisher faces. This is not practicable for a developmental robot which should be able to learn many classes unknown at the time of the design of the robot.

2.3.2 Developmental methods for object recognition

Developmental methods for object recognition should be incremental so that as each sample comes in the representation of the features is updated to improve the recognition rate. They should not use any model-specific information and should be flexible enough to train the system on many different kinds of objects. Incremental HDR [44] is one such method for general object recognition. In this thesis, we will use HDR to refer to the incremental version of HDR unless specified otherwise. We used HDR for general object recognition experiments in Chapter 4.

2.4 Optimization methods

To derive feature extraction algorithms we need two things - one, a criterion, and two, an optimization technique to optimize the criterion. In this section, we will review some of the optimization techniques used later in the thesis.

2.4.1 Stochastic gradient methods

The stochastic gradient methods generally converge linearly to a solution. In these methods a criterion function $F(w)$, which is to be maximized or minimized, is taken. The derivative of the function, or the gradient $\nabla F(w)$ is found. To find the minimum

of the function $F(w)$, we use equation 2.4:

$$w_{k+1} = w_k - \eta_k \nabla F(w_k),\ k = 0, 1, ... \tag{2.4}$$

where η_k is called the *learning factor* or the *step size* which can be a function of the number of steps k. The choice of step size can be such that $\eta_k = c$: a constant, or it can be a diminishing step size such that $\eta_k \to 0$ but satisfies the infinite travel condition, $\sum_{k=0}^{\infty} \eta_k = \infty$. Gradient ascent methods find the maxima and have the following learning rule:

$$w_{k+1} = w_k + \eta_k \nabla F(w_k),\ k = 0, 1, ... \tag{2.5}$$

A faster gradient algorithm uses the second order derivative instead of the learning rate and is presented next.

2.4.2 Newton-Raphson method

Newton-Raphson's (or Newton's) method converges faster than the gradient methods discussed above. But the problem with this method is that it may not converge when started far from a nonsingular local minima. Generally, there is a tendency of using stochastic gradient methods to estimate a good guess for starting the Newton's iteration for convergence. The Newton's iteration is given in Equation 2.6, where w_0

is an initial guess.

$$w_{k+1} = w_k - (\nabla^2 F(w_k))^{-1} \nabla F(w_k),\ k = 0, 1, ... \tag{2.6}$$

In general, the Newton's iteration to solve any non-linear equation $G(w) = 0$ is given in Equation 2.7

$$w_{k+1} = w_k - \frac{G(w_k)}{\nabla G(w_k)},\ k = 0, 1, ... \tag{2.7}$$

Generally, the rate of convergence for Newton's method is quadratic. Moreover, there is no choice of the learning rate involved, unlike gradient methods. Newton's method is an iterative optimization technique and the iteration can stop once the solution converges within a particular tolerance. Newton's iteration formula can be derived from the Taylor expansion of a function.

There are other methods which form the super-Newton family which converge faster (cubic, quadric rates of convergence), but require the function to be differentiable to the third, fourth degrees.

2.4.3 Fixed-point iteration

In fixed point iteration, we try to find the solution to equation 2.8:

$$F(w) = w \tag{2.8}$$

A solution to such an equation is called a *fixed point* of the function $F(w)$. The general technique to solve this problem is as follows. Choose an initial guess w_0 and

find the next estimate w_{k+1} from w_k as follows:

$$w_{k+1} = F(w_k), k = 0, 1, ... \tag{2.9}$$

If the sequence converges, then a fixed point is obtained and the equation 2.8 is solved.

2.4.4 Constrained optimization

For constrained optimization problems of the form given in Equation 2.10, we use the Lagrange multiplier method.

$$\text{minimize } F(w), \text{ subject to } h_i(w) = 0, i = 1, 2...m. \tag{2.10}$$

The Lagrange multiplier theorem boils down to solving equation 2.11 instead of solving the constrained optimization problem given in Equation 2.10.

$$\nabla F(w) + \sum_{i=1}^{m} \lambda_i \nabla h_i(w) = 0 \tag{2.11}$$

where λ_i are called the Lagrange multipliers. We use this technique to solve constrained optimization problems, the likes of which are abundant in the methods used for ICA.

For more details on optimization and numerical analysis techniques, the reader is referred to [28]

Chapter 3

Some Issues Faced in Computer Vision

This chapter discusses some of the issues which the author would like to address with his ideas. The problems or suggestions presented here can be thought of as a by-product while trying to address some of the research problems in Autonomous Mental Development and Computer Vision.

3.1 Threshold finding problems

The problem of having a discrete computer to simulate seemingly analog realities, like human mental development, brings with it the problem of having thresholds. This problem is also common in cases where we would have to make decisions of where to terminate our search for a solution. The term "search for a solution" is general and could mean finding what action to choose, or how accurate we want our solution, or

a simple gradient search of a function, etc. Wherever we have numerical constants, which are not well justified and are dependent on the data, we need to have algorithms for finding those thresholds. Some solutions to threshold finding problems have been proposed. An example would be to find the threshold so that we have the least error, based on some empirical runs. For autonomous mental development, the problem is still aggravated as the data could follow any distribution. For such cases, we would be better off with making the assumption of a Gaussian distribution unless we have reasons not to make that assumption. Using this assumption we should be able to derive stopping criterion based on how much time and space we have to compute the solution. The threshold should be dependent on such constraints. The space-time constraints are discussed next.

3.2 Space-time constrained algorithms

We live in a world constrained by time and space. Time doesn't wait for us and we cannot remember each and every detail we see in a scene, even though we may have seen it many times. There is a need to develop algorithms which adapt to the constraints of space and time, i.e., which give results no matter how limited time and space are. The computer algorithms should be such that they should be able to work with these two constraints so that if we have many algorithms, which are constrained based on their importance, they can run for the time and in the space based on their importance. We cannot delay the output of an algorithm and have a backlog to achieve a particular threshold set for a particular algorithm. We

would have to give the best results calculated so far. For example, we may have a developmental algorithm for speech, and a separate one for vision, and another one for memory, etc., but there are only a few things which we will be able to do based on the time constraints. The system will have to make a choice of which modalities to pay attention to and by how much. Even though many tasks run in parallel, we should be able to give the output and process how much information we could based on the constraints of time enforced by the existence of the system in the real-world.

3.3 Single modality for autonomous mental developmental ?

Instead of thinking of an agent as having many sensors and effectors, the agent could be thought of as a neurological system that has "add-ons" - sensors that convert the external signals to neurological signals and effectors that convert neurological signals to external signals. The processing can then be based on input and output which are both neurological signals. What we perceive to be seeing images and hearing sounds are actually sensations which our neurological system gives to us. The neurological system is responsible for all sensations and actions. If we can convert all of the sensations into a common neurological sensation, we can then talk about only one modality of sensation and action. *It is true that when we close our eyes, we can see vivid images in our dreams, experience fear, and other emotions.* Though these sensors seem external, the sensation is only being produced by the human neurological

system or the mind. The effect of *anesthesia* can be taken as a classic example of how all the sensations are seamlessly integrated into the neurological system and how the neurological system's inhibition will inhibit our senses . Instead of trying to focus on processing the high-dimensional input from images and sound, we can focus on the transformation of those external signals to neurological signals and the processing of those neurological signals to produce other neurological signals or make temporary or permanent neurological records in the brain.

Chapter 4

General Object Recognition: Missing Label Problem

This chapter presents the results for a developmental architecture used to address the problem of general object recognition. The architecture is based on the autonomous mental development paradigm for mobile robots. A developmental robot learns from its interactions with its environment which includes the reinforcement signals given by a teacher. The robot constantly samples the video it receives through its eyes and learns from its input signals from the environment. Evidently, we cannot have a strict supervised learning situation, where each and every input sample image is labeled by a teacher. The robot operates in an environment with both supervised and unsupervised learning modes, just like a human child does. There can be reinforcement signals which give feedback about the robot's response. Such a learning mode is termed the *reinforcement learning* mode.

We consider two experiments, viz., *dense label problem* and *sparse label prob-*

lem [49]. In the dense label problem, each image is *labelled* by a teacher, while in the sparse label problem, the label is available only at certain points in time. Most of the intermediate samples, between two labelling attempts by the teacher, are not labelled. The dense label problem is only of theoretical importance in testing how good the system works with all of the labelling information, while the sparse label problem is defined in a more practical set up.

4.1 Introduction

The main challenge in building autonomous robots is to build a core system that is *task non-specific* [46], and that learns from its interactions with the environment. Such a system cannot be model-based because model-specific information would limit the robot to only those models that have been built into the robot at the time of its "birth." Developmental robots follow a new paradigm that gives hope of an entirely new generation of robots that are built task non-specific and develop new skills unknown at the time of development without having to be re-programmed.

Developing a mechanism that lets the robot learn any task autonomously is a challenging problem. Some ideas have been proposed to achieve this goal. First, since our developmental algorithm [43] [45] should be task non-specific, it should not be using any user-defined features that would help recognize task specific patterns, be it auditory or visual patterns. One method is to use the raw data that comes from the sensors called the *sensation* or the *sensory input.* We then associate a particular action to a sensory input, say, we assign the name of a person to the image of his

frontal view. Now, when a similar pattern is seen later, the robot knows what action it should perform based on its experience, say, recognize the person by *priming* the name of the person.

It is important that *context information* be built into the robot so that under different contexts, the same sensory input would lead to different actions. The robot should first be trained for a particular task and it would learn, without any re-programming, any given task on which it is trained. We should be able to initially teach the robot some basic tasks and then scale up its behavior by defining new tasks based on the current knowledge the robot has acquired. This is called *behavioral scaling up* [52]. The robot scales up its behavior based on the primitive behaviors that have been taught earlier. Some experiments [52] related to behavioral scaling up have been done, in terms of *action chaining* in which the robot learns a sequence of actions and associates a set of basic actions to a complex command. This is an important milestone towards achieving the goal of development.

To avoid being monotonic and executing the same action in a given context, the concept of novelty [50] should be introduced that brings in some unpredictability into the robot. This could lead to a better development. To train the robot to give preference to some actions over others, a rewarding system or a value system [15] [52] should be used. This is analogous to saying "good" and "bad" to a child. If the value of a particular action is high compared to the others, the robot executes that particular action. A Q-learning algorithm [42] is used to back propagate the rewards from the present to the past so that the robot can choose the action that corresponds to the best Q-value when it encounters a similar situation in future. We

will not be focusing on the novelty and reward system as it is beyond the scope of the thesis. Next, we present the basic model of an agent.

4.2 The basic model of an agent

To start with, we define a few terms to help understand the basic model of an agent.

Definition *A context* is defined to be made up of three entities -

1. Sensation (or the sensory input). This can be an image, or audio signals, or a mixture of both.

2. Action (or response). This can be motor signals, or any means by which the robot can respond. For recognition experiments, we will assume the action to consist of class labels.

3. Q-value (or importance). This determines the importance of the action-sensation pair. This is an important parameter for selecting which *context* should be retrieved. The Q-value is updated using Q-learning algorithm.

Definition *Last context* or previous context is defined to be the context which was sensed in the immediate past. The last context is made up of last sensation, last action and has a corresponding Q-value associated with it.

Definition *Primed context* is defined to be the context which is predicted based on the past experience of the robot. Note that there can be many primed contexts and the primed context with the highest *Q-value* is usually selected to perform the next action if no external action is imposed.

The basic model uses the *last context* (or previous context), that consists of last sensation and last action. The model then primes or predicts the next context, called primed context that consists of primed action and primed sensation. The robot can *"think"* what predicted sensation the given sensation would result in and from the predicted sensation the robot could decide on what action to perform. The primed action is the action corresponding to the primed sensation which the robot would most probably execute next if no action is imposed. This model is shown in Figure 4.1 [50] [49]. The agent shown in Figure 4.1 is a mapping that predicts the primed context from the last context. In Figure 4.1 [49], the four types of information shown

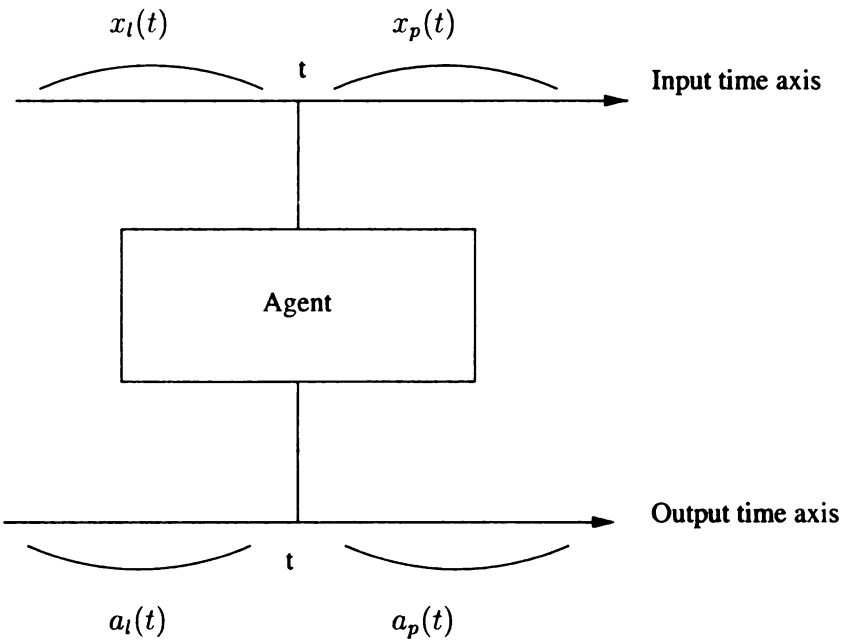


Figure 4.1: Basic model of an agent.

are the last sensation, $x_l(t)$, the last action $a_l(t)$, the primed sensation $x_p(t)$, and the primed action $a_p(t)$ along the time axes of input and output, respectively, around the current time.

The core of the agent shown in Figure 4.1 is a Hierarchical Discriminant Regression tree classifier [17]. HDR is a novel method to learn to classify patterns autonomously. But, HDR is invariant to rotation, scale and translation. We can mitigate the problem of *translation invariance* by having an attention selection mechanism that focuses on objects of interest in an entire scene. The other two invariants have to be taken care of for the HDR method to be used effectively. One such problem that deals with *rotational invariance* is the *sparse label problem.* We will address the sparse label problem later in the chapter. We group the dense label problem and the sparse label problem under the term the missing label problem.

4.3 Architecture for the missing label problem

The basic architecture of the SAIL robot, shown in Figure 4.2, was used for the missing label problem. The HDR in the cognitive mapping was used to classify patterns. The input is obtained from the vision module. Speech and touch were not used. The output space of the HDR classifier is an action vector. The action vector represents the labels of the training samples. Each element positioned in the action vector related to one class. The highest value in an array position indicates that the class label to be chosen is the one corresponding to the position of the highest value.

The HDR tree clusters the samples based on both the input and output space, resulting in better clustering and a good classification performance.

HDR employs a coarse-to-fine clustering method based on both the input X-space

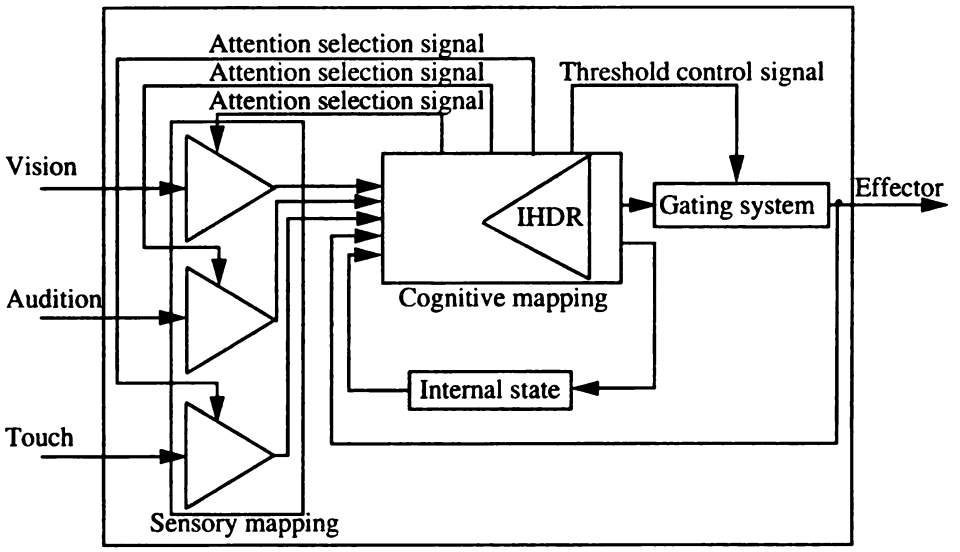


Figure 4.2: Basic architecture of the SAIL robot. (Adapted from [53])

and the output Y-space. The fineness of the clusters increases with the depth of the tree. First, dynamic clustering is performed in the Y-space to form coarse Y-clusters called virtual labels. In the X-space the X-data points, that correspond to same Y-cluster, are grouped to form X-clusters. Each Y-cluster is further divided by creating finer clusters or virtual labels, which group points in the input space to form finer X-clusters. Cluster statistics such as mean, covariance matrix are maintained at each node of the tree and are used to compute the probability that a sample (x,y) belongs to a given cluster. Based on this probability, the search progresses down the tree. Each node of the tree maintains some number of clusters. This type of clustering is called double clustering. An illustration of double clustering is shown in Figure 4.3. Given an input sample, the robot checks to see if an action is imposed. If so, it performs that action. If not, it primes a list of primed contexts and selects the one with the highest Q-value. The robot performs the action corresponding to the selected primed context. To retrieve the primed context with the closest match

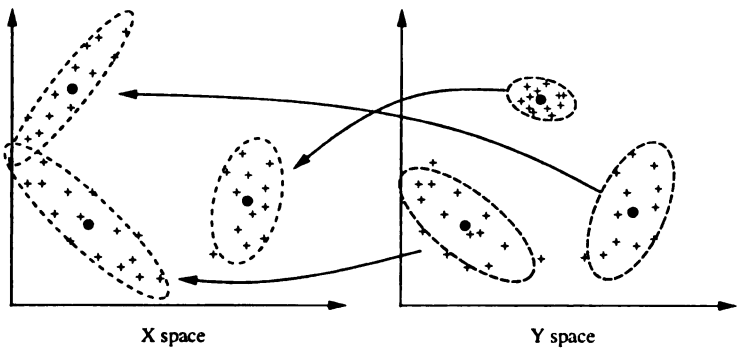


Figure 4.3: Double clustering in both input and output spaces. (Adapted from [53])

to the current sensation, we do not make an exhaustive search of all the samples in the HDR tree. Moreover, such a search is infeasible in real-time as the robot acquires more knowledge. The complexity of the search made in the HDR tree is proportional to the height of the tree and the number of samples allocated at each node of the tree. The search technique used to retrieve the primed context speeds up the search process. Tracing through the tree to retrieve a sample is done based on the cluster statistics stored at each node. Based on negative log likelihood, it is decided as to which child should be the next node in the path traced to retrieve the closest match to the current sensation. For more details on the HDR tree, the reader is referred to [17] [53].

4.4 Dense label problem

As a first step, the dense label problem is attempted, which doesn't take into account the contextual information about the sequence of presentations of an object. Every image is labeled in the training set. The test set is exclusive of the training set. The HDR algorithm is used to recognize the objects.

4.4.1 Experimental setup

The experiment was done using 13 toys, which were kept in focus of one of the SAIL robot's camera. The toys were rotated by hand to introduce some kind of variance. No image registration techniques were applied and no preprocessing was done. This makes recognition a very hard problem for any good classifier existing today. A common background was used while collecting pictures of the toys, as we do not want the robot to base its decision on background information. The 13 toys correspond to 13 classes, and each of these class objects have different orientation views, all corresponding to the same class label. Test patterns are randomly sampled from the collected images and are not given class labels or used during training. The HDR tree is trained using the collected training samples and the tree is tested on the test samples.

There are a total of 282 image samples that have been used for training. These images come from 13 classes or 13 toys. Thirty-eight test images were randomly sampled from among the orientations captured for each toy and were used as test images. The results are shown in Section 4.4.2. Each image is of dimension $60 \times 80 = 4800$.

4.4.2 Results

The results are shown in the table below. The first column specifies the class label. The second column shows how many of the randomly chosen test samples are correctly classified, and the third column shows how many of the randomly chosen test samples

S.No	Class label	Correct classifications	Incorrect classifications
1	Baby 1	2	0
2	Mickey	2	0
3	Dinosaur	3	0
4	Doggy	2	0
5	Baby 2	2	0
6	Dwarf	5	0
7	Girl	2	0
8	Barbie	3	0
9	Helicopter	3	1
10	Kitty	3	0
11	Ms.Mickey	4	0
12	Hug me	3	0
13	Zebra	0	4

Table 4.1: Recognition results for the dense label problem

# training samples	282
# test samples	38
% correct	86.8%
% incorrect	13.2%

Table 4.2: Percentage error for the dense label problem.

are incorrectly classified.

The final results are shown in Table 4.2. The error rate is 13.2%

4.4.3 Analysis

The performance was good except for the zebra toy class. The image of the zebra is relatively small and much of the area of the zebra coincides with the background (because of the stripes). Also, since our method has *translation invariance* and the zebra has alternating black and white stripes, there is a problem of the image vectors of two views of the zebra toy to be very far apart in the high-dimensional space resulting in poor clustering with respect to the nearby orientations. The disadvantage of positional or translation invariance is prominent for the zebra class as all four test

samples of this class are incorrectly classified. There is 86.8% accuracy and the error rate is 13.2%.

It is important to note that our method did not use any *user pre-defined features* to classify the objects. Only the raw sensory data was used to classify the objects. Moreover, no preprocessing or image registration techniques were applied. This was a task non-specific method, as opposed to methods used to solve a particular problem, say, face recognition, where the classification performance could be high because of predefined knowledge being used. There are some issues that have to be addressed when using the HDR method on raw sensory data. In the real world scenes, the HDR method should be used only after passing the image through an attention selector, which would give the object of interest. This object could then be matched in a task non-specific way to the prototypes that are stored in the HDR tree. The matched prototype also stores the primed context, which contains the primed action. This primed action vector could be used to execute a particular action. The space requirements of the basic form of the HDR algorithm are high, as we have to remember all the training samples. One optimization, which would decrease this space requirement, stores only prototype features (sensory inputs) in the HDR tree. Consequently, if two orientations were relatively close to one another, only one of them would be stored as the prototype.

The introduction of context information for recognition of the different orientations would improve the performance of the recognition system for cases where we do not have labels for each and every input sensation. We should take into account the time-related information as to which orientation followed which orientation, and

back propagate the primed sensation interpolating, in some way, the intermediate orientations. This brings us to the sparse label problem.

4.5 Sparse label problem

Here, we are given only some labels among a set of orientation views of a particular object. Our task is to label the other set of views. Let us take a simple example. If the robot sees a chair and the teacher says "chair" the robot associates that label to the visual sensory input it receives. So, first, there is a problem of associating and synchronizing the audio input and the video input. Now, the chair has to be identified in different orientations. So, the teacher rotates the chair and the robot has to know that the current sensory input still corresponds to the chair. To simplify, the teacher would say "chair" again. And the training can be repeated for different orientations, but without the teacher having to say "chair" for every frame of input in the video stream of the robot. The robot has only a few or *sparse labels* for the sensory input, and the task is to associate a new view of the chair during the testing phase, or when the label isn't given to it by the teacher.

To enable learning at contexts which are not labeled, the label information has to be passed on from those contexts which are labeled. This is done by using the Q-learning [42] algorithm. In Q-learning, each state $s(t)$ has a set of action values called the Q-values. Each Q-value corresponds to an action a. $a(t)$ is the action that brings the state $s(t-1)$ to state $s(t)$. $r(t)$ is the reward received at time t. The

Q-learning algorithm shows that Q-values are updated according to the equation 4.1.

$$Q(s(t-1), a(t)) \leftarrow (1-\alpha)Q(s(t-1), a(t)) + \alpha[r(t) + \gamma \max_{a'} Q(s(t), a')], \quad (4.1)$$

where α and γ are the learning rates. The action with the largest Q-value will be selected as the next action to be performed. Note that the Q-values are updated based on the reward $r(t)$ and the value of the next state $s(t)$, which allows delayed reward to be back-propagated in time during learning.

Because of this algorithm, the contexts which are not labeled get labeled in the back-propagation stage of the algorithm. So, after enough number of training samples, the system would prime the correct action when it sees a similar context in future.

4.5.1 Experimental setup

Sparse label problem was trained and tested in real time. The SAIL robot acquired images through its camera at the rate of approximately 5 frames per second. The robot was trained on 13 toys. The total number of training samples was 5283. A toy was placed in the robot's hand, and the robot would rotate the toy so as to see different orientation views of the toy. The robot was given labels at random intervals in time for the toy which it is seeing. Note that this gives rise to the sparse label problem, as each frame does not have a corresponding label. The label was given roughly every 10 seconds manually by the teacher in real-time. This experiment was repeated for each of the 13 toys. At the end of the training, the total number of samples collected was found to be 5283. Now, the testing phase began with no

Class#	Toy Name	# samples	# unlabeled	# correct	# incorrect
10	apeman	202	5	197	0
2	winnie	199	14	185	0
6	mickey	195	195	0	0
9	doggy	208	32	176	0
4	baby	219	14	204	1
5	kitty	198	37	161	0
8	MsMickey	191	40	151	0
1	girl	201	23	172	6
3	potter	189	106	83	0
11	heli	225	21	202	2
0	dwarf	198	22	161	15
12	gymbarbie	209	10	188	11
7	barbie	192	192	0	0

Table 4.3: Recognition results for the sparse label problem

difference in the robot architecture, except that the results of its action were written to a log file. The robot was allowed to develop its capability, just like a human child taking an examination. So, there will be no input from the human teacher. The robot will have to perform on its own. During testing, each of the 13 toys were presented to the robot but in a different order from that of testing. Some random order was used. The results of testing are presented in Section 4.5.2. The dimension of each image is $30 \times 40 = 1200$. The images used are gray-scale images.

4.5.2 Results

The results collected are shown in Table 4.3 in the order in which the toys were presented to the robot while testing. The class label starts from 0. The recognition rate and the error rate are shown in Table 4.4.

In Table 4.4, the first row shows the results for all 13 toys, while the second row shows results for the 12 toys which were recognized.

Expt	# of samples	% correct	% incorrect	% unlabeled
All samples	2626	71.59	1.33	27.08
Recognized samples	2434	77.24	1.44	21.32

Table 4.4: Percentage error for the sparse label problem.

4.5.3 Analysis

The performance of the robot was good considering that it was trained for just one epoch. The robot recognized 12 out of 13 classes. The robot was not able to recognize the last test sample class, *barbie*, which it was presented. It did not give incorrect results either. It considered this toy as a new object.

Note that the error rate or the percentage that the robot recognized incorrectly is less than 1.5%. The robot shows good learning rate. It makes mistakes rarely, and the un-labeled percentage shows that the robot is still learning. When a new sample which it recognized comes later in its view, the robot recognizes that the previous samples also belong to the same class as the current sample, and labels the old samples accordingly. So, the longer the toy is kept before the robot, the better its recognition rate becomes, unless the toy is not recognized at all. Each toy was tested for roughly the same amount of time.

In the experiment, some part of the toys went out of the area being captured while rotating the toys. It is not always possible for the robot to find the center axis of rotation so that the object is rotated in its view. If the view of the robot is increased, we will be giving way to more background information. To avoid that, we need some mechanism in future to pay attention to only the area which belongs to the object, or to the area in the image that is showing motion. Some kind of value

system should be reinforced in the robot. A preliminary collection of face data and experiments using human faces showed that the same problem is worsened while the subject rotates his or her head. This calls the need for attention selection mechanisms. In autonomous mental development framework, as task-specific programming is not allowed, we resort to extracting spatial features and spatio-temporal features from input images to train the robot to pay attention to a moving object based on a value system. Towards achieving this next step, we focus on feature extraction mechanisms in Chapters 5 and 6.

4.6 Conclusion

The performance of HDR to solve the dense label problem is very good for patterns that have more stability to positional and rotational invariance of the object. To overcome the positional invariance problem, we can preprocess the image using an attention selector through the sensory mapping. The sparse label problem deals with the rotational invariance part. The dense label problem and sparse label problem were addressed in this chapter with experiments and results. Results show good performance of the robot for dense and sparse label problem. In sparse label problem, the robot is seen to show learning capability and very less tendency to give inaccurate results. Also, there arose a need for addressing attention selection based on extracting spatial and spatio-temporal features. In the following chapters we shall present some feature extraction algorithms.

Chapter 5

New Incremental Independent Component Analysis

Principal component analysis has been one of the most often used methods for feature extraction. This is because, PCA derives a linear transformation to any sub-space that is optimal in the least square sense. PCA uses second order statistics (co-variance matrix) to derive mutually orthogonal set of features that are oriented along directions of highest variance. But, the information in the real world data is not fully captured by second order statistics. Methods to extract higher order information have gained importance in recent years. In most cases, it will be interesting to see the directions of the underlying independent components for the data. These "interesting" features vectors are extracted by a class of feature analysis techniques called *Independent Component Analysis* (ICA) . ICA aims at deriving a representation which has the minimal statistical dependence of the components in the new representation.

Many fast algorithms for ICA have been proposed. We can broadly classify the

kinds of algorithms proposed based on their usage of data into two categories -

1. **Batch Algorithms:** These algorithms use the statistics of a group of samples to update the estimates of the feature vectors. Batch algorithms find their use in off-line processing, where the entire data set is available. Eg: FastICA

2. **Adaptive or Incremental Algorithms:** These algorithms update the estimates of the feature vector as each new sample comes in. Typically, they are very popular in real-time online applications that need to be adaptive. Also, for appearance based vision applications which have gained popularity in recent years, we have to discard the old samples as a new sample comes in, because we operate in high-dimensions. We classify hybrid algorithms under batch algorithms, as they use a batch of data, instead of just one sample per update. Eg: Gradient algorithms, NPCA, ExtBS, etc.

Another popular categorization is based on the number of features the algorithms have to estimate.

1. **Symmetric algorithms:** Symmetric algorithms are those that try to estimate all the components simultaneously. They update the entire W matrix per update. Some of the popular algorithms under this class are [2], [25], [31].

2. **One-unit algorithms:** While symmetric algorithms try to estimate all the components simultaneously, it is often enough if we estimate only a subset of the ICs. For high-dimensional data of dimension d, it will be computationally very complex to estimate d independent components. Moreover, there might

not be so many independent components. So, for real-time online applications that operate in high dimensions, it is required that we use one-unit learning algorithms. One-unit learning rules based on kurtosis have been proposed in [23]. Typically such algorithms use deflation approaches [7].

As this thesis focuses on using appearance based methods for general object recognition, we have to use adaptive algorithms for estimating the features. In this chapter, we have derived new algorithms for incremental ICA, and show their faster convergence over existing adaptive methods.

5.1 Independent component analysis

Independent Component Analysis (ICA) [6] is a statistical technique to extract the components that are as independent as possible from multi-dimensional data. ICA has its roots in Blind Source Separation. One popular example to illustrate the use of ICA is the cocktail party problem. Assume a hall in which there are many sources of sound with many microphones. Let there be m sources and n microphones. The sound recorded by each of the n microphones is a function of all the m sources and their distance from the microphone. In other words, each microphone records mixed signals of different sounds. The *cocktail party problem* deals with extracting the different independent sound sources from the mixed sound signals recorded by the microphones. Assuming the source signals are given by matrix S with each column being a source signal, and let X be the mixed signal matrix, with each column being the information recorded by each microphone. Typically, it is assumed that $m = n$,

though it is possible to extract the independent sources for $n >= m$. Let A be the mixing matrix which determines the ratio in which the signal have been mixed. So,

$$X = A * S \tag{5.1}$$

The only source of information we are given is the matrix X and the goal of ICA is to estimate the matrix A, so as to get the independent sources S from A and X. In many cases, it is not unrealistic to assume that the sound signals from different sources are statistically independent in time. So, assuming this property of statistical independence of the components of S, we can estimate the matrix A, or equivalently estimate a matrix W such that

$$W^T * X = W^T * A * S \tag{5.2}$$

gives back the set of independent signal directions. Most of the methods used to estimate W have better convergence properties if they work on whitened data. So, pre-whitening is usually done on X before estimating W as follows.

$$W^T * M * X = W^T * M * A * S = Q * S \tag{5.3}$$

where, M is the whitening matrix, and Q is a matrix which takes care of the fact that independent signals can be estimated only up to a permutation, sign or magnitude.

The reason for this is because of equation 5.4

$$A * S = A * P * P^{-1} * S \tag{5.4}$$

where, P is any permutation matrix. In a similar way, we can also see how the sign and magnitude can also be split up between A and S. However, if we assume that the independent components in S are unit vectors, then, this makes matrix A determinable up to a sign and permutation. Also, as S is made up of independent components, they are also orthogonal with respect to each other.

$$E(S * S^T) = I \tag{5.5}$$

The estimation of W is done my maximizing the statistical independence of $W^T * X$. Two popular methods for maximizing the statistical independence are

1. **Maximizing the non-Gaussianity:** The reason for maximizing the non-Gaussianity to achieve independence is because of the *Central Limit Theorem* (CLT). According to the central limit theorem, a linear combination of many non-Gaussian sources is more Gaussian. This means that the independent sources can be obtained if we can maximize the non-Gaussianity of the mixture of signals. There are two popular methods for maximizing the non-Gaussianity of the signals. They are

   (a) *Maximizing absolute value of kurtosis:* Kurtosis is a popular measure of

non-gaussianity. Kurtosis is defined as

$$\text{kurt(y)} = \frac{E(y^4)}{(E(y^2))^2} \tag{5.6}$$

where, y is a random variable with zero mean. According to this definition the kurtosis of a Gaussian distribution has the value 3. So, there has been another definition of kurtosis, which makes the value for Gaussians to be zero.

$$\text{kurt(y)} = E(y^4) - 3 * (E(y^2))^2 \tag{5.7}$$

The more the deviation from 0 for kurtosis, the more the non-Gaussianity. Kurtosis is positive for super-Gaussians and negative for sub-Gaussians. In this way,the super-Gaussians have been defined as those signals characterized by positive kurtosis and sub-Gaussians have been defined as those signals characterized by negative kurtosis. Super-Gaussians are characterized by having a heavy tail. Laplacian distribution is super-Gaussian while uniform distribution is sub-Gaussian. Figure 5.1 shows the shape of super and sub-Gaussian *p.d.f* with respect to the Gaussian *p.d.f.* Kurtosis uses higher order information to estimate the non-Gaussianity. Kurtosis belongs to a class of higher order functions called *cumulants.* Let us assume that we want to find a linear combination $w^T x$ of samples x_i, such that it has the maximum non-gaussianity (maximal or minimal kurtosis). We have to impose a constraint on w in-order for our search to be meaningful.

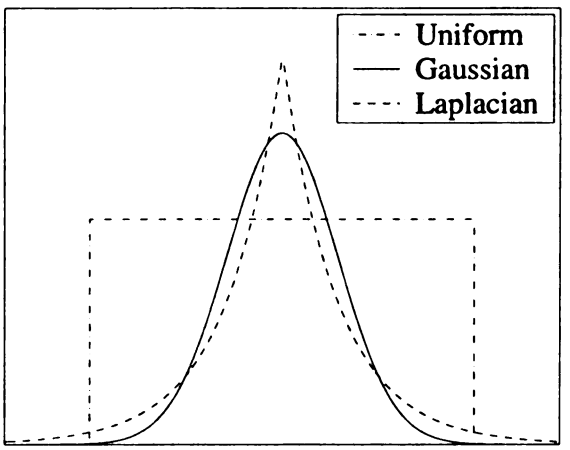


Figure 5.1: Comparison of analytical distributions.

So, let us assume that the variance of the linear combination of x is unity, or, $E\{(w^T x)^2\} = 1$. Let $z = A^T w$. Then,

$$\|z\|^2 = w^T A s s^T A^T w = w^T A A^T w = w^T E\{x x^T\} w = E\{(w^T x)^2\} = 1 \tag{5.8}$$

Also,

$$\text{kurt}(w^T x) = \text{kurt}(w^T A s) = \text{kurt}(z^T s) \tag{5.9}$$

We know that kurtosis is a measure of non-Gaussianity. Kurtosis of $w^T x$ can be maximized only if $z^T s$ is such that there is only one independent component and all other additive terms are zero. This is a direct consequence of the central limit theorem. A more rigorous proof of the above statement could be found in [7]. So, assuming that there exists at least one independent component in the data, there exists i, such that $z_i = \pm 1$, and $z_j = 0, \forall i \neq j$ for the linear combination $w^T x$ estimated in Equation 5.9.

Though kurtosis has been a popular option to estimate independent components, it suffers from non-robustness, as do most cumulant-based estimators. This is because, high order cumulants mainly measure the tails of a distribution and are very sensitive to out-liars because of their high-order. A simple example of how non-robust kurtosis is can be seen from the illustration given below. Assume a zero mean, unit variance Gaussian distribution with 1000 samples, with one out-liar at 10. Calculate the value of kurtosis using equation 5.7. The kurtosis of the sample set without out-liars was found to be -0.0078, while the kurtosis of the samples with one out-liar of value 10 was found to be 9.33. This shows the sensitivity of kurtosis to out-liars because of its high order. Many higher-order cumulants have a similar problem.

(b) *Maximizing negentropy:* Another method to measure the non-Gaussianity is using negentropy. It is known that Gaussian distribution has the maximum entropy. So, non-Gaussianity can be measured as a difference of the entropy of the sample distribution from the entropy of the Gaussian distribution. Negentropy is given by $J(y)$,

$$J(y) = H(y_{gauss}) - H(y) \tag{5.10}$$

$$H(y) = -\int p(y) \log p(y) dy \tag{5.11}$$

where $H(y)$ is the entropy of a random variable y.

This measure is a statistically sound method to measure non-Gaussianity. Negentropy is always non-negative and is zero if and only if the distribution is Gaussian. The disadvantage of this method is that we need to know the probability distribution of the samples, which is unavailable in most cases. To overcome this disadvantage, many approximation functions have been proposed for negentropy that have a smoother functional form than kurtosis. These approximations are typically of the form given in Equation 5.12,

$$J(y) \approx \frac{1}{12} E\{y^3\}^2 + \frac{1}{48} kurt(y)^2 \tag{5.12}$$

where y is a random variable with zero mean and unit variance. This approximation suffers from the same problems as kurtosis for symmetric distributions, because, for symmetric distributions the first term in Equation 5.12 becomes zero. Therefore, more sophisticated approximations were developed using "nonpolynomial moments" [19], to yield approximation functions of the form

$$J(y) \propto [E\{G(y)\} - E\{G(\nu)\}]^2 \tag{5.13}$$

where ν is a Gaussian random variable with the same mean and variance as y. Generally, choosing functions that do not grow too fast for $G(y)$, one can obtain robust estimators. Some of the popular choices for $G(y)$

are [19].

$$G(y) = \frac{1}{a} \log \cosh ay, 1 \leq a \leq 2 \tag{5.14}$$

$$G(y) = -\exp(-y^2/2) \tag{5.15}$$

2. **Minimizing mutual information:** Mutual information between random variable y_i is defined as follows

$$I(y_1, y_2, ..., y_d) = \sum_{i}^{d} H(y_i) - H(y) \tag{5.16}$$

where H is differential entropy defined in Equation 5.11. Mutual information is always non-negative and zero if and only if the variables are all statistically independent. But, this approach suffers from the drawback of having to know the probability distribution. We will not be dealing with these methods in this chapter.

There are other techniques like infomax [2], maximum likelihood estimation [21] to perform ICA. Most of the methods existing today estimate W by maximizing the non-Gaussianity of $W^T * x$. In general, these techniques for ICA have better convergence properties for whitened data. In the methods that are discussed in this chapter, we will go with the assumption that the data is whitened before extracting independent components.

5.2 Existing methods for independent component analysis

Having discussed the principles that can be used to estimate independent components, we will now look at algorithms that use some of the criteria discussed above. The performance of an algorithm depends both on statistical properties of the criterion used to estimate the independent components (ICs), and also on the optimization technique used. In this section we shall look at some of the existing techniques to extract ICs.

There have been many batch ICA algorithms, among which FastICA is worth mentioning because of its convergence speed. All the other algorithms presented in this chapter are incremental or adaptive algorithms.

5.2.1 FastICA

FastICA is a batch algorithm originally developed by Hyvärinen and Oja [22] using kurtosis, and was later generalized for general contrast functions in [18] [20]. There exist both symmetric and one-unit algorithms for FastICA. Here, we shall see the underlying derivation of FastICA [18].

We know that the extrema of kurtosis of $w^T x$ will give the components corresponding to the maximum non-Gaussianity. So, we have to maximize F_1, where

$$F_1 = kurt(w^T x) = E\{(w^T x)^4\} - 3E\{(w^T x)^2\}^2 \tag{5.17}$$

We have to constraint $w^T x$ in some way, or else the maximization of F_1 will not be meaningful. So, let the variance of $w^T x$ be equal to 1.

$$E\{(w^T x)^2\} = 1 \Rightarrow w^T C w = 1 \tag{5.18}$$

We have to maximize F_1 subject to the constraint that $w^T C w = 1$. So, applying Kuhn-Tucker's conditions, we have to find the extrema for F

$$F = E\{(w^T x)^4\} - 3E\{(w^T x)^2\}^2 - \lambda(w^T C w - 1) \tag{5.19}$$

Taking the derivative,

$$\frac{\delta F}{\delta w} = 4E\{(w^T x)^3 x\} - 3 * 2 * 2E\{(w^T x)^2\}E\{(w^T x)x\} - 2\lambda C w \tag{5.20}$$

$$\frac{\delta F}{\delta w} = 0 \Rightarrow 2\lambda C w = 4E\{(w^T x)^3 x\} - 3 * 2 * 2E\{(w^T x)^2\}E\{(w^T x)x\}$$

$$\Rightarrow 2\lambda C w = 4E\{(w^T x)^3 x\} - 3 * 2 * 2 * C * w, \text{ since } E\{(w^T x)^2\} = 1$$

$$\Rightarrow w = \frac{2}{\lambda}[C^{-1}E\{(w^T x)^3 x\} - 3 * w]$$

where, λ is a constant that can be eliminated by normalizing w. For whitened data, $C = I$. Using fixed point optimization technique discussed in Section 2.4.3, the algorithm reduces to

$$\boxed{\begin{array}{c} w^*(k) = E\{(w(k-1)^T x)^3 x\} - 3 * w(k-1) \\ w(k) = w^*(k)/\|w^*(k)\| \end{array}} \quad (5.21)$$

Generalizing for any arbitrary non-linearity g, the algorithm can be derived in the same fashion to get

$$\boxed{\begin{array}{c} w^*(k) = E\{g(w(k-1)^T x)x\} - E\{g'(w(k-1)^T x)\} * w(k-1) \\ w(k) = w^*(k)/\sqrt{(w^*(k))^T C w(k)} \end{array}} \quad (5.22)$$

This derivation gives one column of W matrix. To get the other columns of W, we have to make sure that we do not derive the same vector again and again. This is done by de-correlating the data w.r.t the previous columns of W estimated. The representation W stands for a matrix whose columns are $w(k)$

5.2.2 Extended Bell-Sejnowski algorithm

Bell-Sejnowski (BS) derived an infomax algorithm that was effective in separating super-Gaussian sources with heavy tails. One form of the original BS algorithm is given in Equation 5.23

$$\Delta W \propto [I - 2\tanh(u)u^T]W, \text{where } u = W^T x \quad (5.23)$$

The extended infomax algorithm [29] derives two learning rules - one for super-Gaussian and another for sub-Gaussian, and switches between super-Gaussian and

sub-Gaussian learning rule depending on a switching parameter k_i defined in Equation 5.25. The extended infomax algorithm is given in Equation 5.24

$$\Delta W \propto [I - K\tanh(u)(u^T) - uu^T] \times W \tag{5.24}$$

$$k_i = \text{sign}\{E\{\text{sech}^2(u_i)\}E\{u_i^2\} - E\{\tanh(u_i)u_i\}\} \tag{5.25}$$

with $k_i = 1$ for super-Gaussians and $k_i = -1$ for sub-Gaussians and k_i are elements of the diagonal matrix K.

5.2.3 Nonlinear PCA (NPCA)

Non-linear PCA criterion for ICA was proposed by Oja [31] [32]. The criterion function is defined in Equation 5.26

$$J(W) = \|x - Wg(x^TW)\|^2 \tag{5.26}$$

Accordingly, an algorithm based on gradient descent using this criterion is given in Equation 5.27, where g is a suitable non-linear scalar function.

$$W_{k+1} = W_k + \mu_k[x_k - W_k g(u_k)]g(u_k^T), \text{where } u_k = W_k^T x_k \tag{5.27}$$

The least-squares method for faster convergence properties was proposed in [34] and a comparison with other approaches as to how NPCA criterion relates to existing

algorithms was given in [26]. The sequential one-unit recursive NPCA algorithm is as follows.

$$
\begin{aligned}
&x_1(t) = x(t) \\
&\text{For each i = 1,...m compute} \\
&\quad z_i(t) = g(w_i^T(t-1)x_i(t)) \\
&\quad d_i(t) = \beta d_i(t-1) + [z_i(t)]^2 \\
&\quad e_i(t) = x_i(t) - w_i(t-1)z_i(t) \\
&\quad w_i(t) = w_i(t-1) + e_i(t)[z_i(t)/d_i(t)] \\
&\quad x_{i+1}(t) = x_i(t) - w_i(t)z_i(t)
\end{aligned}
\tag{5.28}
$$

where a good choice of β should be close to 1 and $0 \leq \beta \leq 1$. $g(x)$ is a suitable non-linear function. A choice for $g(x)$ is $\tanh(x)$ for sub-Gaussians and $x - tanh(x)$ for super-Gaussians.

5.2.4 Gradient algorithms

The gradient algorithms for ICA can be derived easily from the contrast functions or criterion developed for extracted independent components like those based on kurtosis, negentropy, higher order cumulants, etc. For example, taking the contrast function defined in Equation 5.13, we can obtain the following Hebbian-like learning

rule in Equation 5.29

$$\Delta w \propto [E\{G(w^T x)\} - E\{G(\nu)\}] * x * g(w^T x), \text{normalize } w \tag{5.29}$$

These one-unit gradient algorithms were first introduced using kurtosis in [7]

5.3 Incremental methods developed

We can broadly classify the incremental algorithms into two types - iterative and non-iterative algorithms. Iterative algorithms are those that can iterate over the same sample again and again to get better estimates. Non-iterative algorithms just use the sample once to update their estimates. Examples of non-iterative algorithms are gradient descent, NPCA, etc. Iterative incremental algorithms are typically derived using iterative optimization techniques like fixed point, Newton-Raphson, etc. In this section we improved the performance of gradient descent algorithms using amnesic parameter and derive two incremental iterative algorithms based on the fixed point optimization technique, though some of these can also be derived using Newton-Raphson techniques. One of the methods presented is an approximation of the incremental version of FastICA.

5.3.1 Amnesic gradient ascent of the square of kurtosis

For empowering gradient methods with faster convergence, we use the amnesic parameter in the learning rate of the function. Amnesic parameter provides a way to

forget the old data so that the methods using this parameter to forget old data can adapt to newer data faster. We use the amnesic parameter to estimate the incremental amnesic mean. The data coming in should be subtracted off its amnesic mean estimated incrementally. The gradient of kurtosis is given as

$$\Delta kurt(w^T x) = 4E\{(w^T x)^3 x\} - 3 * 2 * 2E\{(w^T x)^2\}E\{(w^T x)x\} \tag{5.30}$$

Now, we can not cancel out $E\{(w^T x)^2\}$, or even substitute $E\{(w^T x)x\}$ by w like we did in the derivation of the FastICA method for kurtosis in Section 5.2.1. The reason for this is because, during the initial few iterations that estimate the value of w, these conditions imposed in the derivation of FastICA are not true. So, we need to keep track of all the estimates in Equation 5.30 as follows.

$$\begin{aligned} A_n &= \frac{n-1-\mu(n)}{n} A_{n-1} + \frac{1+\mu(n)}{n}(w^T(n-1)x_n)^3 x_n \\ B_n &= \frac{n-1-\mu(n)}{n} B_{n-1} + \frac{1+\mu(n)}{n}(w^T(n-1)x_n)^2 \\ C_n &= \frac{n-1-\mu(n)}{n} C_{n-1} + \frac{1+\mu(n)}{n}(w^T(n-1)x_n)x_n \end{aligned} \tag{5.31}$$

So, the incremental rule to estimate w is

$$w(n) = w(n-1) - \frac{1+\mu(n)}{n} * 4\,[A_n - 3 * B_n * C_n] \tag{5.32}$$

This gradient descent extracts sub-Gaussian components. For super-Gaussian components, the formula is

$$w(n) = w(n-1) + \frac{1+\mu(n)}{n} * 4\left[A_n - 3 * B_n * C_n\right] \tag{5.33}$$

To combine both the formulae, we can either consider gradient ascent of the modulus of kurtosis, or gradient ascent of the square of kurtosis. This will result in extraction of both super-Gaussian and sub-Gaussian components. Here, we shall consider the gradient ascent of $kurt(w^T x)^2$. Deriving the formula in the same way with the criterion function as

$$F = kurt(w^T x)^2 \tag{5.34}$$

and,

$$D_n = \frac{n-1-\mu(n)}{n} D_{n-1} + \frac{1+\mu(n)}{n} (w^T(n-1)x_n)^4 \tag{5.35}$$

we get the new learning rule to be,

$$\boxed{w(n) = w(n-1) + [(1+\mu(n))/n] * 8 * [D_n - 3 * B_n^2] * [A_n - 3 * B_n * C_n]} \tag{5.36}$$

Amnesic mean

The amnesic parameter is defined in Equation 5.37.

$$\mu(n) = \begin{cases} 0 & \text{if } n \leq n_1 \\ c(n - n_1)/(n_2 - n_1) & \text{if } n_1 < n \leq n_2 \\ c + (n - n_2)/m & \text{if } n_2 < n \end{cases} \tag{5.37}$$

where $\mu(n)$ is the amnesic function depending on n. For incremental estimation, we use what is called *amnesic mean* [48].

$$\bar{x}_{(n)} = \frac{n - 1 - \mu(n)}{n} \bar{x}_{(n-1)} + \frac{1 + \mu(n)}{n} x_n \tag{5.38}$$

The way to compute a mean incrementally is not new but the way to use the amnesic function of n is new for computing a mean incrementally. The amnesic parameter $\mu(n)$ has three intervals. When this amnesic parameter is used for calculating amnesic mean as in Equation 5.38, straight incremental average is used when n is in the first interval. Then, $\mu(n)$ changes from 0 to 2 linearly in the second interval. Finally, n enters the third region where $\mu(n)$ increases at a rate $1/m$, meaning the second weight $(1 + \mu(n))/n$ in Equation (5.38) approaches to a constant $1/m$. Thus about m last observations are kept by the amnesic mean. This effect is equivalent to forgetting old data.

5.3.2 Incremental fixed point on kurtosis

Here we derive an iterative incremental method using fixed point optimization technique to find the extrema of the gradient of kurtosis. Let us use the same notation of A_n, B_n, C_n used in Section 5.3.1 in Equation 5.31. As the data is whitened, we can assume that the covariance matrix $C = I$ in the long run. The following equation can be got applying Kuhn-Tucker conditions to find the extrema of the kurtosis under the condition discussed in Section 5.2.1

$$2\lambda Cw = 4E\{(w^T x)^3 x\} - 3 * 2 * 2E\{(w^T x)^2\}E\{(w^T x)x\} \tag{5.39}$$

In order to avoid the matrix inverse due to the term C on the left hand side of the equation 5.39 during the initial stages of computation, we instead tried to enforce other constraints during the iteration stages that $A_n^T w > 0$, so as to keep w along the valid direction. This condition is got from the underlying formula for A_n, as the fourth power of any real number $(w^T x)$ is positive. So, we got the following incremental rule to update w.

$$\begin{aligned} w^*(n) &= A_n - 3 * B_n * C_n \\ w(n) &= w^*(n)/\|w^*(n)\| \end{aligned} \tag{5.40}$$

This method does not involve matrix inversion, while it has been approximated by a reinforcement of the direction of w towards the direction of A_n.

5.3.3 Incremental fixed point on negentropy

The incremental fixed point algorithm for negentropy can be derived using the same principles as used in Section 5.3.2. The incremental updating rule for w is as follows

For each sample x_n

begin

For i = 1,2,...d, where d is the dimension of the sample

begin

(a) $w_i^*(n) \leftarrow A'_n - B'_n * C'_n$

(b) $w_i(n) = w_i^*(n)/\|w_i^*(n)\|$

Iterate over steps (a) and (b) till w_i converges to a value or till maximum number of iterations exceeds a threshold.

end

Perform residue method: $x_n \leftarrow x_n - (w^T x)w$

end

(5.41)

where,

$$
\begin{aligned}
A'_n &= \frac{n-1-\mu(n)}{n} A'_{n-1} + \frac{1+\mu(n)}{n} g(w^T(n-1)x_n)x_n \\
B'_n &= \frac{n-1-\mu(n)}{n} B'_{n-1} + \frac{1+\mu(n)}{n} g'(w^T(n-1)x_n) \\
C'_n &= \frac{n-1-\mu(n)}{n} C'_{n-1} + \frac{1+\mu(n)}{n} (w^T(n-1)x_n)x_n
\end{aligned}
\tag{5.42}
$$

Here, g is a smooth function unlike kurtosis, and should be selected to be robust against outliers. A good choice of g is $g(y) = \tanh(y)$, which means, $g'(y) = \cosh^{-2}(y)$ and $G(y) = \log\cosh(y)$. In this notation, the derivative of G is g and the derivative of g is g'

In this method, we do not use GSO unlike the batch FastICA method to decorrelate the samples. We use the residue method which is of very less complexity than the GSO. Residue method has only one update per decorrelation, while the GSO has $(d-1)$ updates for the d^{th} decorrelation. The reason for this is because, using this method, w is in the same space as x as can be seen from the formula , and so the residue method can be applied.

5.3.4 ccNegIICA

In this method, we have a novel method of *pulling* the w vector in or against the direction of the sample vector with a weight equal to the non-Gaussianity contributed by the sample. In order to avoid oscillation of the w vector, the pull should always

be in the direction of the existing w vector. This is possible if the weight function which determines the importance of the sample vector is an *odd function of* $w^T x$. So, we have the following incremental updating rule for updating w.

$$w_i(n) = \frac{n-1-\mu(n)}{n} w_i n - 1 + \frac{1+\mu(n)}{n} f(\frac{w_i^T(n-1)\cdot x_n}{\|w_i(n-1)\|}) * x_n \tag{5.43a}$$

$$x_n \leftarrow x_n - (w_i^T(n)\cdot x_n) * \frac{x_n}{\|x_n\|} \tag{5.43b}$$

where,

$$f(x) = \|E\{G(x)\} - E\{G(\nu)\}\|^{1/4} * sign(x)$$

$$G(x) = \log\cosh(x)$$

where,ν is a (0,1) Gaussian random variable.

5.4 Results and analysis

In this section, we will see the results of running the algorithms developed against NPCA which is supposed to be the best of the previously existing algorithms [14]. The NPCA algorithm converges for sub-Gaussian sources with $g(x) = \tanh(x)$. For super-Gaussians $g(x) = y - tanh(x)$ was suggested [14]. We need one algorithm that converges for both super-Gaussian and sub-Gaussian sources. The later function was tried out for super-Gaussians for 25-dimensions and did not converge. So, here we use the same algorithm for NPCA that was used for sub-Gaussian data. The error

is measured in radians between the best matching ideal w vector and the w vector which is derived. The algorithm used to measure error is as follows.

5.4.1 Measuring convergence error

It has to be noted that the independent components that are derived can be permuted, and could be different in sign too. So, the features derived ,W, can only indicate the directions of the feature vector. So, we need an algorithm to measure the convergence error between the ideal W and the derived W. Error is measured by using the following algorithm for artificial data, with known mixing matrix A . Let M be the whitening matrix, and W be the incremental estimate obtained using the adaptive algorithms, whose columns are of unit length. Let m be the dimension of the input vectors.

1. $Q \leftarrow$ orthonormalize $((A^{-1} * M^{-1})^T)$

2. $D = W^T * Q$

3. $F_r(0) = \{\}, F_c(0) = \{\}$

4. For $k = 1, 2 \ldots m$

   $$E_k = cos^{-1}\{D_{i,j} = max\ D | i \notin F_r(k-1) \cap j \notin F_c(k-1)\}$$

   $$F_r(k) \leftarrow F_r(k-1) \cup \{i\}, F_c(k) \leftarrow F_c(k-1) \cup \{j\}$$

5. Total Error $\leftarrow \frac{1}{m}\Sigma_{k=1}^{m} E_k$

This algorithm finds the best match between any vector in Q and W and removes these two vectors from the matrices and finds the next best matching pair and so on. The total error is computed as the sum of errors for all the matches.

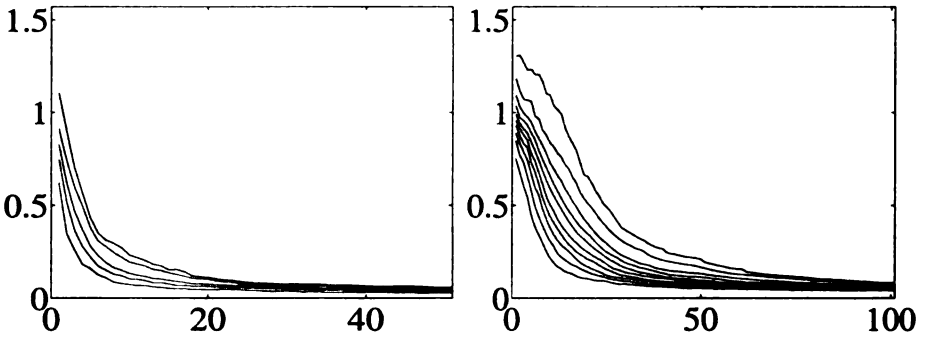
5.4.2 Uniform distribution - sub-Gaussian

While reading the results, it has to be understood that the independent components extracted do not have an order. So, while taking the mean across many runs, we sorted the error for each of the w's estimated and then took an average. This is why you do not see one curve crossing over the other. Also, each of the curves is not labeled because the w's are unordered, and also because of the way we took the mean.

Figure 5.2 shows the results of using amnesic gradient ascent of the square of kurtosis. The method converges faster than the existing methods, the best of which is NPCA as reported in [14]. For the figures given in this section, the x-axis is the number of samples $\times$ 100 and y-axis is the error in radians of the features derived. Each of the figures shows the convergence of the incremental ICA method used.

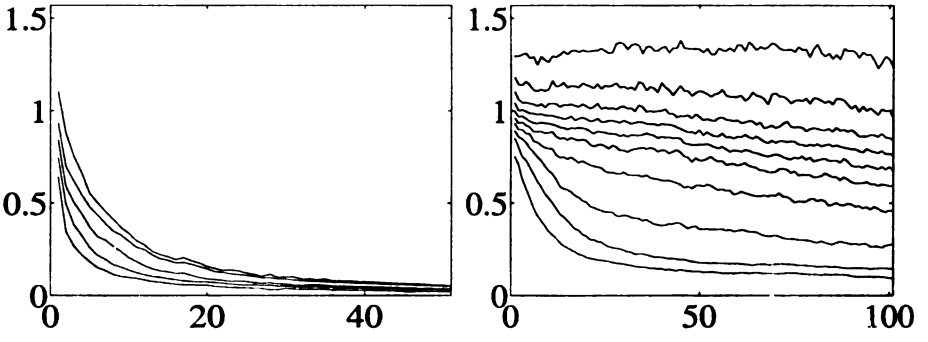
NPCA's results are shown in 5.6. The results for incremental fixed point on kurtosis discussed in Section 5.3.2 is shown for data with sub-Gaussian components in Figure 5.3. Figure 5.4 shows the convergence of incremental fixed point on negentropy (refer section 5.3.3) for sub-Gaussian data. The convergence for this method is the fastest of all the methods for sub-Gaussian data.

The results for ccNegIICA method is shown in Figure 5.5. This method is faster



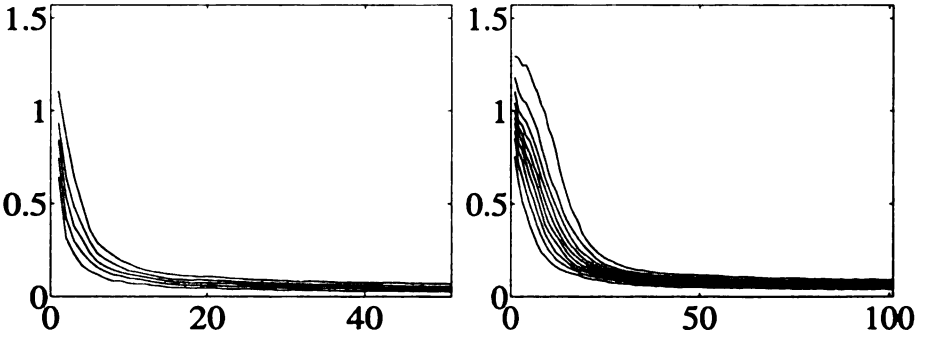
(a) (b)

Figure 5.2: Amnesic gradient ascent of kurt^2 for sub-Gaussians. (a) 5-D b) 10-D.



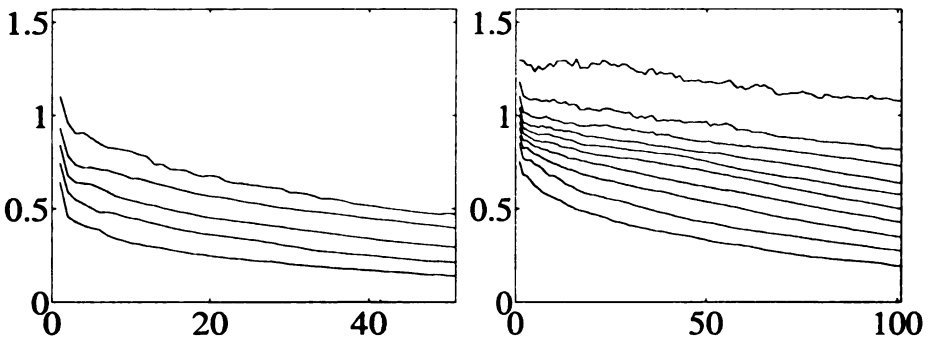
(a) (b)

Figure 5.3: Fixed point on kurtosis for sub-Gaussians. (a) 5-D b) 10-D.



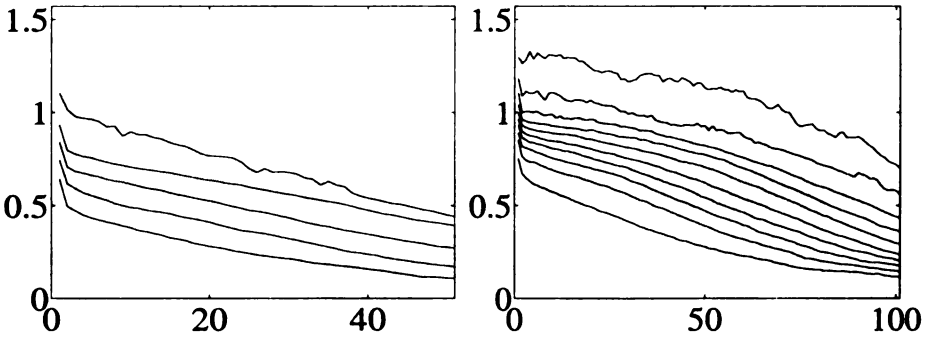
(a) (b)

Figure 5.4: Fixed point on negentropy for sub-Gaussians. (a) 5-D b) 10-D.



(a) (b)

Figure 5.5: ccNegIICA for sub-Gaussians. (a) 5-D b) 10-D



(a) (b)

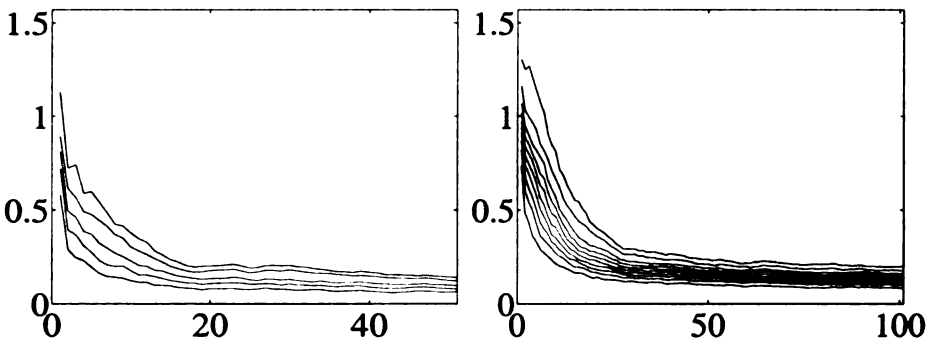
Figure 5.6: NPCA for sub-Gaussians. (a) 5-D b) 10-D.

than the three methods discussed above, because of its use of residue method to de-correlate the data.

The results for NPCA in Figure 5.6 show that it converges faster than ccNegIICA but is slower than the other methods developed.

5.4.3 Laplacian distribution - super-Gaussian

In this section we present the results of convergence for super-Gaussian data. It is seen that amnesic gradient ascent of the square of kurtosis shown in Figure 5.7



(a) (b)

Figure 5.7: Amnesic gradient ascent of kurt^2 for super-Gaussians. (a) 5-D b) 10-D.

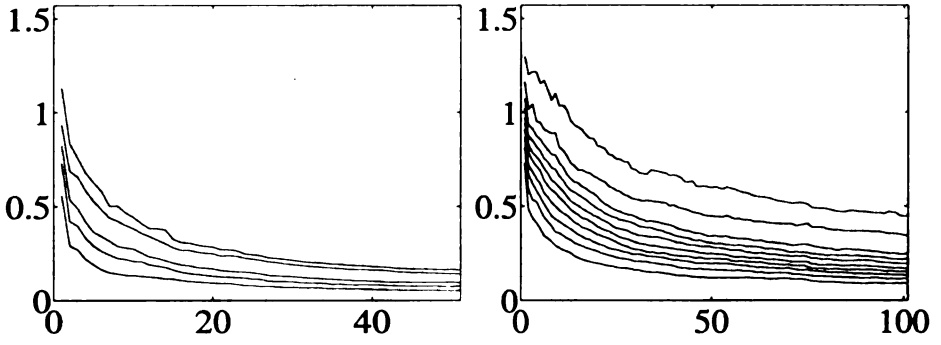
converges faster than other methods.

Fixed point on negentropy does not converge for super-Gaussians, because of its failure to converge during the initial few samples while applying the fixed point algorithm iteratively. This is because of our approximation of $C = I$, which is true as $n \rightarrow \infty$, but is a coarse approximation for the initial few samples. We see that incremental fixed point of kurtosis tends to converge slowly even with the same approximation of $C = I$ during the initial stages of the algorithm. The results for this method are shown in Figure 5.8.

ccNegIICA does not converge for super-Gaussians and neither does NPCA shown in Figure 5.11.

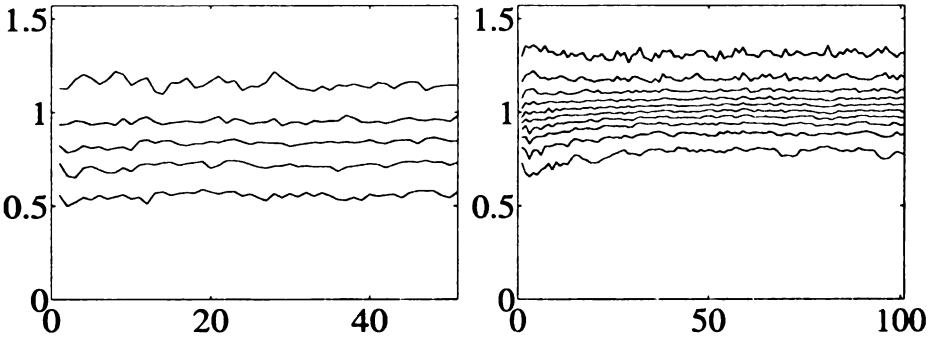
5.5 Conclusion

In this chapter, we developed incremental ICA algorithms - amnesic gradient ascent of the square of kurtosis, incremental fixed point of kurtosis, incremental fixed point of negentropy, ccNegIICA and compared their results with one of the best known



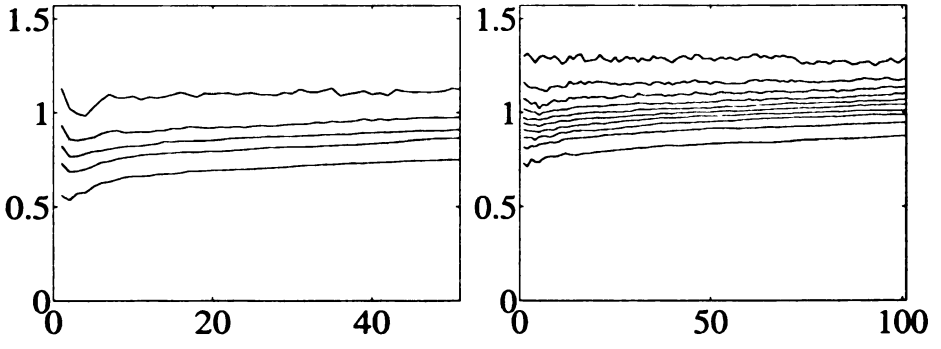
(a) (b)

Figure 5.8: Fixed point on kurtosis for super-Gaussians. (a) 5-D b) 10-D.



(a) (b)

Figure 5.9: Fixed point on negentropy for super-Gaussians. (a) 5-D b) 10-D.



(a) (b)

Figure 5.10: ccNegIICA for super-Gaussians (a) 5-D b) 10-D.

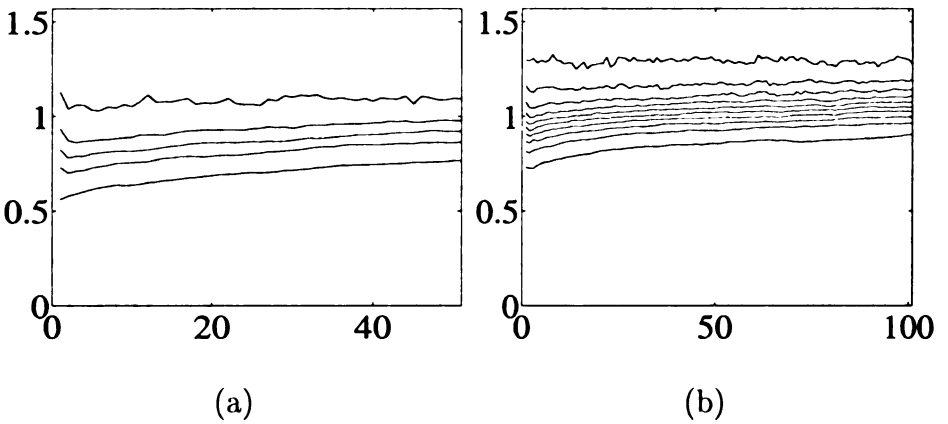


Figure 5.11: NPCA for super-Gaussians (a) 5-D b) 10-D.

incremental ICA algorithm (NPCA) for both sub-Gaussian and super-Gaussian data. The gradient ascent of the square of kurtosis converges faster than all the algorithms for the super-Gaussian case. The incremental fixed point on neg-entropy converges faster than other methods for sub-Gaussian case, though it did not converge well for the super-Gaussian case. The amnesic gradient ascent of the square of kurtosis was found to be the most stable in convergence for both super-Gaussian and sub-Gaussian data. In the next chapter, we introduce a new domain of feature analysis and present a much faster and better converging algorithm for extracting super-Gaussian features, which are abundant in real world data.

Chapter 6

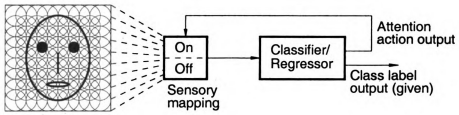
Lobe Component Analysis

Principal Component Analysis (PCA) and Independent Component Analysis (ICA) have been widely used to extract features of interest in a given data set. In many applications, the features we want to extract are the major directions along which the data is distributed. Imagining the data to be distributed in high dimensional space as lobes of data, the representative directions of these lobes are very informative of the data distribution. In this chapter, we propose a new feature analysis technique called *Lobe Component Analysis* (LCA) that deals with extracting such lobe directions or components. We define lobe components as those directions, not necessarily mutually orthogonal, which best express the sample set based on a criterion. We propose a new measure of expressiveness called the *Collective Best Match* (CBM) criterion biologically motivated by *sparse coding* properties. We show how this criterion can lead to the derivation of features that express data better than the least mean square error criterion upon which PCA is built. We derive an fast adaptive algorithm based on the CBM criterion and discuss its properties like representation, expressiveness,

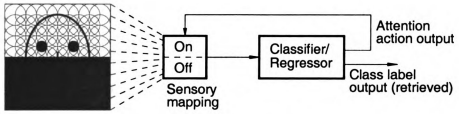
independence, clustering ability, filter extraction from natural images, over-complete estimation, real-time applicability and data compression ability. We compare the expressive power of LCA with PCA in original space, subspaces and in high dimensions. We demonstrate the unsupervised clustering ability of LCA against PCA for real-world face data. We show the use of LCA to derive super-Gaussian independent components faster than existing adaptive ICA methods in high dimensions. Further, we derive spatial LCA filters for natural images. The features derived by LCA are edge filters for whitened natural images and LOG type filters for non-whitened natural images. Also, we discuss how PCA and symmetric super-Gaussian ICA are special cases of LCA. Various advantages and applications of LCA are also mentioned.

6.1 Introduction

The main motivation for lobe component analysis (LCA) comes from the need for good feature analysis techniques that are implementable in real-time for developing a sensory mapping architecture [51]. A sensory mapping architecture is a computational model for visual perception. Computational visual perception aims at developing computational models that simulate the perception phenomenon of the retina, lateral geniculate nucleus (LGN) and the primary visual cortex, which are considered to be the "feature detectors" for higher cognitive layers. Developing such computational perception models is a vital step in the development of a cognitive architecture based on the autonomous mental development paradigm [9]. One of the important goals of sensory mapping, which is a perception model, is to extract high-level features



(a) Learning session



(b) Performance session

Figure 6.1: Recognition under occlusion.

for receptive fields of different sizes and at different locations on the retina. This goal is important because humans have the ability to focus their attention on local areas of different sizes and at different positions in a scene. The reader is referred to [51] for how a sensory mapping developed from natural images can be used to pay attention to local areas on the scene and to recognize trained objects even though they are occluded. Figure 6.1 shows the capability which is sought through the use of a sensory mapping architecture. In Figure 6.1, in the learning session the sensory mapping learns the features of the entire face (indicated by "On/Off"). In the performance session a part of the face is occluded, and attention signals from higher layers allow the sensory mapping to deliver features for the non-occluded part of the face("On" part of the block), which are fed to the classifier. The classifier should be able to recognize the face when attention is paid to only the upper part of

the This is because the classifier was trained on receptive fields at different location and sizes in the learning session. Much of the cognitive models for perception stem from the work of Hubel and Wiesel [16], who showed that the spatial receptive fields of neurons in primary visual cortex (V1) are selective to location and orientation. We can assume that the properties of the cells that are observed are related to the properties of natural images due to which such receptive fields have developed. A justification for this comes from the kitten carousel experiment by Blakemore and Cooper [4] who reported that the visual cortex of kittens, that were constrained to see only certain orientations, was not sensitive to other orientations. Another justification is from Sur's recent work [38], which reported that auditory cortex of ferrets showed orientation selective cells upon rewiring the visual pathway to the auditory cortex. These evidences are motivating factors to study the properties of natural images and to develop sensory mapping architecture that can derive filters similar to those observed in the visual cortex. Moreover the architecture should be able to derive filters from natural images, just like how the filters of primate's cortex develop after seeing natural images. In this chapter we shall show how LCA can be used to extract orientation filters like those observed in the visual cortex.

For the computational visual perception models that deal with images as raw input, the use of automatic feature derivation techniques is imperative. Automatic feature derivation techniques gained popularity in the field of computer vision due to the advent of appearance based methods. The appearance based methods deal with high dimensional images as input. When operating in the high dimensional space of images, the curse of dimensionality comes into play. To mitigate this problem,

the well established discreet version of the Karhunen-Loéve expansion is used for dimensionality reduction. The error and entropy minimization (of the coefficients of the K-L expansion) properties of K-L expansion imply that it is an optimal way to compress information into lower dimensions. The proof of the optimality of K-L expansion can be found in [41, 13]. Though K-L expansion existed in signal processing since 1965, it appears that this expansion was applied to appearance based methods in computer vision in 1982. Murakami and Kumar [30] were the first to apply K-L expansion to sets of images to extract basis images, also called principal components in 1982. The power of K-L expansion also called principal component analysis (PCA) was realized when Sirovich and Kirby [37, 27] applied K-L expansion for face image representation in low dimension with good reconstruction capability in 1987. In 1991, Turk and Pentland applied PCA for face recognition [39]. With this, the popularity of appearance based methods and automatic feature derivation techniques grew.

Many application domains that deal with high dimensional data apply dimensionality reduction in early preprocessing stages to mitigate the curse of dimensionality. Principal Component Analysis is popularly used for linear dimensionality reduction. The derivation of the PCA is typically done by finding the direction which has the most variance first followed by a de-correlation of the data based on the estimated vector(s), and then finding the subsequent most varying direction(s). This step is repeated until the last principal component is estimated. This process leads to mutually orthogonal principal component vectors. It is often misunderstood that principal component vectors give the directions which have the highest variance. This is not true because, the most varying directions need not necessarily be orthogonal. This

observation was one of the factors that lead to a new feature analysis domain called lobe component analysis (LCA) discussed in Section 6.2.

Also, LCA is biologically motivated by sparse coding property which is considered as a technique for redundancy reduction coding strategy of the neurons as suggested by Barlow [1] and Field [10]. Sensory coding property of LCA is discussed in Section6.3.

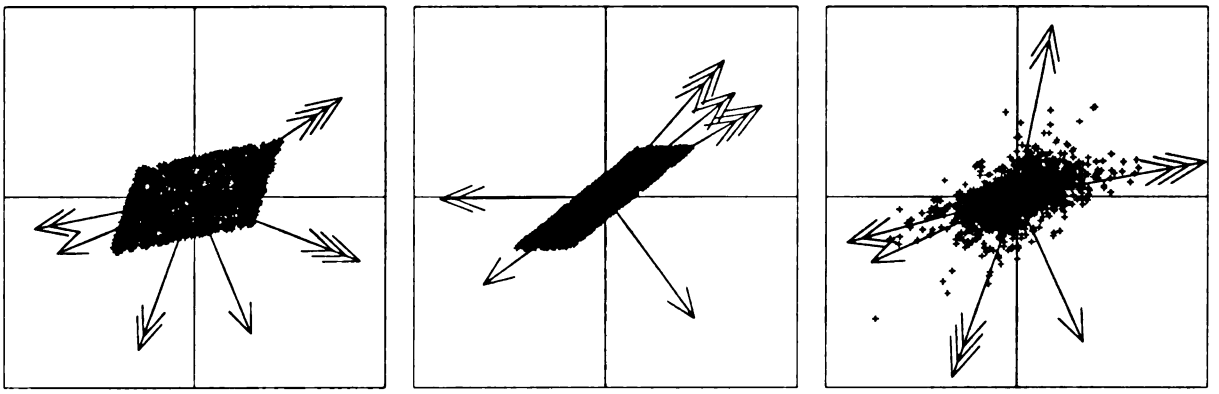
Another source of inspiration for LCA is from independent component analysis (ICA) termed by Comon [6] in 1994. The goal of ICA is to extract components which are as statistically independent as possible. ICA has its roots in blind source separation which is a well known problem in signal processing. ICA's popularity grew when it was used to address the blind source separation problem. The reader is referred to [24] for a survey on ICA. Most of the applications that use ICA to extract independent components from real world data can be thought of as extracting LCA features. We justify this claim later in the chapter in Section 6.6.1.

With the above discussion showing how LCA is related to both PCA and ICA, we proceed with showing how LCA can be used to extract features that are more useful and interesting than those extracted by PCA or ICA. The rest of the chapter is organized as follows. In Section 6.2, we present LCA, a criterion for expressive-ness, an adaptive algorithm for LCA and discussions of the algorithm. Section 6.3 discusses the properties of LCA like sparse coding, representation, expressive-ness, data compression, independence, clustering ability, over-complete estimation, real-time applicability and data compression ability. Section 6.4 discussed how PCA and super-Gaussian ICA are special cases of LCA. Section 6.5 presents results of various

experiments that demonstrate the properties of LCA when compared to PCA for artificial and real world data. Section 6.6 presents results of the similarity of LCA and ICA for real-world data. It also presents results for LCA derived spatial filters for natural images. In Section 6.7 we conclude with a discussion of LCA and its applications. We shall defer the applicability of LCA to sensory mapping architecture to the next chapter.

6.2 Lobe Component Analysis (LCA)

To study a data distribution better, it would be useful to derive expressive features which show the way the data is distributed in the sample space. Imagine a d-dimensional space in which the data is distributed along k major directions. We would get much insight into the shape of the data distribution if we could estimate these k directions in the d dimensional space. Imagining a 3-dimensional space with 5 major directions can help us visualize that these 5 major direction are of most interest in analyzing the data. We call the feature vectors which are oriented along the directions of the lobes as *lobe components*. It can be observed that these lobe components need not necessarily be orthogonal. For high dimensional cases, these features can be used to understand the distribution of the data as to where the data clusters are or in which direction they are oriented. These directions are more meaningful in terms of expressiveness than the directions given by PCA which are orthogonal, and not necessarily informative of the orientations of the data distribution. Therefore, we introduce a new domain of feature analysis called *lobe component analysis* (LCA).



(a) Case I (b) Case II (c) Case III

Figure 6.2: Illustration of LCA versus PCA and ICA.

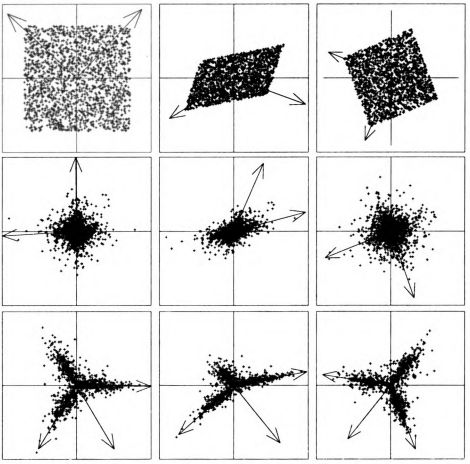
Definition *Lobe Component Analysis* is defined as a technique to extract features, not necessarily orthogonal, which best express the data based on a criterion.

6.2.1 Illustration

To visually illustrate the difference between PCA, ICA, and LCA we show $2D$ distributions with PCA, ICA, and LCA derived vectors in Figure 6.2. In this figure, PCA vectors are shown with single arrow head, ICA vectors with double arrow head and LCA vectors with triple arrow head. In Figure 6.2(a) LCA features were calculated in whitened space and transformed back to original space for sub-Gaussian Uniform distribution whose signals are mixed. In Figure 6.2(b) LCA features were calculated in original space. In Figure 6.2(c) LCA features were computed in whitened space and transformed back for super-Gaussian Diamond distribution whose signals are mixed. LCA vectors computed using both the methods for both the distributions resulted in the same vectors.

As can be seen in Figure 6.2, the set of directions which are visually more expressive of the direction in which the data is distributed are given by LCA rather than by PCA or ICA. For a distribution which is more narrower like in Figure 6.2(b), we can see how LCA is not constrained to extract orthogonal features unlike PCA, and derives features which are more indicative of the directions in which the data is distributed. ICA on the other hand extracts directions which are parallel to the edges of the parallelogram shaped distributions. As ICA uses higher order statistics to estimate the directions of the independent components, the signals had to be whitened first and the ICA vectors were computed and transformed back to original space. LCA does not have the constraint of working only in whitened spaces unlike ICA (though some methods have been proposed for ICA which do not need prewhitening) , or working only in non-whitened space unlike PCA (any orthogonal set of vectors are valid solutions for PCA in whitened space) . The LCA vectors shown in Figure 6.2 were computed in both whitened and original space to obtain similar vector directions which are most expressive of the direction in which the data is distributed. Figure 6.2(c) shows the PCA, ICA, and LCA vectors for a super-Gaussian distribution. You can notice how LCA and ICA extract features which have very similar directions (sign does not matter). The reason for this will be discussed later in the chapter.

We show the LCA vectors for sub-Gaussian Square distribution, super-Gaussian Diamond and Triangle distribution. The signals generated were called original signals. They were mixed using a mixing matrix to generate mixed signals and were then whitened to obtain whitened signals. The LCA vectors for each of the three spaces (original, mixed and whitened), shown in Figure 6.3, demonstrate how LCA can



(a) Original signals (b) Mixed signals (c) Whitened signals

Figure 6.3: Illustrating the expressiveness of LCA derived features.

operate both in whitened and non-whitened spaces to derive vectors which are visually most expressive of the data distribution. We use the term *lobe* to indicate the space covered by the samples which are closest to a particular vector. So, each vector is associated with its own lobe. Though the notion of "lobe" is more evident w.r.t the Triangle distribution, we use the same term for any distribution in general to indicate the set of mutually exclusive spaces which correspond to each LCA vector derived. In Figure 6.3 the first row shows Square distribution generated by two Uniform random variables, the second row shows Diamond distribution generated by two Laplacian

random variables, and the third row shows Triangle distribution generated by using a function of Laplacian and Chi random variables.

Details for the generation of the data distributions are given next.

Square Distribution

The square distribution was generated by using two uniform distributions from 0 to 1. The graphs shown are generated with 2000 samples. The mixing matrix used for the mixed signals shown in Figure 6.3 is $A = \begin{pmatrix} 1 & 0.2 \\ 0.2 & 0.6 \end{pmatrix}$

Diamond Distribution

Two Laplacian distributions were used to generate a diamond distribution with $\sigma = 0.87$ The mixing matrix used is A.

Triangle Distribution

The two distributions used to generated the data are Laplacian $L(0.3)$ and Chi $Chi(n = 2, \sigma = \sqrt{(5)})$distribution. The Chi distribution was shifted by 0.2 units to the center. So, let $s1 = L(0.3)$, $s2 = Chi(2, \sqrt{(5)}) - 0.2$,

$S = \begin{pmatrix} s1 \\ s2 \end{pmatrix}$

6.2.2 Measure of expressiveness: collective best match criterion

We formally define a criterion for expressiveness of the data as the maximum of ϕ defined in Equation 6.1.

Definition *A match between a sample and a feature vector* is defined as the square of the dot product of the sample vector and the unit feature vector.

Definition *A best match of a sample* (or sample vector) is defined as the maximum of the matches between the sample vector and all the feature vectors. We call the feature vector with which a sample has the best match as the *best expressive feature* for that sample.

Definition The *Collective Best Match (CBM)* criterion is defined to be $\max \phi$, where ϕ is given in Equation 6.1. A discreet version of the criterion is given in Equation 6.2.

$$\phi(V_1, V_2, ..., V_d) = \int_{-\infty}^{\infty} \max_j \left[x^T \frac{V_j}{\|V_j\|} \right]^2 p(x)dx, \; j = 1, ...d \tag{6.1}$$

$$\phi(V_1, V_2, ..., V_d) = \frac{1}{N} \sum_{i=1}^{N} \max_j \left[X_i^T \frac{V_j}{\|V_j\|} \right]^2, \; j = 1, ...d \tag{6.2}$$

where X_i is a data point, V_j is a feature vector, d is the number of feature vectors we select to express the data and $p(x)$ is the p.d.f of the distribution of x. The range of d is 1 to ∞.

The CBM criterion aims at finding the collection of feature vectors that maximizes the sum of the *best matches* of all the samples. By this definition, we can see that every

sample vector has its *best expressive feature vector.* The group of samples which have the same best expressive feature vector are defined to belong to a *lobe.* It is evident how this criterion can lead to feature vectors that best express the directions of the lobes of data. In other words, these are the directions which gives the directions in which the data is distributed. The set of vectors V_j maximizing ϕ is said to form the *LCA feature set* for the given data X.

Another way of looking at the CBM criterion is that it tends to maximize the local variance of the samples in terms of the dot product. We know that variance is given by

$$\sigma^2 = \int_{-\infty}^{\infty} (x - \bar{x})^2 p(x) dx \tag{6.3}$$

The variance is measured in terms of the Euclidean distance from the mean. In the case of CBM, the variance is measured in terms of the square of the *deviation* from its local representing unit vector. So, we say the CBM criterion maximizes the *local variance in terms of the dot product.*

In the next section, we propose an algorithm for deriving this LCA feature set based on the CBM criterion.

6.2.3 LCA algorithm

The LCA algorithm that tries to find the LCA features set for a given sample set is as follows.

1. $V_i(0) = X_i$ and $n_i = 1$, $i = 1, 2, ..., d$.

2. For $t = 1, 2, ...,$ do

(a) $j = \max_i\{\mid V_i(n_i - 1) \cdot X_t \mid / \|V_i(n_i - 1)\|\}$

(b) Compute $V_j(n_j)$ as follows

$$V_j(n_j) = \frac{n_j - 1 - \mu(n_j)}{n_j} V_j(n_j - 1) + \frac{1 + \mu(n_j)}{n_j} \frac{(X_t \cdot V_j(n_j - 1))}{\|V_j(n_j - 1)\|} X_t$$

(c) $n_j = n_j + 1$

(d) for other j's, $V_j(n_j) = V_j(n_j - 1)$.

where, $\mu(n)$ used in Step (2b) is the amnesic parameter used to speed up the convergence of the algorithm to the desired expressive features. The reader is referred to [48] for details on the amnesic parameter. Unless specified otherwise, we shall assume X_i's to have zero-mean, though this is not a constraint for the algorithm. For real-time algorithms, we can incrementally estimate the mean. This algorithm is a *one-winner takes all* algorithm which is the primary reason for its fast speed. Later, we found that a one-winner takes all algorithm was proposed for overcomplete ICA to extract super-Gaussian components [40]

6.2.4 Discussion of the LCA algorithm

The algorithm proposed is a greedy algorithm, with each vector V_j establishing its region of influence due to Step (2a). Each V_j has a count n_j as to how many samples X_i contribute to its orientation, and the strength of V_j is given by n_j or the number of contributors it has. Step (2b) of the algorithm doesn't distinguish between $-X_i$ and X_i, and the two X_i terms cancel their sign out. So, we can consider X_is to be in

the upper (in the direction of the existing V_j) hemi-hyper sphere of the distribution. So, the contribution of a X_i will always tend to pull V_j in the direction of the data distribution in the region of influence of V_j, i.e., where most of the contributing X_i are concentrated. Also, because each of the V_js has its own region of influence, which is mutually exclusive, the V_js do not find the same directions unless they are not updated by any sample. The algorithm always converges, as $n \rightarrow \infty$ because the second term in (2b) tends to zero, as $n \rightarrow \infty$.

6.2.5 Intuitive proof of expressive power of the algorithm

In this section we will prove that the LCA algorithm finds the directions which are most indicative of the direction of the lobes or clusters in the data under the assumption of good initialization. By good initialization, we mean that the LCA vectors should avoid grouping together in the same cluster. This assumption need not always be true as we do not yet have a criterion to indicate what demarcates the clusters. But this assumption is closely met because of the one winner takes all strategy adopted by the algorithm. Each LCA vector will have its own region of influence which represents the region covered by that lobe. Now, under the assumption of good initialization, we will show how an LCA vector derived is the local first principal component for the lobe of the LCA vector. We know that the first principal component gives the direction along which there is maximum variance in the data. We define the local first principal component as the direction along which there is maximum variance in the lobe. Naturally, this direction is most expressive of the lobe.

Let X_j be the data samples that contribute to updating the LCA vector V_j. A stabilization point for V_j is reached as $n \rightarrow \infty$ because of the update rule of the algorithm. So, the membership of the lobe of a particular vector V_j stabilizes as $n \rightarrow \infty$. Under the assumption of good initialization we will assume that most of the samples in X_j that contribute to updating the vector V_j are members of the lobe j at the stabilization point. Let C_j be the covariance matrix of the samples X_j

$$C_j = X_j {X_j}^T \tag{6.4}$$

The first eigen vector can be calculated as the eigen vector corresponding to the maximum eigen value in the following equation.

$$\lambda_j u_j = C_j u_j \tag{6.5}$$

where u_j is the eigen vector of the matrix C_j and λ_j is the corresponding eigen value or the strength of the eigen vector, i.e., $\lambda_j = \|u_j\|$. Replacing u_j of (6.5) with its incremental estimate at each time step i, the equation for $v_j = \lambda_j u_j$ becomes,

$$v_j(n) = \frac{1}{n} \sum_{i=1}^{n} X_j {X_j}^T u_j(i) \tag{6.6}$$

where $v_j(n)$ is the estimate of v_j at time n. The above equation can be rewritten in an incremental form as,

$$v_j(n) = \frac{1}{n}\sum_{i=1}^{n-1} X_j {X_j}^T u_j(i) + \frac{1}{n} X_j(n) X_j(n)^T u_j(n)$$
$$= \frac{n-1}{n} v_j(n-1) + \frac{1}{n} X_j(n) X_j(n)^T u_j(n)$$

The estimate of the eigen vector $u_j(n)$ can be got back from $v_j(n)$, as $u_j(n) = v_j(n)/\lambda_j$. For incremental estimation of $u_j(n)$, we approximate $u_j(n)$ by using the previous estimate of v_j at $(n-1)^{\text{th}}$ update as $u_j(n) = v_j(n-1)/\|v_j(n-1)\|$. So, the incremental update rule for $v_j(n)$ becomes

$$v_j(n) = \frac{n-1}{n} v_j(n-1) + \frac{1}{n} X_j(n) X_j(n)^T \frac{v_j(n-1)}{\|v_j(n-1)\|} \tag{6.7}$$

The vector u_j that solves the (6.5) gives the first principal component with maximum variance for the data $X_j(i)$.So, the vector $v_j(n)$ which we derived from (6.5) is the first principal component or the direction with most variance for the lobe associated with vector V_j.

The derivation of (6.7) is motivated by statistical efficiency. An unbiased estimate $\hat{Q}$ of the parameter Q is said to be the *efficient estimate for the class D of distribution functions* if for every distribution density function $f(u, Q)$ of D the variance $D^2(\hat{Q})$ (squared error) has reached the minimal value given by

$$D^2(\hat{Q}) = E[(\hat{Q} - Q)^2] \geq \frac{1}{n \int_{-\infty}^{+\infty} \left[\frac{\partial \log f(u,Q)}{\partial Q}\right]^2 f(u, Q) du}. \tag{6.8}$$

The right side of inequality (6.8) is called Cramér-Rao bound. It says that the efficient estimate is one that has the least variance from the real parameter, and its variance is bounded below by the Cramér-Rao bound. For example, the sample mean, $\bar{w} = \frac{1}{n}\sum_{i=1}^{n} Y_j(i)$, is the efficient estimate of the mean of a Gaussian distribution with a known standard deviation σ [11].

If we define $Y_j(i) = X_j(i)X_j(i)^T u_j(i)$, $v_j(n)$ in (6.6) can be viewed as the mean of "samples" $Y_j(i)$. That is exactly why our method is motivated by the statistical efficiency in using averaging in (6.6). In other words, statistically, the method tends to converge most quickly or the estimate has the smallest error variance given the currently observed samples. Of course, $Y_j(i)$ is not necessarily drawn from a Gaussian distribution independently and thus the estimate using the sample mean in (6.7) is not strictly efficient. However, the estimate $v(n)$ still has a high statistical efficiency and has a fairly low error variance as we will show experimentally.

To speed up the convergence property of the (6.7), we use an amnesic parameter [48] $\mu(n)$ to forget the old data.

6.3 Properties of LCA

In this section we discuss some of the properties of LCA based on the CBM criterion. We shall give experimental results supporting the properties discussed below in Sections 6.5 and 6.6.

Sparse coding

Sparse coding [10] [33] is a technique of representing the data by using only a few vectors. Sparse coding is between the extrema of global coding employed by PCA or Factor analysis and local coding employed by vector quantization. In sparse coding each observation is reconstructed using just a few features from a larger collection. We can use the best expressive feature of a sample to code that sample, which implies the applicability of LCA for vector quantization. The more the number of features the better its reconstruction ability. In fact, one can achieve loss-less reconstruction using all the features in the LCA feature set. Sparse coding is motivated by biological evidence [12] that suggests that the human brain uses sparse coding. It is said to have advantages like fast learning, good representation, better adaptiveness, generalization and tolerance to faults.

Representation

LCA does not necessarily find mutually orthogonal directions. So, there is a problem of representing the data where we do not have orthonormal basis. We do not need orthonormal basis vectors as long as we have a method to reconstruct the original data from the representation. Let V be the LCA feature set. The representation Y of data X in the LCA space is given in (6.9)

$$Y = W^T X \tag{6.9}$$

To reconstruct the data back from LCA space, we use matrix inverse as in (6.10).

$$X = (W^T)^{-1}Y \tag{6.10}$$

Dimensionality reduction can be applied to LCA. Dimensionality reduction can be done in two different ways - one, set $d = r$ in the LCA algorithm, where r is the dimensionality of the subspace, and two, drop the V_js with lower number of hits, thereby neglecting less populated clusters. For high-dimensional data sets with noisy data, the second alternative can be adopted. To reconstruct the original signals in the reduced dimensionality case, a pseudo-inverse can be applied. From (6.9) and (6.10), it can be seen how we have loss-less reconstruction ability when representing the data in the original dimension LCA space.

Expressive ability

The LCA algorithm extracts features that are expressive of the directions of the lobes or clusters of data. They give the directions along which most data is concentrated. The collective best match criterion ensures that the algorithm does not miss out on the clusters which are closer to origin than some clusters which have their centers far away from origin. The ability of the algorithm to quickly establish their regions of influence or the lobe boundaries enables it to estimate the lobe directions better. The algorithm yields expressive directions as shown in Figure 6.3. A comparison of the expressive power of LCA against PCA is given in Section 6.5.1.

Clustering ability

A direct consequence of the expressive ability of the LCA algorithm based on the CBM criterion is its power to identify clusters in the distribution. These clusters are the lobes identified by the LCA algorithm. So, for applications where the classes have less intra-class variability, an unsupervised clustering algorithm based on LCA will be a good choice. For appearance based applications that deal mostly with images, the pattern represented by an image remains the same even when factors like contrast, brightness change. Image points along the same direction typically represent variations in brightness or contrast. In the high dimensional space, it is the angular deviation of the vectors that creates different patterns. The CBM criterion measures variance based on the deviation and hence is a better choice for clustering image data. Results on clustering experiment for real-world face data are given in Section 6.5.2.

Independence

Though independence is not a direct property of LCA, LCA can be used to extract super-Gaussian independent components from the data. Independence is a well defined concept that implies that the p.d.f of the joint distribution of a set of random variables can be expressed as a product of the p.d.fs of the marginal distributions as follows.

$$f(x_1, x_2, ..., x_n) = f_1(x_1)f_2(x_2)...f_n(x_n) \tag{6.11}$$

Estimating the p.d.f for high dimensional data is impracticable. So, other methods are used to estimate the independence. For super-Gaussians, the independent com-

ponents are extracted by finding the directions of the tails of the distribution. This is typically done using higher order cumulants like kurtosis [18, 22, 23, 7]. LCA uses a similar approach to detect super-Gaussian independent components as will be proved later in Section6.4.

To our knowledge, the LCA algorithm proposed is the fastest of existing adaptive ICA algorithms to extract super-Gaussian independence components.

Over-complete estimation

The LCA algorithm can determine over-complete (when the number of LCA features is greater than the dimensionality of the data) feature set. For high dimensional data, there exist many quasi-orthogonal independent components. Over-complete estimation methods help to detect such components. Note that the range of d in the LCA algorithm can be set from 1 to ∞, which implies the capability of LCA for over-complete estimation.

Real-time applicability

The LCA algorithm proposed is incremental and very fast, as it just updates one best expressing feature vector per sample. The convergence speed is also fast as seen in Figure 6.9 because of its greedy nature. This makes LCA a good choice for real-time applications.

Data compression

The use of LCA for data compression is evident because of its sparse coding property. In the extreme case, the vector quantization can be used when each high dimensional sample can be represented by just one LCA vector. When using LCA for data compression we will be having a "code-book" containing the set of, say d, LCA vectors which are derived. Each sample in the input space is represented by a set of r numbers each corresponding to the top r *best match* vectors among the d LCA vectors. To reconstruct, one would use the code-book and apply a inverse transformation to get back the sample in the original space.

6.4 PCA and ICA - special cases of LCA ?

LCA can be used to extract PCA vectors.

Theorem 6.4.1 *LCA computed in residual space one at a time computes PCA vectors.*

Proof: If $d = 1$ in the LCA algorithm, it extracts the first principal component. To extract other principal components, we should de-correlate the data w.r.t the estimated vector(s) and then find another vector with $d = 1$. Repeat this process until you get the desired number of principal components. LCA extracts the highest variability in the region or lobe which it is allocated. If $d = 1$, the lobe spans the entire space and hence extracts the first principal component direction. ■

Theorem 6.4.2 *LCA performs super-Gaussian ICA in whitened space.*

Proof: Super-Gaussians are defined as those random variables that have positive kurtosis. Distributions which have positive kurtosis are characterized by a heavy tail. Upon whitening, the tails of the distributions are indicative of the lobe directions, as they are the most expressive local directions in the sample space. LCA finds these lobe directions which are most expressive of the data distribution and hence can perform super-Gaussian ICA in whitened space. ■

In the next section we present various results of experiments to demonstrate the properties of LCA. We present the results comparing LCA with PCA in Section 6.5 and show the results with ICA in Section 6.6.

6.5 Results with PCA

The features derived by LCA algorithm have various desirable properties. We show the following results to demonstrate the superiority of LCA over PCA.

- We show that LCA performs better than PCA in the new CBM error criterion to express the data distribution in Section 6.5.1. Also, the effect of reducing the dimensionality of the data using both LCA and PCA derived features is shown.

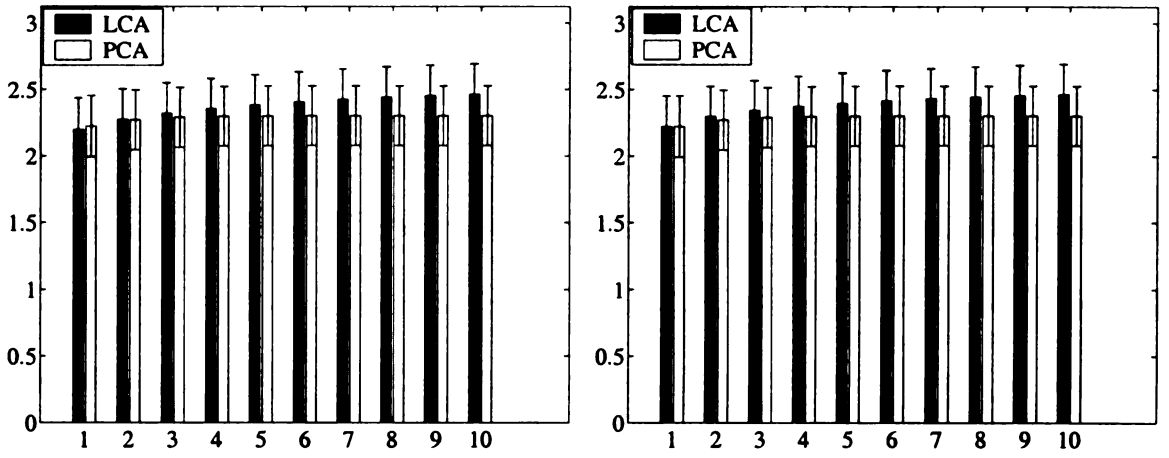
- We show experimental results for the use of LCA in clustering high dimensional real-world data, and compare the results with PCA in Section 6.5.2. Also, we show the 10 *LCA faces* and the top 10 *Eigen faces* for the FERET data set.

- We show the data compression ability of LCA over PCA with respect to the mean square error criterion, which demonstrates LCA's sparse coding property.

6.5.1 Dimensionality reduction - PCA vs LCA

PCA derives a linear transformation to a subspace such that the resulting data representation is optimal for reduced dimensions in the least square error sense. This linear transformation is composed of a set of eigen vectors of the covariance matrix of the data set. The eigen vectors are ordered based on their eigen values and to reduce the dimensionality of the data, the top few eigen vectors corresponding to the highest eigen values are selected. In online real-time applications, the PCA vectors have to be estimated incrementally. For high-dimensional appearance based vision applications, the computation of the covariance matrix becomes cumbersome. This can be seen by considering a data set of images of size 64×88. Each image is considered as a vector. So, the dimension of each sample becomes 5632. Computing the covariance matrix would involve estimating $d * (d + 1)/2 = 15862528$ entries, where d is the dimensionality of the sample space. Moreover, computing the covariance matrix incrementally would mean incrementally estimating over 15 million entries each time a sample comes in. To avoid such computational complexity, a fast (in computational time and convergence speed) incremental PCA algorithm called CCIPCA [48] was proposed which doesn't involve estimating the covariance matrix. Assume that we want to estimate only the top k eigen vectors, this algorithm would need one update of all the k eigen vectors of dimension d per new sample. The LCA algorithm on the other hand updates only one d dimensional vector per new sample and does not involve any covariance matrix estimation.

The vectors derived by the LCA algorithm can be ordered based on two criteria



(a) Select top r LCA features

(b) Compute only r LCA features

Figure 6.4: Mean of ϕ and its variance for LCA and PCA for dimensionality reduction.

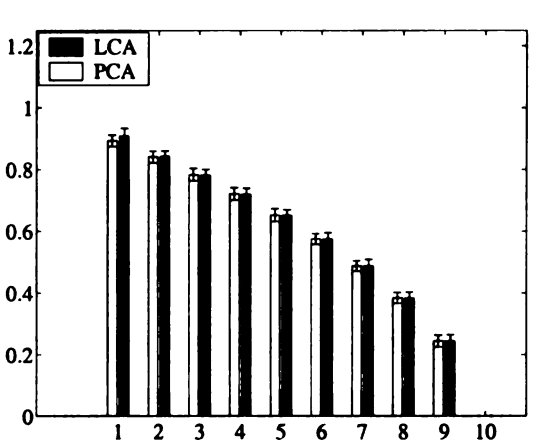
- one, by the number of samples n_j that contributed towards the derivation of the vector V_j, and two, by the magnitude of the vector. We call n_j, the *number of hits* for the vector V_j. Ordering the vectors based on the *number of hits* gives more importance to the vector which expresses the largest cluster. Ordering the vectors based on the magnitude of V_j gives more importance to the vector that represents samples farther away from the origin. We chose the first criterion for ordering the LCA derived vectors. This ordering can be used to select the top few vectors for reducing the dimensionality. Another method for reducing the dimensionality is by setting d, the number of LCA vectors to be derived, equal to the dimensionality of the subspace.

For testing which of the methods (LCA or PCA) expresses data better in the original dimension and in reduced dimensions, we generated a 10-D zero mean uniform distribution, applied a random linear transformation, and used LCA and PCA to derive feature sets. To reduce the dimensionality for LCA, we used the two methods

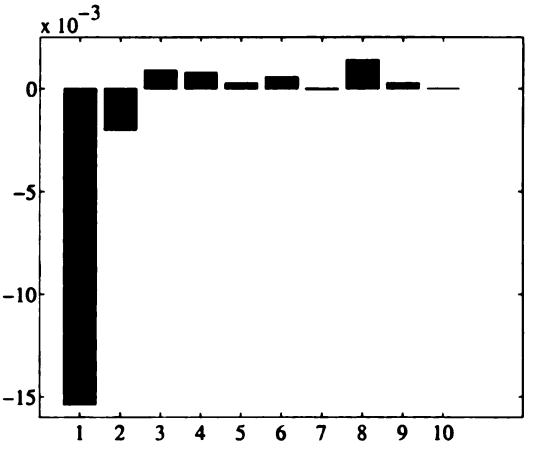
described above - one, derive 10 LCA vectors and select the top few from those 10 vectors and two, derive only r LCA vectors for reducing the dimensionality of the data to r.

We computed the PCA vectors using a batch algorithm. The CBM criterion's ϕ defined in (6.1) was computed for the feature set derived by PCA and LCA and was averaged over 50 runs. Figure 6.4 demonstrates the expressive power of the LCA algorithm over PCA for both methods of dimensionality reduction for LCA described above. Figure 6.4 shows the mean ϕ and its variance . The higher the bar, the better is the expressiveness of the feature set derived. From the figure it is clear that LCA extracts features that are more expressive than the features derived by PCA. In Figure 6.5 x-axis shows the reduced dimensionality. In Figure 6.5(b) and Figure 6.5(c), a bar above zero indicates that LCA performs better than PCA.

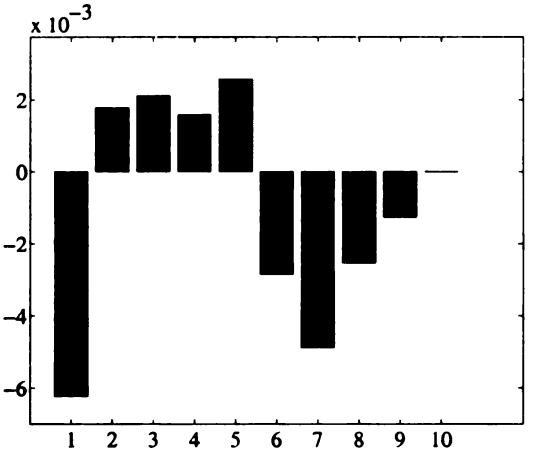
Also, we compared the mean-square error (MSE) of the reconstruction using PCA and LCA. We have to keep in mind that LCA was not designed to have optimal reconstruction in sub-spaces, but only good expressiveness of the data distribution. For dimensionality reduction using LCA, the top $r < d$ vectors were selected from $d = 10$ dimensional uniform distribution. The results were averaged over 50 runs and both variance and mean of the MSE error are analyzed in Figure 6.5. Also, loss-less reconstruction ability of both PCA and LCA in Figure6.5. It can be seen how LCA performs equally better at the MSE criterion for which PCA is optimal. LCA performs better than PCA for reduced dimensionality from 3 to 9 for uniform distribution. It is evident that for dimensionality of 1, the first PCA vector captures most of the variance and hence performs better. Also, we have shown in Section 6.4



(a) MSE Error



(b) MSE(PCA)-MSE(LCA)



(c) Var(PCA)-Var(LCA)

Figure 6.5: Mean-square error (MSE) results for PCA and LCA.

how LCA and be used to derive the first PCA vector (all the PCA vectors in general), if only the first dimension is needed.

6.5.2 Clustering experiment on faces

Face recognition has become a hot research topic ever since the usage of PCA to reduce the high dimensionality of face images. The eigen vectors obtained using PCA on a face data set form eigen faces [39]. Many face recognition algorithms use eigen faces in their classifiers. In this section, we do not want to get into the recognition experiments as the recognition rate depends on the type of classifier one uses to recognize faces besides the dimensionality reduction technique. So, we adopt another illustrative technique to demonstrate the expressiveness of the LCA algorithm over PCA for high dimensional face data.

We took the FERET [35] data set consisting of frontal views of 457 subjects. Most of them have two views while 34 of them have four views and two of them have one view each. There are totally 982 images of size 64×88. So, the dimensionality of the data is 5632. *Sample-to-dimension ratio* is defined as n/d, where n is the number of samples and d is the dimensionality of the sample space. The lower this ratio, the harder it becomes to statistically estimate features. The FERET data set has a very low sample-to-dimension ratio of $982/5632 = 0.7$, which makes it a tough problem.

To illustrate the expressiveness of LCA over PCA, we measured the clustering power of both the algorithms in terms of the clustering error. We define *clustering error* of a feature set to be the number of classes which have at least two images that

do not have the same *best expressive feature* in the feature set. Results using this definition assume that samples in the same class are naturally clustered together.

We derived the top 10 features using PCA and LCA, and performed the following experiment. A data sample X_i is taken and its *best expressive feature* V_j is found as in Step (2a) of the LCA algorithm. We assign X_i to j^{th} cluster. The error in clustering is measured by the number of classes that are *split up*. A class is defined to be split up if there exist two or more samples that do not share the same best expressive feature. Each class is defined as the set of images belonging to a person. We repeated this experiment for each of the dimensions from 2 to 10. For LCA, the dimensionality reduction was done by selecting the top few features from the set of 10 features that were computed. In Figure 6.6 the dimensionality of original data

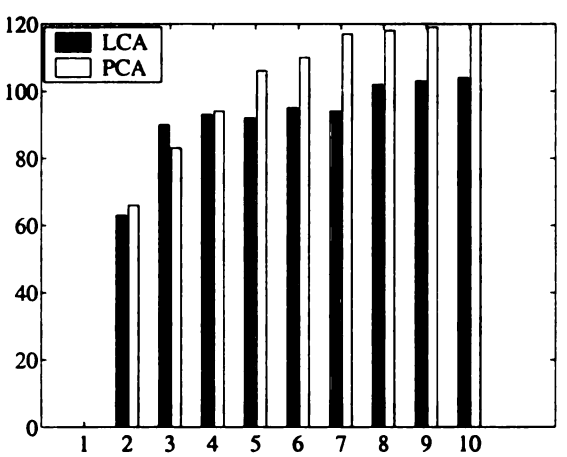


Figure 6.6: Clustering Error for PCA and LCA on FERET Data set.

is 5632. The x-axis shows the reduced dimension and y-axis shows clustering error in terms of number of classes which are split-up. Total number of classes = 457. It can be seen from Figure 6.6 that using the LCA derived features, we have lesser number of split-ups of the classes than we have when using the PCA derived features

(a) First 10 Eigen faces.

(b) First 10 LCA faces.

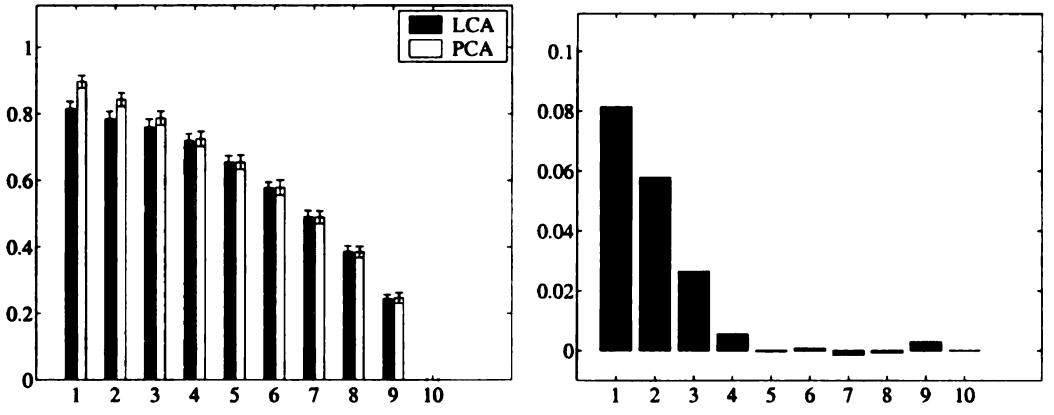
(b)

Figure 6.7: Eigen faces and LCA faces

except for the reduced dimension 3. This can be due to the method of selection of feature vectors. Discarding the other 7 expressive features could have caused the samples, which had the best expressive feature among those 7 features, to be distributed among the remaining 3 feature vectors. For all other dimensions the performance of LCA was better than the PCA. Thus, we have shown experimentally that the LCA features are better in clustering the data than the PCA features for high dimensional real world data. This result gives an experimental support to our claim that *LCA expresses a data distribution better than PCA*. We name the features derived by LCA for the face data set as *"LCA Faces"*. The Eigen faces and LCA faces are shown in Figure 6.7. This result demonstrates the applicability of LCA using the CBM criterion for dimensionality reduction and clustering that are commonly used by various appearance based vision applications.

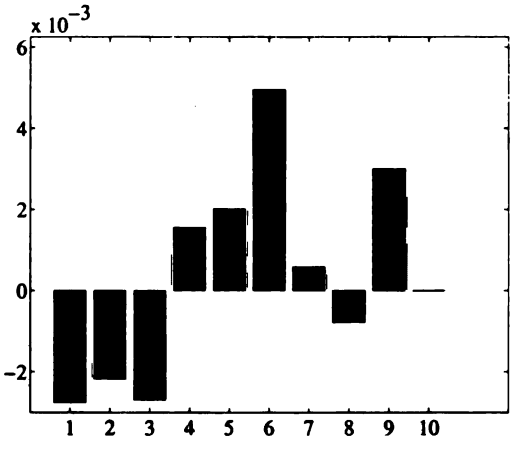
6.5.3 Data Compression

As LCA has sparse coding properties, we expect that LCA can be used for data compression in which samples are coded using a sub-set of features. PCA is also used in many data compression applications to reduce the dimensionality of the data because of its optimality of representation in subspaces based on MSE error. For sparse coding, we can select different set of features to represent the sample unlike the traditional way of selecting the most varying eigen features for PCA. Using LCA, we find the top r best matches for a particular sample vector and coded it based on the dot product value with those vectors. For this method, we need to store the "code-vectors" for reconstructing the sample to original space. We investigated the data compression ability of PCA and LCA. For PCA, we selected the eigen vectors with highest eigen values, and for LCA, we selected the top r best matches and we present the error in reconstruction in Figure 6.8. As we can see, the data compression power of LCA is better than PCA, especially for lower dimensions (sparseness). In Figure 6.8 the x-axis shows the dimension of the subspace. For Figures 6.8(b), 6.8(c), a bar above zero implies that LCA performs better than PCA. Using this coding scheme for LCA, we need to store which features were selected to represent the given sample. For applications which need to compress very high dimensional data into very low dimensions, the additional cost of storing the indices of the feature vectors is not very high.



(a) MSE Error

(b) MSE(PCA)-MSE(LCA)



(c) Var(PCA)-Var(LCA)

Figure 6.8: Error in reconstruction for PCA and LCA for data compression.

6.6 Results with ICA

We show the following results comparing LCA with ICA.

- In Section 6.6.1, we show how LCA algorithm can extract super-Gaussian independent components.
- In Section 6.6.2, we derive natural image filters using LCA and show how the most expressive directions in whitened natural images are edges. Also, we show how we can visualize LCA directions in non-prewhitened natural images using our algorithm.
- We illustrate how LCA algorithm can be used to find an over-complete set of features in Section 6.6.3.

6.6.1 LCA can extract super-Gaussian independent components

To support the claim made in Section 6.1 that extracting independent components for real world data can be thought of as extracting the *LCA features*, we show how LCA algorithm can be used to extract super-Gaussian independent components from the data. It is thought that real-world data is made up of a lot of super-Gaussian independent components. One more evidence to the above statement is presented in Section 6.6.2.

We generate 10 independent super-Gaussian signals (S) from Laplacian distribution. We mix those signals using a random transformation called the mixing matrix

A,

$$X = A * S \tag{6.12}$$

The goal of ICA is to estimate the matrix A or equivalently estimate a matrix W such that

$$W^T * M * X = W^T * M * A * S = P * S \tag{6.13}$$

where P is a matrix which can either permute or change the sign or magnitude of the independent components in S, and M is the whitening matrix usually used so as to improve the convergence properties of the algorithm used to estimate W. The reader is referred to [24] for more details on ICA. The independent components can be estimated up to a sign magnitude and permutation.

$$Y = M * X \tag{6.14}$$

Figure 6.9 shows the convergence speed for 25,000 samples in 25 dimensions over 50 runs for Laplacian sources compared to existing adaptive ICA algorithms. In this experiment, we give the pre-whitened signals Y as input to the LCA algorithm and to other existing adaptive ICA algorithms like Non-linear PCA Least-Squares (NPCA) [34] and Extended Bell-Sejnowski (ExtBS) [29]. The set of feature vectors derived by LCA are the columns of the matrix W. As we know the matrix A for artificial data, we can compute the ideal W matrix. We match the directions of the derived W with the ideal W and measure the deviation of the matched vectors in

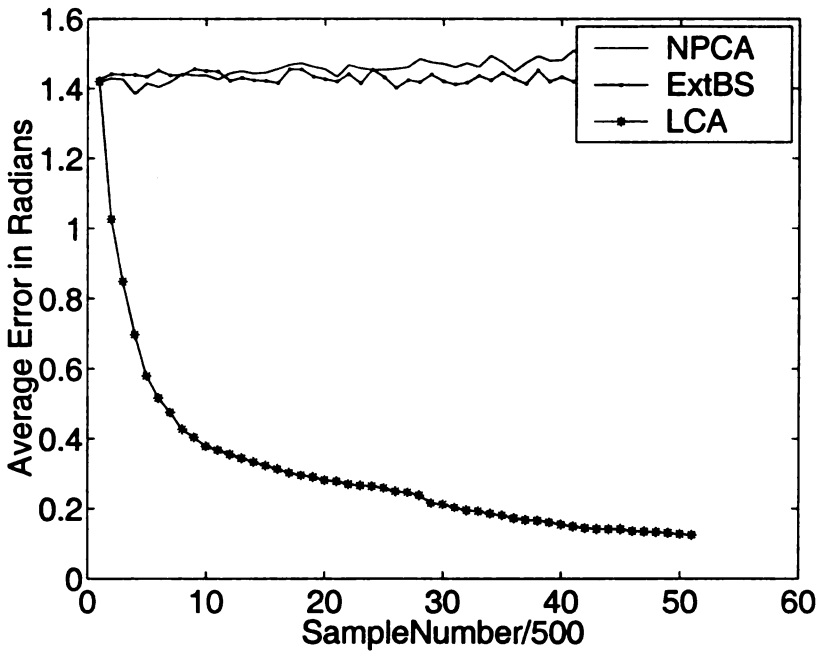


Figure 6.9: Convergence speed of LCA versus existing adaptive ICA algorithms.

radians. The mean of the angular deviations of all the vectors with their best match to the vectors in the ideal W matrix is computed as the angular error. We show the results of the convergence speed of the algorithms in Figure 6.9. This result serves to validate the claim that LCA algorithm extracts expressive features, which are super-Gaussian tails for whitened data.

6.6.2 LCA filters for whitened and non-whitened natural images

In this section we derive the expressive features for whitened and non-whitened natural images, and also substantiate the claim made in previous section by giving supportive evidence that real world signals are mostly composed of super-Gaussian signals. LCA algorithm can not extract sub-Gaussian independent components as they do

not have a heavy tail. It is shown in [3] that ICA extracts edge filters for natural images. If we can show that LCA also extracts edge filters for whitened natural images, then we can claim that those edge filters are super-Gaussian, and that natural images (which are real world signals) are rich in super-Gaussian components. We took 13 natural images [36], randomly sampled them to obtain small images of

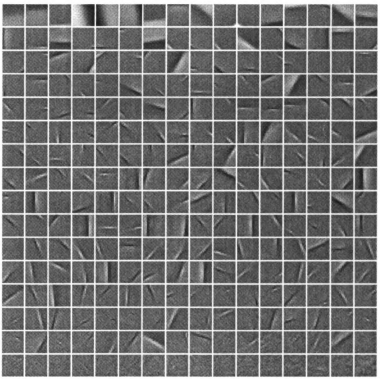


Figure 6.10: LCA derived features for natural images with pre-whitening.

size 16×16, and whitened them. We extracted LCA features (W matrix) from the whitened images and displayed the filters obtained (A matrix or the mixing matrix) as images in Figure 6.10. The filters shown are similar to the independent component filters derived in [3] for whitened natural images. The similarity is due to the fact that most of the independent components in natural images are super-Gaussian in nature, and super-Gaussians are characterized by a heavy tail. The direction of

these heavy tails are the LCA directions for whitened data. So, the LCA algorithm extracts super-Gaussian independent component filters for whitened signals. The

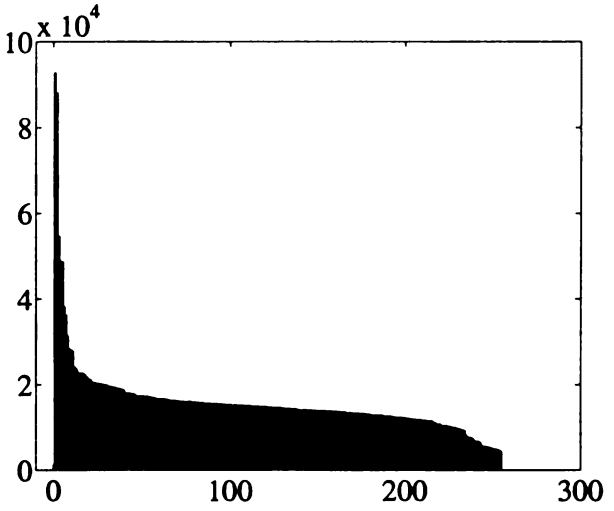


Figure 6.11: Number of hits for the LCA features for natural images.

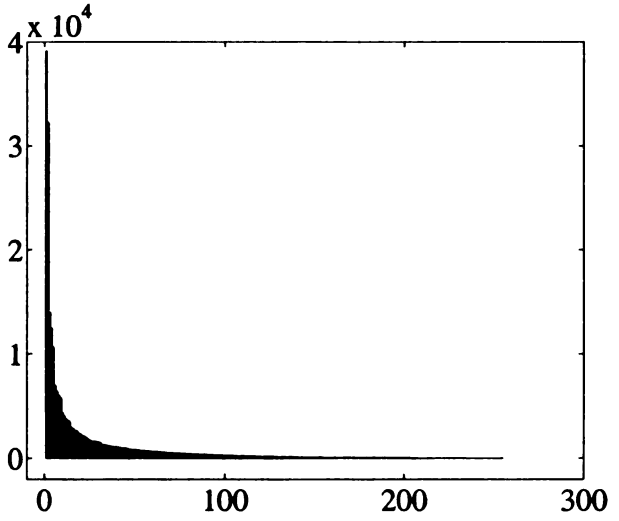


Figure 6.12: Distribution of Eigen values for a set of natural images.

distribution of the number of hits for each of the filters shown in Figure 6.10 is given in Figure 6.11. This distribution shows the strength of the clusters or lobes existing in whitened natural images. The strength of the eigen values for natural images is shown in Figure 6.12.

LCA does not have the limitation of working only with pre-whitened data unlike most of the ICA algorithms. So, this would allow us to visualize the LCA directions for non-prewhitened natural images. We ran the LCA algorithm on natural images with zero mean without pre-whitening. The results obtained are in the form of Laplacian of Gaussian (LOG) type filters. Some of them are bar-type and some others are circular. These filters are shown in Figure 6.13.

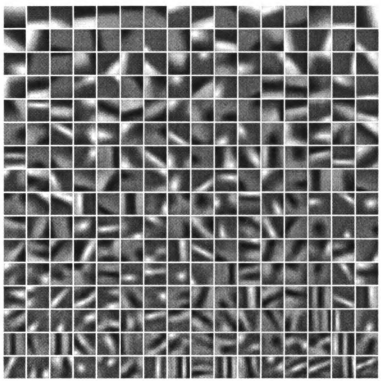


Figure 6.13: LCA derived features for natural images without pre-whitening.

6.6.3 Over-complete basis estimation

Unlike most ICA algorithms which can only detect as many independent components as the dimension of the data, the LCA algorithm can also be used to determine an overcomplete basis for whitened signals. For whitened distributions with only super-

Gaussian components, the complete set (the number of feature vectors is same as the dimensionality of the data) of LCA features are usually orthogonal with respect to each other and hence form a basis. For non-whitened signals, the LCA algorithm derives the LCA directions for cases where the number of features vectors to be estimated is greater than the dimensionality of the data.

The expressiveness of the LCA derived features for such cases can be seen from Figure 6.3. It can be seen that LCA extract 3 expressive directions in 2-dimensions. Also, assuming each of the 3 clusters in the Triangle Distribution belong to three different classes, the clustering ability of the LCA algorithm is evident.

6.7 Conclusion

We proposed a new feature analysis technique called lobe component analysis that extracts features which are more expressive of the directions in which the data is distributed. We proposed a criterion for expressiveness which is biologically motivated and compared its properties such as, expressive-ness in reduced dimensions, clustering, independence, estimating over-complete basis, and data compression with existing feature analysis techniques. This opens up enormous applications for LCA in unsupervised learning, coding, compression, estimating independent components, applicability in cognitive perception models, pursuit projection, ICA, etc. It was also shown how using ICA for real world data can be thought of as extracting LCA features, and we also showed evidence to the claim about how real world data is mostly composed of super-Gaussian independent components. We also showed the

applicability of LCA in the spatio-temporal domain to extract motion filters and to do motion segmentation. Also, the speed of convergence, incremental efficiency, completeness, and sparse coding properties makes LCA the only method practicable for use in the sensory mapping architecture for autonomous mental development.

Chapter 7

Motion Features and Feature Maps

The second law of Thermodynamics states that the universe tends towards maximum entropy or to attain a state of equilibrium. This well known physical rule of minimum potential energy or maximum entropy that any system would try to attain. This could be seen analogous to having as must independent components as possible in the spatio-temporal domain. The idea of time and the temporal features are linked to the other dimension we live in but rarely pay attention to. How come we remember the sequence of information in a roughly correct order in which they occurred if we do not have any temporal information stored in our brain. There is some information which links the spatial information to the temporal information. We can try to understand this relationship by attempting to study the spatio-temporal features that our brain could derive based on the maximum entropy principle. Before we analyze the spatio-temporal filters, we thought of considering the power of ICA and LCA to extract moving objects from moving background.

7.1 Applicability of LCA for motion segmentation

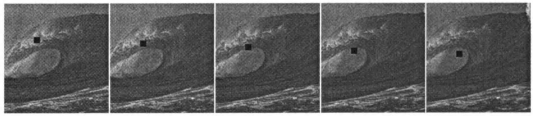
Motion segmentation, tracking have been tough problems for general settings. Model-based approaches exist for detecting motion of objects that have been modeled. But, can we extract moving objects, without modeling objects, from the statistical properties of a set of images? In this section we suggest a framework which allows us to extract moving objects from moving background from the image properties. As one can expect, detecting moving objects from moving background is a tougher problem than detecting moving objects from stationary background. Moreover, in this case, we will consider not just two frame, but a window of frames and try to detect the motion in that time-frame. We take a background natural image which shows the waves, and impose on it a black-filled square. To generate the sequence of images, we move the waves to the left, and move the object from top to bottom. This is the case of a moving object in a moving background. We generate five such images. We will consider each image as one signal. We assume that the signals are mixed with independent components(ICs), the ICs being the objects which move independently of each other. We try to extract the independent components from the image using LCA. We know that natural signals are mostly made up of super-Gaussian independent components. Figure 7.1 shows the components which are extracted by both ICA and LCA. They show how motion information has been captured by ICA and LCA components. More specifically LCA components separate out a static image (the first component of LCA) from its motion information. The LCA components that are extracted are ordered based on the number of hits received by each component, and so

are ordered based on their strength. It is interesting to observe how the strongest component has the information of the moving waves and the moving object captured in a single image. The other images can be considered as motion information which when used with the static image will give the images that are observed. The components extracted by ICA do not have an order. A fast batch algorithm for ICA called FastICA[22] was used to compute the ICA components. As the components extracted do not have sign information preserved, we show the strength of the coefficients of the components that are extracted as images in terms of the absolute value.

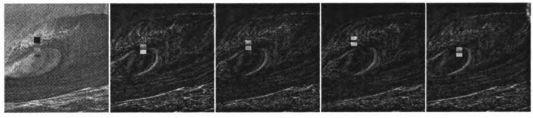
7.2 Motion filters in natural video

Here, we try to extract the independent component filters from natural video stream. As seen in the previous chapter, LCA can be used to extract super-Gaussian components from the data. We saw its results in the spatial domain. LCA can also be extended to the spatio-temporal domain to obtain motion filters. We can find the spatio-temporal filters underlying natural video stream. We extracted LCA features for natural video and derived LCA filters. We used a movie clip from SAIL[1] robot's outdoor navigation video. This video clip consisted of 663 frames, each frame being of dimension 240×360. Each frame was sampled to obtain 9600 small images, each of size 16×16. We grouped 2 such images in subsequent frames in time to form a video block. The LCA algorithm was run for 1.78 million frames or $890,000$ video blocks, and results were collected. Each frame's dimensionality is 256. So, the video

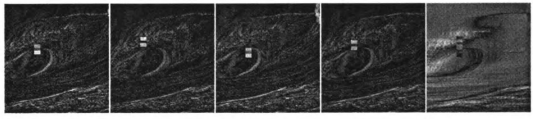
[1]SAIL or Self-organizing Autonomous Incremental Learner is a developmental robot at Michigan State University



(a) Input images



(b) LCA components



(c) ICA components

Figure 7.1: Motion segmentation using LCA and ICA.

block's dimensionality is 512. One part of the data with 26520 frames or 13260 video blocks was used to compute the pre-whitening matrix, and the same matrix was used on the rest of the data to pre-whiten the data. Each frame was locally deprived of its DC component and normalized, before the whole data was pre-whitened. Only top 95% of variation in the data was kept using PCA. The dimensionality was reduced to 353 dimensions. LCA algorithm was run on this dimension to get 353 filters each of dimension 512. To display the filters corresponding to the frame which they belong to, the first 256 dimensions were taken to be filters for frame 1, and the next 256 dimensions were taken to be filters for frame 2. The difference of the two filters was computed and plotted as motion filters for the video block in Figure 7.4. The ordering of the filters was done based on the *number of hits* $n(j)$, for each component V_j in the LCA algorithm. The filter with the most number of hits is displayed first and other subsequent filters are displayed in the row major order from left-top to right-bottom. Among the 353 filters, the first 352 filters are shown in a matrix form to fill the rectangular display. The filters for frame 1 and frame 2 are shown in Figures 7.2 and 7.3 respectively.

Most filters have a dark and bright edge or circular patch. These filters are motion filters indication motion in the direction from the black area to the white area. It can be seen from the figure that the motion filters detect motion in various directions. Around 29% of the filters detect horizontal or vertical motion. Other filters detect motion along various other directions.

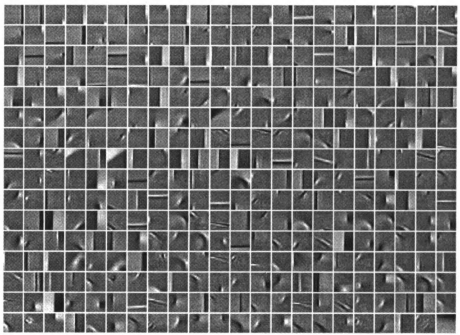


Figure 7.2: LCA filters for frame 1 of the video block.

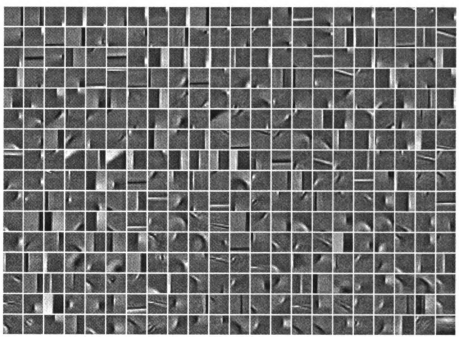


Figure 7.3: LCA filters for frame 2 of the video block.

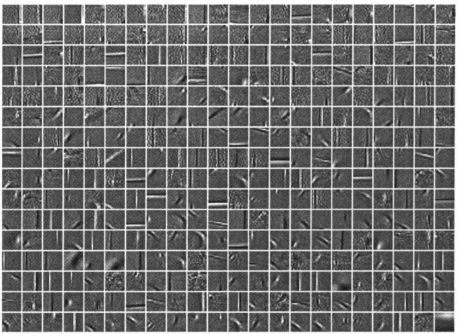


Figure 7.4: Motion filters for the video block.

7.2.1 Variation of eigen values in natural video block

For the experiment used to detect motion filters, we used video blocks, each consisting of two frames. Each frame has 16×16 pixels. Totally, there are 512 features in the video block. We computed Eigen Vectors and Eigen Values for a batch of 26520 frames or 13260 video blocks, and the length of the eigen values is shown in Figure 7.6. To reduce dimensionality, the magnitude of the eigen values are used to neglect the least varying dimensions in the Eigen space. For 512 dimensions, there are some directions whose variation is much higher than others. So, taking the sum of the eigen values gives more preference to the higher varying direction and so the number of eigen values that we would consider to obtain 95% variation is only 82. Instead if we take the ratio of the sum of the square root of the eigen values to the total sum of the square root of the eigen values, then we get to consider 353 eigen vectors. In

Figure 7.5, the X-axis shows the Number of Eigen values considered and the Y-axis shows the % of variation captured. captured.

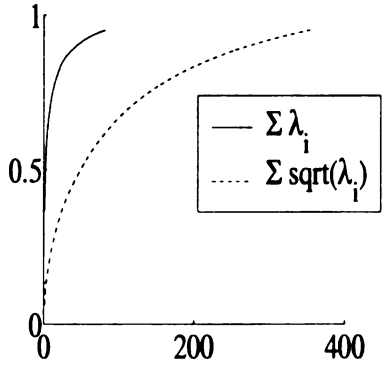
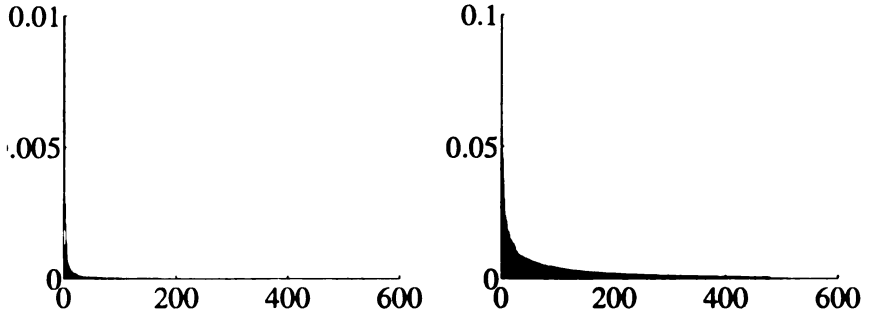


Figure 7.5: Number of Eigen values to be considered for 95% variation.



(a) Eigen values (λ) (b) Root of Eigen values ($\sqrt{\lambda}$)

Figure 7.6: Eigen variation for 512 dimension natural video blocks.

To process one sample in the video block, our algorithm takes approximately 37 milli-seconds to process in 353 dimensional space on 700MHz processor with 256MB RAM running Matlab. This performance is much faster than other methods.

Now that the power of the method is evident in the spatio-temporal domain, we can try to built a sensory mapping that derives features for receptive fields of all sizes and at all positions on the retina (the raw image). As a first step, we shall attempt to

implement a self-organizing spatial feature map which can yield representations for receptive fields at various sizes and various positions.

7.3 Self organizing feature map

We used the basic architecture of a staggered hierarchical map (SHM) suggested in [51] for realizing a *sensory mapping*. In SHM instead of using PCA we used the technique adopted in LCA to address lateral inhibition. Every neuron checks for the winner in the lateral neighborhood and the winning neuron is updated. The set of images that are input to the feature map are first normalized to range from 0 to 1 and then made zero mean. The reader is referred to [51] for details on the architecture of SHM. Neurons are arranged in a staggered fashion to cover different receptive fields. Each neuron has afferent connections to the lower layers. There are many neural layers. Layer 0 is the retina. Each neuron has a different receptive field so as to cover the entire retina. The neural layers are retinotopic. Updating the weights of a neuron is done using the LCA algorithm discussed in Section 6.2.3 with $d = 1$. Lateral inhibition is realized by updating the weights of the winner for every neighborhood. In this way, each neuron can be a winner many times as it is part of many neighborhoods. Each neurons updates itself as many times as it wins. The results for the first layer are shown in Figure 7.7. The parameters that are used for the network are given in Table 7.1. The receptive fields show ON and OFF cells as seen in the LGN. They also show other PCA filters with one side dark and other side bright detecting contrast.

Parameter name	Parameter value
Neural layer 1's size	20×20
Receptive field (RF) size	10×10
Lateral inhibition size	10×10
Inter-neuron distance	2
Increment % of retinal RF size per neural layer	20%
Number of Layers	5
Average number of hits	135000

Table 7.1: Sensory mapping parameters using 1 neuron per RF of the retina

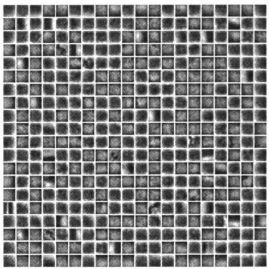


Figure 7.7: Filters for the first neural layer in sensory mapping architecture.

Parameter name	Parameter value
Neural layer 1's size	80×80
Spatial receptive field (RF) size	10×10
Lateral inhibition size	40×40
Inter-neuron distance	2
Number of neurons per receptive field of the retina	16
Average number of hits	8438

Table 7.2: Sensory mapping parameters using 16 neurons per RF of the retina

To make the feature map derive more complete set of features, we tried to increase the scope for competition between the neurons. In the previous experiment, the number of features that are to be extracted per neuron, given by the parameter d in the LCA algorithm discussed in Section 6.2.3, is set to 1. But now, we will have $d = 16$. This is equivalent to having 16 neurons that share exactly the same receptive field. So, the size of the neural layer becomes 80×80 as each receptive field is shared by 4×4 neurons, and there are 20×20 pixels on the retina on which the neurons are centered . So, 16 neurons are centered on the same pixel on the retina. We show the center 40×40 neurons' filters in Figure 7.8 and the parameter values of the architecture are shown in Table 7.2.

For Figure 7.8, the parameters used in the experiment are given in Table 7.2. The center 40x40 neurons are shown.

7.3.1 Spatio-temporal SHM

Spatio-temporal SHM extracts features in both space and time domain. The reader is referred to [49] for the spatio-temporal SHM architecture. We carried out an

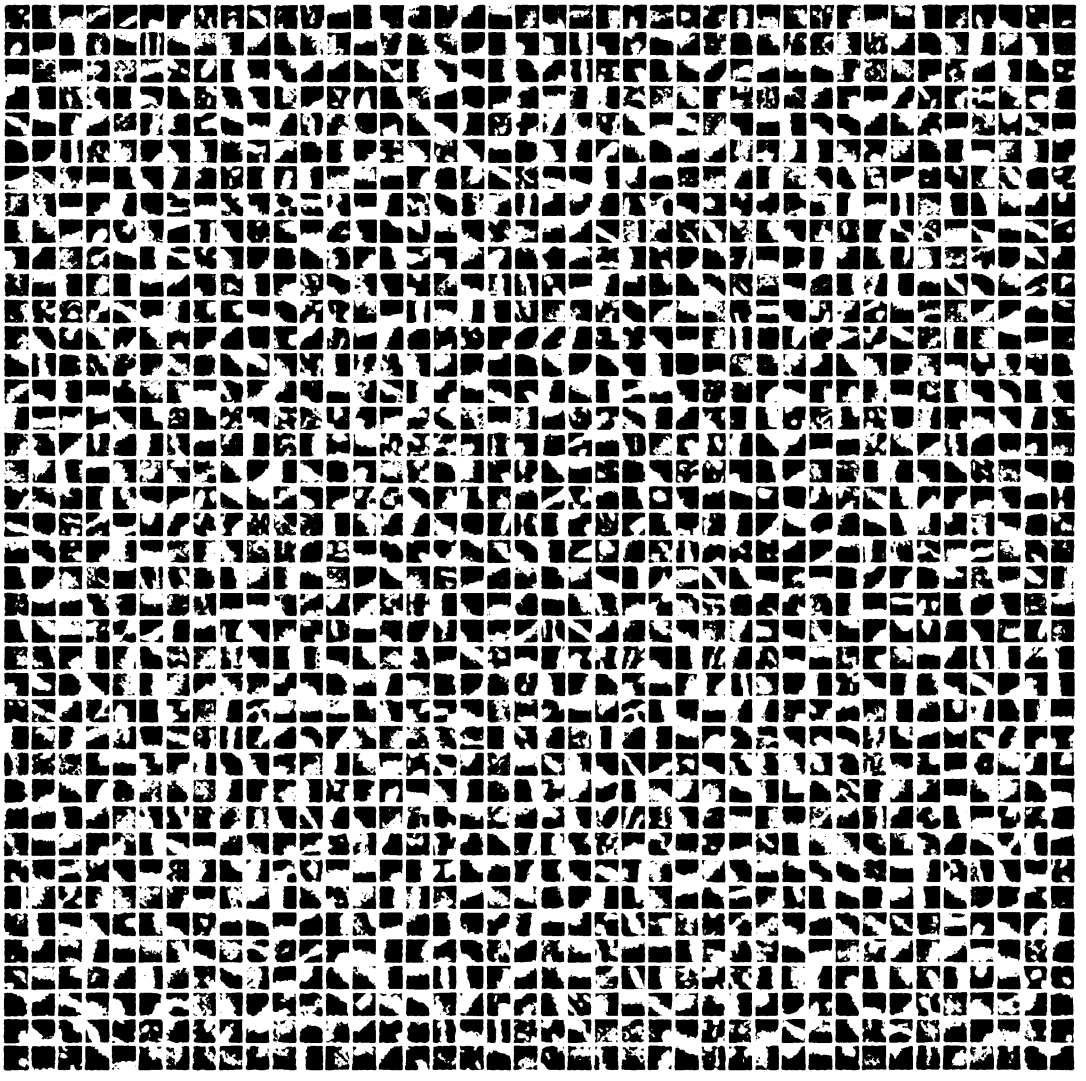


Figure 7.8: Sensory mapping filters for 16 neurons per RF of the retina.

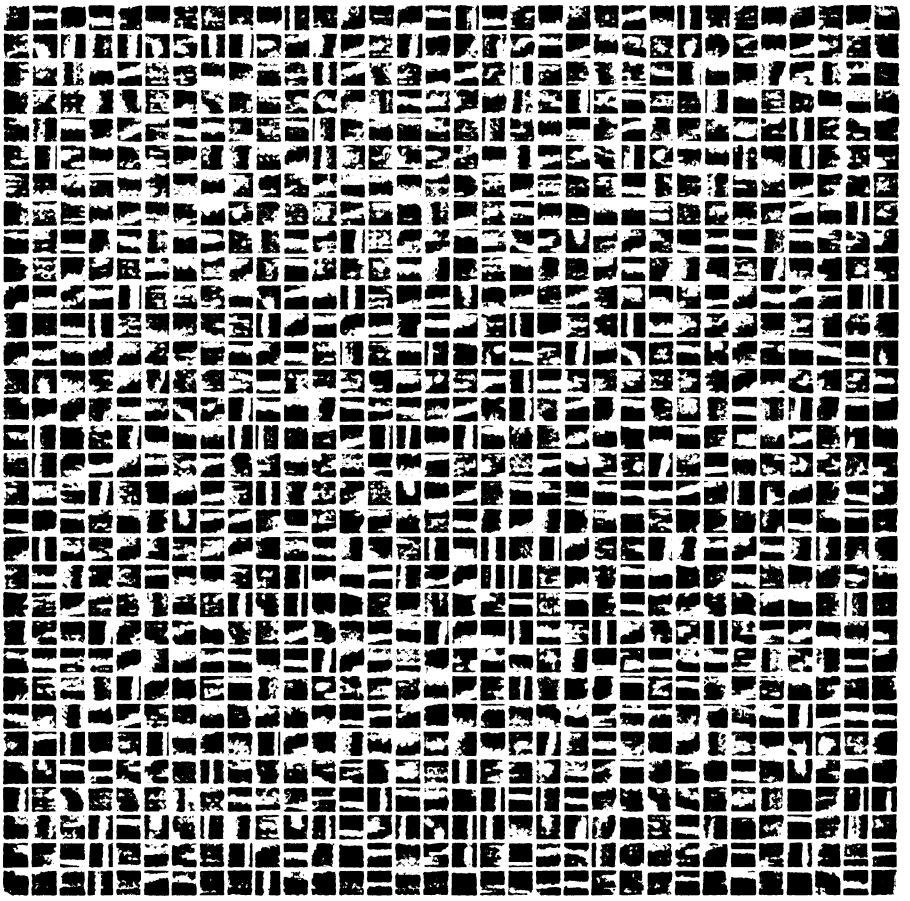


Figure 7.9: Motion Filters for spatio-temporal SHM

experiment with each neuron having a temporal receptive field of 2, which means that two frames from the retina at time t and $t-1$ are given as input to the neuron's receptive field. So, the filters for each neuron can be organized to belong to two frames. The difference in the filters is given as the motion filters in Figure 7.9. Filters for frame 1 are given in Figure 7.10 and filters for frame 2 are shown in Figure 7.11. The parameters for the SHM architecture that were used to derive the filters are given in Table 7.3. As can be seen from the motion filters, most of the filters indicate motion along vertical and horizontal directions. The input sequence fed to the architecture was from SAIL robot's outdoor navigation video.

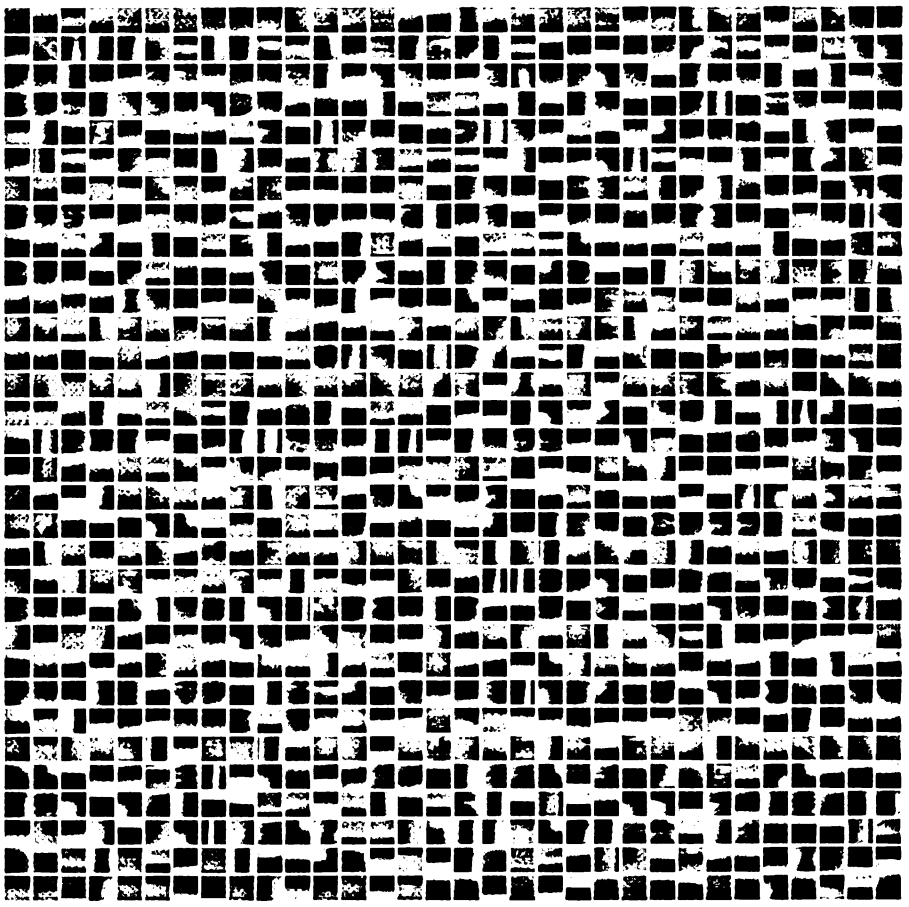


Figure 7.10: Filters for frame 1 of spatio-temporal SHM

Parameter name	**Parameter value**
Neural layer 1's size	16 × 16
Spatial receptive field (RF) size	10 × 10
Temporal receptive field size	2
Lateral inhibition size	10 × 10
Inter-neuron distance	1
Number of neurons per receptive field	4
Number of samples	2000000
Average number of hits	500001

Table 7.3: Spatio-temporal SHM parameters for 16x16 neurons.

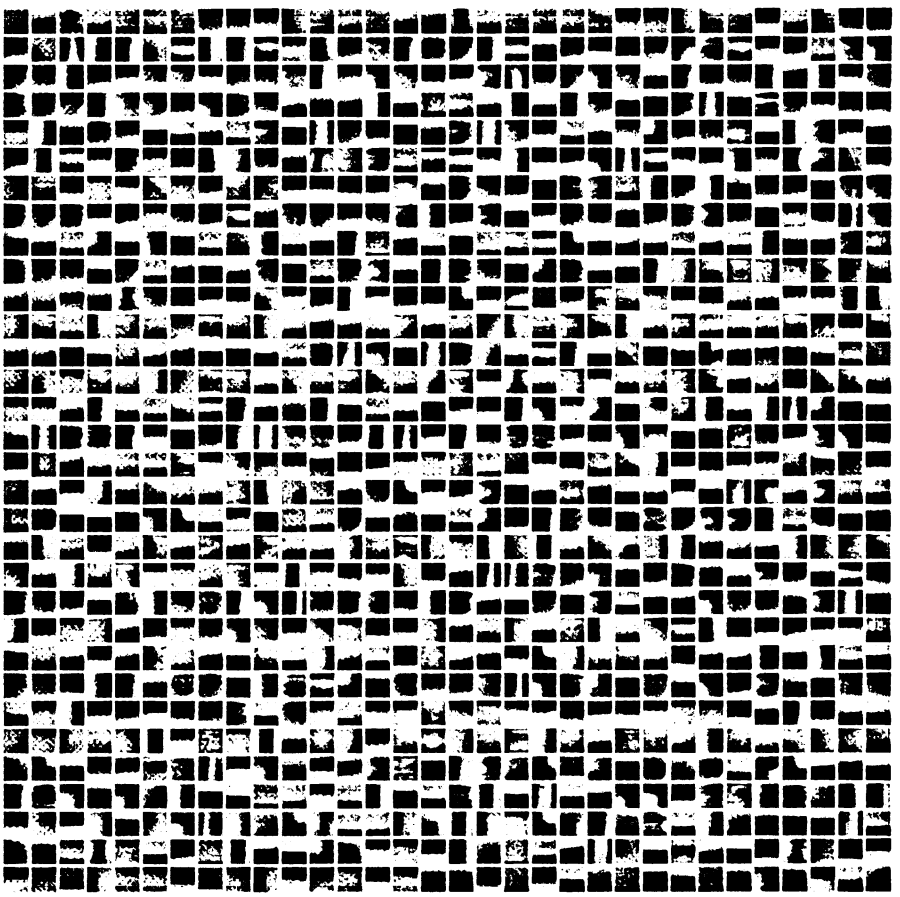


Figure 7.11: Filters for frame 2 of spatio-temporal SHM

Chapter 8

Conclusion

This chapter summarizes all of the work done in the thesis with a discussion of contributions and comments on future work. Section 8.1 discusses the summary and contributions. Future directions for the research are suggested in Section 8.2.

8.1 Summary and Contributions

The main motivation of the thesis was to address the problem of general object recognition. Many model based approaches for general object recognition have been proposed, but are specific to a class of objects and cannot learn new objects for which the system is not designed. Therefore, the autonomous mental developmental framework was adopted to perform general object recognition. We addressed the problem of general object recognition using the sparse and dense label problem experiments. The robot was able to recognize general objects, on which it was trained, with an inaccuracy of 1.33%. But in uncontrolled settings we would need to know where the

object is located in order to recognize it. Also, as the developmental framework uses appearance based methods, we needed to address segmentation or attention selection first.

As a first step in attention selection, we had to extract spatial and spatio-temporal features from images. These features extracted from the images are then used by the higher layers to train the robot to pay attention to moving areas in the image or to those areas which have high novelty without reprogramming the robot. As a result, we focused our attention on feature extraction in sensory mapping. We analyzed various feature extraction techniques and found ICA to be a suitable candidate to extract features that have sparse coding properties. So, we derived fast incremental ICA algorithms in high dimensions.

We introduced a new domain of feature analysis which is biologically motivated by the property of sparse coding. This domain is called lobe component analysis (LCA). An algorithm for LCA was proposed based on the collective best match (CBM) criterion. The expressive power of the LCA algorithm was compared against that of PCA and ICA. Experiments studying the properties of LCA showed desirable features of the algorithm such as expressive ability, dimensionality reduction, clustering, extracting super-Gaussian ICA features, speed of convergence, etc. LCA was used to extract spatial features from natural images.

The use of ICA to extract moving objects from a moving background was demonstrated. Motion filters were extracted from a natural video stream to see the properties of spatio-temporal data. An architecture for a self-organizing staggered hierarchical map based on the techniques used in LCA was proposed and the resulting filters

Topic	Contributions
Incremental ICA	Amnesic gradient ascent of square of kurtosis; incremental fixed point on kurtosis; incremental fixed point on negentropy; ccNegIICA.
Feature analysis	Lobe component analysis (LCA), collective best match (CBM) criterion, LCA algorithm.
LCA	Effect of dimensionality reduction on expressiveness of PCA and LCA; clustering ability of LCA versus PCA; LCA faces versus eigen faces; applicability of LCA in whitened and non-whitened spaces; extraction of super-Gaussian ICA; overcomplete basis estimation.
Filters	Extracting LCA filters from natural images and extracting motion filters from natural video.
Motion segmentation	Extracting a moving object from moving background using ICA.
Sensory mapping	Extracting filters for staggered receptive fields based on LCA techniques. Extracting motion filters for spatio-temporal SHM.

Table 8.1: Contributions of the thesis.

that were learnt are shown. Spatio-temporal SHM was implemented and motion filters for the staggered receptive fields were also shown.

Table 8.1 summarizes the contributions made by this thesis[1].

8.2 Future work

Lobe component analysis finds a great deal applications in areas where you need to know the direction of clusters. Future work can be done in the hierarchical discriminant regression (HDR) tree classifier. Instead of using PCA to extract the most discriminating features in the discriminant sub-space, we can use LCA to extract more discriminating features. This is because, LCA yields the direction of the clusters and

[1]Contributions to LCA algorithm were collectively made by Dr.Juyang Weng, Nan Zhang and the author

has better clustering ability than PCA as demonstrated in the thesis.

A spatial and spatio-temporal self organizing feature map developed can be used to perform attention selection using the higher cognitive layer. The robot can be trained to pay attention to regions of the image based on novelty. After the robot learns to pay attention to moving objects, general object recognition can be attempted in unconstrained environments.

Bibliography

[1] H. B. Barlow. Possible principles underlying the transformations of sensory messages. In W. A. Rosenblith, editor, *Sensory Communication*, pages 217–234. MIT Press, 1961.

[2] A.J. Bell and T.J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7:1129–1159, 1995.

[3] A.J. Bell and T.J. Sejnowski. The 'independent components' of natural scenes are edge filters. *Vision Research*, 37:3327–3338, 1997.

[4] C. Blakemore and G. F. Cooper. Development of the brain depends on the visual environment. *Nature*, 228:477–478, Oct. 1970.

[5] G. G. Blasdel. Orientation selectivity, preference, and continuity in monkey striate cortex. *Journal of Neuroscience*, 12(8):3139–3161, 1992.

[6] P. Comon. Independent component analysis, A new concept? *Signal Processing*, 36:287-314, 1994.

[7] N. Delfosse and P. Loubaton. Adaptive blind separation of independent sources: a deflation approach. *Signal Processing*, 45:59–83, 1995.

[8] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification.* John Wiley and Sons, Inc., second edition, 2000.

[9] J. Weng et al. Autonomous mental development by robots and animals. *Science*, 291:599–600, January 26, 2001.

[10] D.J. Field. What is the goal of sensory coding? *Neural Computation*, 6:559–601, 1994.

[11] M. Fisz. *Probability theory and mathematical statistics.* John Wiley & Sons, Inc., New York, third edition, 1963.

[12] P. Földiak and M. P. Young. *Sparse coding in the primate cortex.* The MIT Press, 1995.

[13] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, 2 edition, 1990.

[14] X. Giannakopoulos, J. Karhunen, and E. Oja. Experimental comparison of neural ICA algorithms. In *Proc. Int. Conf. on Artificial Neural Networks (ICANN'98)*, pages 651–656, Skvde, Sweden, 1998.

[15] X. Huang and J. Weng. Novelty and reinforcement learning in the value system of developmental robots. In *Proc. Second International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems (EPIROB'02)*, pages 47–55, Edinburgh, Scotland, August 10 - 11 2002.

[16] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, 73:218–226, 1962.

[17] W. S. Hwang and J. Weng. Hierarchical Discriminant Regression. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(11):1277–1293, 11 2000.

[18] A. Hyvärinen. A family of fixed-point algorithms for independent component analysis. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP'97)*, pages 3917–3920, Munich, Germany, 1997.

[19] A. Hyvärinen. New approximations of differential entropy for independent component analysis and projection pursuit. In *Advances in Neural Information Processing Systems*, volume 10, pages 273–279. MIT Press, 1998.

[20] A. Hyvärinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Trans. on Neural Networks*, 10(3):626–634, 1999.

[21] A. Hyvärinen. The fixed-point algorithm and maximum likelihood estimation for independent component analysis. *Neural Processing Letters*, 10(1):1–5, 1999.

[22] A. Hyvärinen and E. Oja. A fast fixed-point algorithm for independent component analysis. *Neural Computation*, 9(7):1483–1492, 1997.

[23] A. Hyvärinen and E. Oja. One-unit learning rules for independent component analysis. In *Advances in Neural Information Processing Systems*, volume 9, pages 480–486. MIT Press, 1997.

[24] A. Hyvarinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4-5):411–430, 2000.

[25] J. Karhunen, E. Oja, L. Wang, R. Vigário, and J. Joutsensalo. A class of neural networks for independent component analysis. *IEEE Trans. on Neural Networks*, 8(3):486–504, 1997.

[26] J. Karhunen, P. Pajunen, and E. Oja. The nonlinear PCA criterion in blind source separation: Relations with other approaches. *Neurocomputing*, 22:5–20, 1998.

[27] M. Kirby and L. Sirovich. Application of the karhunen-loève procedure for the characterization of human faces. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(1):103–108, Jan. 1990.

[28] Richard L.Burden and J.Douglas Faires. *Numerical Analysis.* PWS-KENT Publishing Company, Boston, fourth edition, 1989.

[29] T.-W. Lee, M. Girolami, and T. J. Sejnowski. Independent component analysis using an extended infomax algorithm for mixed sub-gaussian and super-gaussian sources. *Neural Computation*, 11(2):417–441, 1999.

[30] H. Murakami and V. Kumar. Efficient calculation of primary images from a set of images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 4(5):511–515, September 1982.

[31] E. Oja. The nonlinear PCA learning rule in independent component analysis. *Neurocomputing*, 17(1):25–46, 1997.

[32] E. Oja. Nonlinear PCA criterion and maximum likelihood in independent component analysis. In *Proc. Int. Workshop on Independent Component Analysis and Signal Separation (ICA'99)*, pages 143–148, Aussois, France, 1999.

[33] B. A. Olshausen and D. J. Field. Natural image statistics and efficient coding. *Network*, 7(2):333–340, 1996.

[34] P. Pajunen and J. Karhunen. Least-squares methods for blind source separation based on nonlinear PCA. *Int. J. of Neural Systems*, 8(5-6):601–612, 1998.

[35] P. J. Phillips, H. Moon, P. Rauss, and S. A. Rizvi. The FERET evaluation methodology for face-recognition algorithms. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 137–143, Puerto Rico, June 1997.

[36] Janne Sinkkonen. http://www.cis.hut.fi/projects/ica/data/images/.

[37] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *J. Opt. Soc. Am*, 4(3):519–524, March 1987.

[38] M. Sur, A. Angelucci, and J. Sharma. Rewiring cortex: The role of patterned activity in development and plasticity of neocortical circuits. *Journal of Neurobiology*, 41:33–43, Oct. 1999.

[39] M. Turk and A. Pentland. Face recognition using eigenfaces. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 586–591, 1991.

[40] Khurram Waheed and Fathi M. Salem. Algebraic overcomplete independent component analysis. In *4th International Symposium on Independent Component Analysis and Blind Signal Separation (ICA2003)*, Nara, Japan, April 2003.

[41] Satosi Watanabe. Karhunen-loéve expansion and factor analysis theoretical remarks and applications. *Transactions of the 4th Prague Conference on Information Theory, Statistical Decision Functions, and Random Processes, Prague*, pages 635–660, 1965.

[42] C. Watkins. Learning from delayed rewards. Technical report, PhD thesis, King's College, Cambridge, England, 1989.

[43] J. Weng. A theory for mentally developing robots. In *Proc. IEEE 2nd International Conference on Development and Learning (ICDL 2002)*, pages 131–140, MIT, Cambridge, MA, June 12-15 2002.

[44] J. Weng and W. S. Hwang. An incremental learning algorithm with automatically derived discriminating features. In *Proc. Asian Conference on Computer Vision*, pages 426 – 431, Taipei, Taiwan, Jan. 8-9 2000.

[45] J. Weng, W. S. Hwang, Y. Zhang, and C. Evans. Developmental robots: Theory, method and experimental results. In *Proc. 2nd International Conference on Humanoid Robots*, pages 57–64, Tokyo, Japan, Oct. 8-9 1999. IEEE Press.

[46] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen. Autonomous mental development by robots and animals. *Science*, 291:599–600, Jan. 26 2001.

[47] J. Weng and Y. Zhang. Developmental robots: A new paradigm. In *Proc. Second International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*, Edinburgh, Scotland, August 10 - 11 2002.

[48] J. Weng, Y. Zhang, and W.S. Hwang. Candid covariance-free incremental principal component analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(8), 2003.

[49] Juyang Weng. *Autonomous Mental Development: Mentally Developing Robots.* Working with MIT Press, New Worky, under preparation edition, Expected to appear within a year.

[50] Juyang Weng and Yilu Zhang. A mentally developing robot and its thinking. Technical Report MSU-CSE-02-26, Computer Science and Engineering, Michigan State University, East Lansing, Michigan, October 2002.

[51] N. Zhang and J. Weng. A developing sensory mapping for robots. In *Proc. IEEE 2nd International Conference on Development and Learning (ICDL 2002)*, MIT, Cambridge, MA, June 12-15 2002.

[52] Y. Zhang and J. Weng. Action chaining by a developmental robot with a value system. In *Proc. IEEE 2nd International Conference on Development and Learning (ICDL 2002)*, pages 53–60, MIT, Cambridge, MA, June 12-15 2002.

[53] Yilu Zhang. *Online development of cognitive behaviors by a robot: a case study using auditory and visual sensing.* PhD thesis, Department of Computer Science and Engineering, Michigan State University, 2002.