Universidad Autónoma de Madrid

Escuela Politécnica Superior

Analysis and Extension of Hierarchical Temporal Memory for Multivariable Time Series

Ph.D Thesis

Author:
David Rozado Fernández

Supervisors:
Dr. Francisco B. Rodríguez Ortiz
Dr. Pablo Varona Martinez

July 29th, 2011

"If you look at the history of big obstacles in understanding our world, there's usually an intuitive assumption underlying them that's wrong."

Jeff Hawkins
American Inventor (1957 -)

Declaration

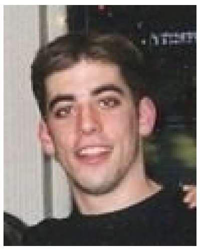
I hereby declare that this thesis is the result of my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for other degree or diploma at Universidad Autónoma de Madrid (UAM), or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UAM or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.

David Rozado

Acknowledgements

I would like to thank all the persons that have helped me, in one way or another, during the development of this work. Particularly, I would like to warmly thank my thesis advisers Francisco de Borja Rodriguez and Pablo Varona and my supervisors during my research visits at the ITU University of Copenhagen and Oxford University, Javier San Agustin, John Paulin and Steve Woolgar. Also, I would like to thank fellow graduate students at labs 208 and 204 of Escuela Politécnica Superior at Universidad Autónoma de Madrid with whom I have collaborated during my Ph.D years and that generously have helped me through their insights, comments, technical know-how or just as “voluntary” research subjects. They are Uwe Pfaff, Fernando Herrero, Richard Rojas, Jesus Diaz, Fabiano Baroni, Pablo Chamorro, and Jacobo Fernández. I would also like to thank the people who have make this work possible by making their codes available, by creating theoretical models or by placing the data sets used in their work on-line; they are Jeff Hawkins and Dileep George for their superb theoretical and technical development of the HTM paradigm, Javier San Agustin and Martin Tall for their outstanding ITU Gaze Tracker and Waleed Kadous for its work on multivariable time series recognition and for making his data sets on sign language widely available. Finally, I want to thank my parents for supporting me during the ups and downs of a Ph.D.

Author's Short Bio and Contact Info



David Rozado Fernández received his Bc. (with honors) in Information Systems from Boston University, US, in 2002. He received his M.Sc. in Bioinformatics from the Freie Universität Berlin, Germany, in 2005. He is currently a Ph.D. student at the Universidad Autónoma de Madrid. His present research topic is on biologically inspired learning algorithms, in particular the usage of Hierarchical Temporal Memory to analyze patterns that unfold over time.

In case the reader of this thesis is interested in contacting the author, here are the coordinates to do so:

David Rozado
Department of Computer Science
Universidad Autónoma de Madrid
28049 Madrid, Spain
☎ +34 628 08 01 13
🖷 +34 91 497 22 35
✉ drozado@gmail.com
ℹ http://www.davidrozado.com

Abstract

Pattern recognition research has not traditionally focused a lot of attention on how the brain solves the very same problem of recognizing patterns itself. Important components of the physical world, such as its hierarchical structure and the existence of the time dimension have often been ignored in machine learning paradigms. Yet, evidence from neuroscience suggests that the brain uses those properties in solving the pattern recognition problem. Classical classifiers usually operate on a static set of attributes and do not mimic the principles used by the brain to recreate invariance during object recognition. This is all in spite of the fact that the brain is a very robust pattern recognition "engine" for many types of problems, one not surpassed yet, by computational approaches.

Hierarchical Temporal Memory or HTM is a neocortically bioinspired algorithm for pattern recognition, time series prediction and control consisting on a network of spatio-temporal pooling nodes arranged in a hierarchy. HTM is a very successful classifier for cases where the whole spatial representation of an instance is present at one point in time, such as for instance image recognition. However, HTMs struggle with problems where instances are composed of a spatial structure evolving over time, i.e. multivariable time series.

In this work, we propose an extension of the HTM algorithm to optimize its performance on the recognition of multivariable time series. Our extension consists of an additional top node in the HTM topology that stores and compares sequences of input data. The spatio-temporal codification of an instance in a sequence serves the purpose of handling its temporal evolution. Sequence comparison is carried out by sequence alignment using dynamic programming. We apply our extended HTM in the representative real world problems of sign language recognition, using data captured with an electronic data glove, and gaze gestures recognition in real time using a low cost video based eye tracking system. Gaze gestures represent an innovative method of human computer interaction consisting on using sequences of gaze movements to generate input commands for electronic devices. Both sign language and gaze gestures recognition are good example of machine learning problems where attributes evolve over time. The position of the hands, or the eyes, at one point in time does not represent the whole instance, and several categories can share certain spatial arrangement of the hands or the eyes. It is the temporal variation over time of the hand position and shape or the eye position what constitutes a sign or a gesture.

On the sign language recognition problem, our extended HTM algorithm reaches the same recognition performance than state of the art techniques employed in sign language recognition such as Hidden Markov Models and Metafeatures Tclasses: 91% recognition performance on a data set of 95 categories of Australian sign language. Furthermore, we surpass state of the art methods on specific aspects such as the need for a low number of training instances, tolerance to increasing vocabulary sizes, low degree of supervision requirements during training, lack of manually curated features models to detect and absence of language or grammar models to support recognition.

Gaze gestures represent another example where instances consist of evolving spatial patterns. We evaluate through a set of user studies, different modalities of carrying out a gaze gesture and determine the superiority of saccadic gaze gestures over gliding gaze gestures. We show how a simple dynamic programming approach outperforms traditional HTMs in the real time recognition of gaze gestures, reaching 94% recognition rate over a simple set of 10 gaze gestures using a self-built, head-mounted and low cost eye tracking system. Furthermore, our extended HTM reaches a 98% recognition rate over the same set of gaze gestures in real time on a remote setup and with higher noise levels. Again, the extended HTM does not use external models, but rather the model is constructed internally by the network through its exposition to the training data. From the robust performance of the algorithm and user reports, we conclude that gaze gestures hold considerable potential as an innovative paradigm of Human Computer Interaction.

In summary, this thesis discusses and extends the features of a neocortically inspired temporal learner capable of producing comprehensible classification of multivariate time series. The problems of sign language and gaze gesture recognition are used as proof-of-concept of the robust performance of the extended HTM method. Nonetheless, the extended HTM algorithm is inherently flexible and has a broad degree of applicability to other recognition problems whose instances are also formed by a spatio-temporal structure unfolding over time.

Resumen

El campo de investigación del reconocimiento de patrones no ha prestado tradicionalmente mucha atención a cómo el cerebro resuelve susodicho problema. Componentes importantes del mundo físico tales como su estructura jerárquica o la existencia de la dimensión temporal, a menudo han sido ignorados en los paradigmas de aprendizaje automático. A pesar de todo, la evidencia que emerge de la neurociencia sugiere que el cerebro utiliza esas propiedades para resolver el problema del reconocimiento de patrones. Los clasificadores clásicos operan a menudo en una serie de atributos estáticos y no mimetizan los principios utilizados por el cerebro para recrear invarianza durante el reconocimiento de objetos. Todo esto ocurre a pesar del hecho de que el cerebro es un motor de reconocimiento de patrones muy robusto para numerosos problemas. Un motor, por cierto, todavía no superado por metodologías computacionales.

La memoria jerárquica temporal, o HTM por sus siglas en ingles, es un algoritmo bioinspirado en la neocorteza capaz de llevar a cabo reconocimiento de patrones, predicción de series temporales y tareas de control usando una red de nodos que realizan un agrupamiento espacio-temporal y que se organizan de forma jerárquica. La HTM es un clasificador robusto para casos donde la representación espacial completa de un patrón está presente en cualquier instante de tiempo, como por ejemplo el reconocimiento de imágenes. Sin embargo, la HTM, no es tan robustas a la hora de trabajar con problemas donde las instancias están compuestas por estructuras espaciales que se desarrollan en el tiempo, por ejemplo las series temporales multivariables.

En este trabajo, proponemos una extensión del algoritmo HTM para optimizar su rendimiento en el reconocimiento de series temporales. Nuestra extensión consiste en la incorporación de un nodo superior adicional en la topología HTM que almacena y compara secuencias de datos entrantes. La codificación espacio-temporal de una instancia sirve el propósito de lidiar con su evolución temporal. La comparación de secuencias se lleva a cabo utilizando el alineamiento de secuencias mediante programación dinámica. Aplicaremos nuestro procedimiento en los problemas representativos de reconocimiento de lenguaje de signos, usando datos capturados con un guante electrónico y de reconocimiento de gestos pupilares en tiempo real usando un sistema de seguimiento de la pupila de bajo coste. Los gestos pupilares representan un método innovador de interacción hombre-máquina consistente en el uso de secuencias de movimientos pupilares para generar comandos de control para aparatos electrónicos. Tanto el reconocimiento del lenguaje de signos como el de los gestos pupilares son buenos ejemplos de problemas de aprendizaje automático donde los atributos evolucionan en el tiempo. La posición de la mano o de los ojos en un instante de tiempo no representa la instancia en su conjunto y varias categorías distintas pueden compartir una cierta configuración espacial de las manos o de los ojos. Es la evolución temporal de la posición de la mano y su forma o de la posición de la pupila lo que constituye un signo o un gesto.

En el problema de reconocimiento de signos, nuestra extensión del algoritmo HTM alcanza unos rendimientos de reconocimiento similares a las técnicas representativas del estado del arte en el reconocimiento del lenguaje de signos, tales como los Hidden Markov Models y las Metafeatures Tclasses: 91% de reconocimiento en un conjunto de datos de 95 categorías del lenguaje australiano de signos. Además, nuestra extensión HTM supera el estado del arte en aspectos específicos tales como la necesidad de un bajo número de instancias de entrenamiento, la tolerancia a tamaños de vocabularios crecientes, bajos requerimientos de supervisión durante el entrenamiento, ausencia de modelos introducidos manualmente en el algoritmo de reconocimiento o la ausencia de modelos gramaticales o de lenguaje para ayudar al reconocimiento.

Los gestos pupilares representan otro ejemplo donde las instancias consisten en patrones espaciales que evolucionan en el tiempo. En este trabajo evaluamos a través de una serie de estudios de usuario, diferentes modalidades de llevar a cabo un gesto pupilar y determinaremos la superioridad de los gestos pupilares sacádicos sobre los gestos pupilares arrastrados. También mostraremos como un simple algoritmo de programación dinámica supera a las HTM en el reconocimiento en tiempo real de gestos pupilares, alcanzando un 94% de reconocimiento sobre un conjunto de 10 gestos pupilares con un sistema de seguimiento pupilar de bajo coste y situado a 5-10cm del ojo. Además nuestra versión extendida de las HTM alcanza un reconocimiento en tiempo real de hasta un 98% sobre el mismo grupo de gestos pupilares utilizando un sistema de seguimiento pupilar remoto y con altos niveles de ruido. Es importante recalcar que nuestro sistema extendido de HTM no usa modelos externos, sino que construye el modelo en la representación interna de la red a través de su exposición a los datos de entrenamiento. Teniendo en cuenta la robustez de los resultados de reconocimiento obtenidos por nuestro algoritmo extendido y la retroalimentación positiva de los usuarios en terminos de usabilidad, concluimos que los gestos pupilares poseen un potencial considerable como forma innovadora y viable de interacción hombre-máquina.

En resumen, esta tesis discute y extiende las características de un clasificador temporal inspirado en la neocorteza capaz de llevar a cabo una clasificación de series temporales multivariables. La aplicación al problema del reconocimiento del lenguaje de signos y al problema del reconocimiento de gestos pupilares se utiliza como prueba de concepto de la robustez de la HTM extendida. En cualquier caso la HTM es un paradigma inherentemente flexible y tiene un amplio espectro de aplicación a otros problemas de reconocimiento cuyas instancias también estén formadas por una estructura espacio-temporal que evoluciona en el tiempo.

Contents

I Introduction **11**

1 Introductory Remarks **12**

2 The Need for Bioinspired Approaches in Artificial Intelligence **15**
- 2.1 Introduction 15
- 2.2 A Short History of AI 16
- 2.3 Symbolic vs. Connectionist Approaches 18
  - 2.3.1 Symbolic AI 19
  - 2.3.2 Connectionist Approaches - Artificial Neural Networks 19
- 2.4 Some Thoughts on the Need for Bioinspired Approaches in AI 21

3 Mammalian Neocortex and Memory **24**
- 3.1 The Neocortex 24
- 3.2 Memory and Functions of the Neocortex 30

4 Hierarchical Temporal Memory **34**
- 4.1 HTM Overview 34
- 4.2 HTM Theoretical Principles 35
  - 4.2.1 Discovery and Inference of Causes in HTM Theory 38
  - 4.2.2 The Importance of Hierarchy 40
  - 4.2.3 The Importance of Time 41
  - 4.2.4 Discovery and Inference of Causes by each Computational Node 42
- 4.3 Comparison of HTMs with other Machine Learning Algorithms . . . 43
  - 4.3.1 General Purpose Probabilistic Models 44
  - 4.3.2 Non Generative Models 45
  - 4.3.3 Empirical Neuro-Biological Models 45

5 Example Applications Of Hierarchical Temporal Memory **47**
- 5.1 Introduction 47
- 5.2 HTMs in Image Recognition 48
- 5.3 HTMs in Sign Language Recognition 49
- 5.4 HTMs in Gaze Gesture Recognition 51
  - 5.4.1 Eye Tracking 51
  - 5.4.2 The Nature of the Eye 52
  - 5.4.3 States of the Eye 54
  - 5.4.4 Eye Tracking Techniques 55

5.4.5 Gaze Estimation 56
5.4.6 Eye Movement Detection 57
5.4.7 Gaze Interaction 58
5.4.8 Gaze Gestures 60
5.4.9 Eye Gaze Based Interaction - Limitations 62
5.4.10 Stroke Complexity 62
5.4.11 Finite Gaze Gestures 63
5.4.12 Continuous Gaze Gestures 64
5.4.13 Final Thoughts on Gaze Gestures 65

II Results 66

6 HTMs in Image Recognition 67
6.1 Introduction 67
6.2 HTM Algorithm - Methodology 68
6.2.1 Task, Training Data and Network Structure 68
6.2.2 Node Operation 69
Operation of the Spatial Pooler 72
Operation of the Temporal Pooler 73
Supervised Pooling at the Top Level 78
6.2.3 Hierarchical Node Operations 79
6.2.4 HTM Implementation of Coincidence Detection 80
6.2.5 Usage of Probability Distributions in Message Passing 81
6.3 Results 81
6.3.1 Recognition 82
6.4 Conclusions and Directions for Future Research 83

7 Extended HTMs in Sign Language Recognition - Comparative Study 85
7.1 Introduction 85
7.2 Methods 87
7.2.1 Data 88
7.2.2 Markovian HTM 89
7.2.3 Metafeatures TClass 90
7.2.4 Hidden Markov Models for SLR 90
7.2.5 Extended HTM 92
7.2.6 Experimental Design 95
7.3 Results 97
7.4 Discussion 105

8 HTMs in the Recognition of Gaze Gestures Offline 109
8.1 Introduction 109
8.2 Eye Tracking and Gaze Gestures 110
8.3 Experimental Setup 110
8.4 Results 112
8.5 Discussion 116

9 Gaze Gestures Recognition in Real Time 118
9.1 Introduction . . . 118
9.2 Methodology . . . 121
9.2.1 Eye tracking . . . 122
9.2.2 Gaze Gestures . . . 122
9.2.3 Hierarchical Temporal Memory . . . 123
9.2.4 Sequence Alignment and Dynamic Programming . . . 124
9.3 User Study . . . 126
9.3.1 Pilot Experiment . . . 127
Experimental Setup . . . 127
Results of the Pilot Experiment . . . 129
Discussion of Pilot Experiment . . . 131
9.3.2 Main User Study . . . 131
Experimental Setup . . . 131
Results . . . 132
9.3.3 Recognition Scores Overlap of Intended and Non-Intended Gestures . . . 133
9.4 Discussion . . . 134
9.5 Conclusion . . . 140

10 Extended HTMs in the Real Time Recognition of Gaze Gestures 143
10.1 Introduction . . . 143
10.2 Methods . . . 144
10.2.1 Eye tracking . . . 145
10.2.2 Gaze Gestures . . . 145
10.2.3 HTM Formalism . . . 145
10.2.4 Extended HTM . . . 145
10.2.5 Experimental Setup . . . 146
10.3 Results . . . 149
10.4 Discussion . . . 154
10.5 Conclusion . . . 159

III Summary and Conclusions 161

11 Summary of Results 162

12 Conclusions 165

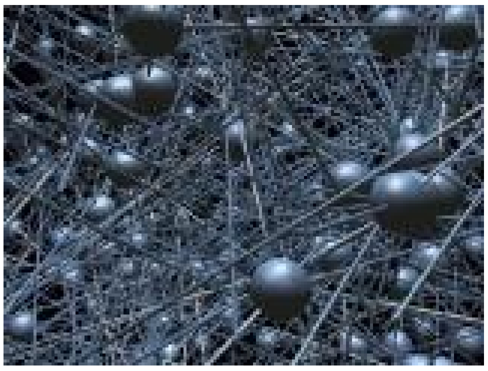
13 Conclusiones 168

Part I

Introduction

Chapter 1

Introductory Remarks

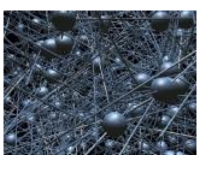


This thesis deals with the subject of pattern recognition within the realms of computational neuroscience and machine learning. Over the course of this work, we lay out a description, analysis and extension of a connectionist and bio-inspired machine learning algorithm known as Hierarchical Temporal Memory or HTM [Hawkins, 2004, 2006; George and Hawkins, 2009].

Numerous tasks that humans find easy to do are very difficult or impossible to replicate on a computer: image recognition, understanding spoken language, manipulation of objects and navigation in 3D space are routinely performed by humans with apparent ease. However, no computer can even come close to human performance in these tasks in spite of numerous research efforts working toward those goals.

In humans, the main cognitive actor carrying out the biggest load of those tasks is the neocortex. Hierarchical Temporal Memory is a connectionist artificial intelligence paradigm that pursues the far fetched goal of encapsulating the structural and algorithm properties of the neocortex into a set of computer algorithms.

Hierarchical Temporal Memory is relatively different to many machine learning approaches trying to emulate human cognitive capabilities. In traditional artificial intelligence paradigms, programmer's create customize programs to solve specific problems. For instance, a program developed to recognize speech or to predict weather. HTM is more like a memory system and it does not execute different



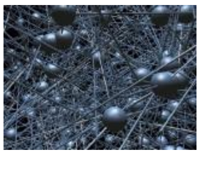
algorithms for different problems. HTM are trained by exposing the network to data encapsulating the characteristics of the problem at hand and hence, the capabilities of the HTM are determined largely by what it has been exposed to.

HTM theory incorporates the hierarchical organization of the neocortex into its topology [Hawkins, 2006], and it uses spatio-temporal codification to encapsulate the structure of problems' spaces. Hierarchical organization and spatio-temporal coding are well documented principles for information processing in neural systems [Rabinovich et al., 2006; Rodríguez and Huerta, 2004; Shepherd, 2004]. HTM algorithms perform robustly in traditional machine learning tasks such as image recognition [George and Hawkins, 2005] where patterns are represented as a fixed set of attributes. For problems where an instance is composed of time series of varying spatial arrangements, HTM performance is not as robust [Numenta, 2006b]. Hence, in this thesis we suggest an extended HTM algorithm to improve performance on this type of problems. To illustrate the validity of our extended HTM system, we measure its performance on datasets consisting of multi-variable time series: a dataset of Australian sign language instances gathered with an electronic data glove and several gaze gesture datasets gathered with video-based eye tracking.

HTMs encapsulate within their hierarchical topology a model of the world they are exposed to during training. Category instances in problem spaces (images, sounds, financial information) possess structure. This structure is often hierarchical in both space and time [Hawkins, 2006]. Since HTM networks are intrinsically hierarchical in the spatio-temporal domain, they can efficiently capture in its structure a model of the world on which they have been trained.

HTMs are similar in some aspects to Bayesian networks. However, they are different in the way they use time and hierarchy. HTM can be implemented in software or in hardware [Hawkins, 2006]. Often, it is useful to think of an HTM as a memory system. Since HTM has been derived from biology, a mapping between HTM and the biological anatomy of the neocortex can be traced [George and Hawkins, 2009].

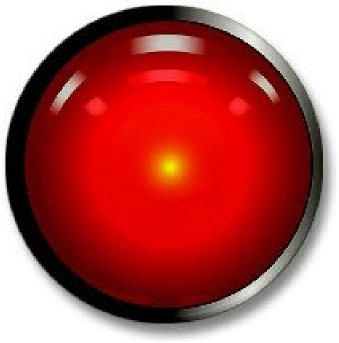
This thesis is structured as follows: an *Introductory* part encompassing the first five chapters is followed by a *Results* part compromising another five chapters. A *Summary of results and Conclusions* part concludes this thesis by providing as its title indicates a wrapping up the main results of this work and the main conclusions that can be derived from them. In chapter 2, a very succinct argument for the need of bioinspired approaches in artificial intelligence is provided. Chapter 3 delineates some of the basic characteristics of the neocortex. Chapter 4 lays down a basic theoretical description of Hierarchical Temporal Memory. Chapter 5 presents the particular pattern recognition problems in which we will work to test traditional and extended HTMs and also provides background information on the topics of eye tracking and gaze gestures interaction. Chapter 6 describes the results of our own HTM implementation applied to an in-house data set of two dimensional binary images. Chapter 7 shows the results of our extended HTM algorithm in the recognition of sign language, and how extended HTM compares with traditional HTM algorithms and traditional machine learning techniques usually employed in the literature for sign language recognition, namely Hidden Markov Models and Metafeatures Tclasses. Chapter 8 provides preliminary results of traditional HTM



employed in the recognition of offline gaze gestures. Chapter 9 consists of an in depth study of different gaze gestures modalities and their corresponding real time recognition with traditional HTMs or using a simple dynamic programming techniques. Chapter 10 describes the application of our extended HTM algorithm to the recognition of gaze gestures in real time. Finally, chapter 11 summarizes the main results described in this thesis and chapter 12 lists the main conclusions to extract from this work.

Chapter 2

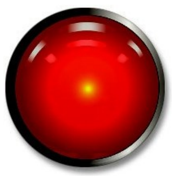
The Need for Bioinspired Approaches in Artificial Intelligence



2.1 Introduction

Artificial intelligence, or AI for short, can be described roughly as the area of computer science aiming to create machines that can produce intelligent behavior [Russell and Norvig, 2002]. Nonetheless, a precise and formal definition of the term AI remains elusive. The imprecise nature of the word intelligence itself has played a role in this difficulty. The concept "intelligence" represents a blurry characteristic that has proved difficult to measure. Some cognitive theorists though, claim that intelligence allows the identification of the right piece of information at the appropriate instant during a decision-making process [Rios, 2007].

AI is a broad subject integrated by experts from different fields: computer science, physics, neuroscience, psychology, physiology and philosophy. Research into the areas of learning, language, and sensory perception have aided scientists in their far fetched goal of building intelligent machines. The enormity of the task can be



grasped by realizing that the final goal of those efforts is to emulate the functionality of the human brain, made up of billions of neurons [Rios, 2007].

The ability to create intelligent machines has attracted humans interest since ancient times. Today, in spite of the ever increasing power of microprocessors and 50 years of research into AI, the dream of truly intelligent machines remains distant. Yet, research continues to be done with the objective of creating machines that can mimic human thought or comprehend emotions. Traditionally, AI research has assumed that machines can match human intelligence without the need to understand how a real brain generates intelligent behavior. But AI performance in comparison to some human capabilities remains unimpressive, this suggests a pressing need to look at neuroscience research in order to incorporate in computational models of AI the principles that the brain uses to process information.

2.2 A Short History of AI

The beginnings of AI research can be traced back to philosophers and mathematicians who built up a theoretical body of knowledge on the principles and foundations of logic. AI consolidated itself as a field of its own a few years after the invention of the electric computer in the 1940s. Computers appeared at first sight as a technology that would easily simulate intelligent behavior.

It is widely assumed that English mathematician Alan Turing laid down the foundations of AI. Turing provided a formal demonstration of the concept of universal computation. This concept states that any computer is equivalent to any other regardless of how it is actually built. As long as the CPU can realize a basic set of operations and it has at its disposal an unending tape (memory), it can carry out any definable sequence of operations. Turing felt that computers could be intelligent and he proposed a way to determine if a machine has gained human intelligence, the Turing test, claiming that a computer could be considered intelligent if it could deceived a person into thinking that it was itself a human. From his writings, it seems like Turing assumed that the brain was just another kind of computer.

The field of AI was formally established in 1956, at the Dartmouth conference. The term AI was coined by the organizer of that conference: John McCarthy. The conference itself did not produce any breakthroughs, but lied the bases for future research. Since then, and although advancements have been slower than first estimated, AI has expanded into a vivid research field.

Pioneers of AI research perceived strong parallelisms between computation and human intelligence. They observed that what is commonly considered as the highest pinnacle of human intelligence: the manipulation of abstract symbols, i.e. language, chess or math, can be formalized in a logical language interpretable by a universal computer. Turing had previously shown that it is irrelevant the hardware with which the symbols are implemented or manipulated. As long as the functional equivalent of the universal Turing's machine could be realized, it could be done with a network of electronic switches or a network of neurons.

In 1943 the neuroscientist Warren McCulloch and the mathematician Walter

Pitts published an influential journal article [McCulloch and Pitts, 1990] providing further support to this notion. In their work, the authors described a theoretical model for real neurons to be connected together to perform logical operations. That is, they showed that it was theoretically possible that neurons could replicate the functionality of logic gates used by digital computers. Logic gates are electric circuits able to perform digitally logical operations such as *AND*, *NOT*, and *NOR*. Thus, McCulloch and Pitts claimed that it was feasible that the brain could theoretically be built and wired in a manner analogous to electronic circuits although, they did not proved that that was how the brain was actually build [Hawkins, 2004].

A dominant school of thought in psychology during the first half of the 20th century was that of behaviorism, which in turn heavily influenced and shaped AI from its inception. Behaviorists thought claimed that the inner-workings of the brain were inaccessible (they even referred to the brain as an impenetrable black box) and hence it was better to focus efforts on just observing and measuring characteristics of the brain. Behaviorists were particularly interested in changing the brain and exploring the technique of conditioning: making an animal adopt new behaviors through reward and punishment schemes. At that time, several scientists working on artificial intelligence also felt that mathematics and logic would be enough for computers to do vision, language, robotics and so on. The prevailing view was that if, as it appeared back then, a computer could emulated the functionality of a brain, research efforts should not constrain themselves with biological boundaries [Hawkins, 2004]. Hence, during that time there was not a lot of interest on the AI community in trying to understand the nuts and bolts of the human brain.

In the 1940s, the emergence of the electrical computer both in the U.S. and Germany revolutionized information processing and storage [Ennals, 2004]. The innovation of the stored programmable computer, late during the 1940s, made the job of entering a program easier. Further advancements in computer theory laid the ground for the emerging field of computer science, and its artificial intelligence subfield. In the early 1950s, the link between human intelligence and machines started to become apparent. Norbert Wiener carried out very influential work on the topic of feedback theory. Early developments of AI were hence heavily influenced by the notion that intelligent behavior was the result of feedback mechanisms.

Digital computers became more pervasive in the second half of the 20th century, pioneers of AI thought that problems such as language translation and vision would be easy to tackle by AI. The idea of using geometric theorems to deal with translations and rotations seemed straight forward. An important breakthrough was the emergence of programs that could infer mathematical theorems (something which it is considered to require high level cognitive functions). However, this programs were very limited in their mathematical skills since they could only derive very simple theorems, which were already proved [Hawkins, 2004] and lacked any type of mathematical "creativity".

The 1970s witnessed the advent of "Expert Systems" which caused a big fuss. An expert system consists of a knowledge database and an inference engine for interpreting incoming data using the knowledge supplied in the knowledge base. The databases are constructed by cropping knowledge from experts in a certain domain, collecting facts about that particular domain and embodying all these knowledge into a computer program for carrying out some task. These systems can only operate on the facts that have been built within them and often are not able to handle contextual differences. This turned out to be of limited use in terms of AI but very important in commercial settings [Hawkins, 2004].

During the 1980s, AI expanded into industry with large companies increasingly relying on "Expert Systems". Some sub-fields failed on their way to the market place. One in particular was machine vision. Although not very robustly, image recognition algorithms could differentiate shapes of objects using black and white differences but not enough computer power was available at the time to create mainstream applications [Hawkins, 2004].

During the late 1980s, demand for AI software reached a plateau. Further advances nonetheless, continued occurring in the research front. Fuzzy logic, that had the unique ability to make decisions under uncertainty created a fuss of its own. Connectionist approaches such as neural networks attracted interest to the up to then somehow disregarded notion of bioinspired AI [Hawkins, 2004].

Over time, computers would manage to match the highest human performance at chess as demonstrated when IBM's Deep Blue won Garry Kasparov in a series of chess matches. But these successes are misleading. The computer was not being smarter than the human, it was just billions of times faster and able to explore billions of possibilities in a huge parameter space.

The subject of artificial intelligence nowadays spans a wide range of subfields and it is not uniform. AI deals with knowledge representation schemes, intelligence search, planning, motion and manipulation, machine learning, natural language processing, social intelligence, creativity, logical AI, deduction, reasoning, problem solving and many others. Such a diverse set of topics has emerged only thanks to the field being enriched with interdisciplinary knowledge from philosophy, psychology, cognitive science, computer science, mathematics and engineering.

2.3 Symbolic vs. Connectionist Approaches

AI research can be divided in two major branches: the symbolic approach or top down and the connectionist approach or bottom up [Sondak and Sondak, 1989]. The symbolic approach seeks to replicate intelligent properties of the brain by studying cognition independently of the biological structure of the nervous system. That is, in terms of the study of symbol processing and manipulation. The connectionist approach towards AI tries to encapsulate some of the biological properties of the brain in computer algorithms with unitary computational components called nodes. Artificial neural networks are the most common example of connectionist approaches [Krogh, 2008].

2.3.1 Symbolic AI

Symbolic AI tries to represent human knowledge in a declarative form (using facts and rules). Symbolic AI tries to translate implicit or procedural knowledge into explicit form by means of symbols and rules for their manipulation. Symbolic approaches have been very successful as expert systems or as chess playing programs. Symbolic AI however possesses certain drawbacks. The main problem of symbolic AI is the common sense knowledge problem.

The common sense knowledge problem refers to the enormous amounts of implicit knowledge that humans possess about the world and ourselves. For instance, implicit knowledge such as “if a person has a father, then the father is always older than the person and will remain older throughout its life” is obvious to humans. However, it is unrealistic to represent in a symbolic form all sort of similar constructs. Trying to replicate intelligence without that kind of knowledge is unlikely to succeed since there exists a surprising amount of implicit knowledge on which we operate on a daily basis [Sondak and Sondak, 1989].

The common sense knowledge problem deals with how to efficiently implement knowledge representation: what is the best approach to represent knowledge? Is it a dictionary or a set of rules? Can everything be formulated as a set of “if . . . then . . . ” rules? What about multiple forms of representation? Should they be used at all? It is apparent that very little amount of human knowledge is represented in the human brain in an explicit form. The implicit nature of knowledge is obvious in tasks such as visual perception, understanding language, creative thinking, etc. It is precisely in those tasks where the brain naturally relies on procedural or implicit knowledge, such as sensory information processing (image recognition, speech recognition, etc) where symbolic AI has struggled. And it is precisely those fields, where symbolic AI has had limited success, where connectionist approached have shown to be more suitable for such tasks.

2.3.2 Connectionist Approaches - Artificial Neural Networks

An artificial neural network (ANN) is a computational model inspired on the structure and/or functional aspects of biological neural networks. An artificial neural network consists of an interconnected group of nodes that process information using connectionist approaches to computation. Most often, artificial neural networks operate as adaptive systems that change the structure of the network according to external or internal information that flows through the network during the learning stage [Specht, 1990].

Neuroscience research has intensively investigated the idea of connection strengths between real neurons. Connection strength refers to how strongly one neuron is able to influence those neurons with whom it makes a synapse. Learning through exposure to a repetitive stimulus causes the brain’s connection to change [Hebb, 1949]: synaptic connections become enhanced or inhibited. Real neurons can operate as excitatory or inhibitory influence in the neurons with whom they connect. The degree of excitation or inhibition of a given neuron is related to the receiving connection strengths. Hence stronger connection, results in more marked inhibition or excitation of the receiving neuron. The neuron transfer function heavily shapes a neuron response to an stimulus. The transfer function defines how the neurons output firing rate should change as a function of the inputs received from other neurons. A very sensitive neuron for instance may increase its firing rate with relatively little input. This neuron is said to have a low threshold for excitation. Other neurons may possess a firing rate function resembling the shape of a bell curve, i.e. its firing rate increases to a maximum and then decreases if over stimulated. Other neurons yet sum up its firing rate or implement other type of aggregation functions. All of these behaviors can be represented mathematical in a transfer function.

An artificial neural network tries to capture the biological aspects discussed in the previous paragraph. The nodes of an artificial neural network are connected to each other. The strength of the connection is indicated by a value that represents the degree of inhibition or excitation that the neuron through its connection is able to convey. Also each node possesses a transfer function that determines its response to incoming input.

Most artificial neural networks are organized as a set of input nodes, hidden nodes and output nodes. Input nodes take information (and emulate the purpose of sensory organs). Information percolates through the network and the impact of any given channel of information flow is determined by the particular connection strengths it goes through. The transfer functions determine how much of the activation value is passed on to the next node. Each node aggregates the input information it receives, computes its own activation value and passes this information to the next node in the network. The information flows from input nodes, through hidden nodes towards output nodes.

Learning in artificial neural networks is an iterative process consisting on adjusting connection strengths according to a scoring function between the input-output relationship and the given category the network is exposed to during training. The most extended method of learning is the back-propagation algorithm. Back-propagation starts of with the network connection strengths' initialized with random values. A training instance is presented, information percolates through the network and the output node values are compared with the correct response. Working backwards from the output node, connection strengths are adjusted such as if a repeated training instance is presented, the network's output will be closer to the desired one. The described process is called a back-propagation cycle. Usually several iterations of this cycle are required for optimal training. After training, the network should become fairly proficient at identifying the categories on which it was trained when exposed to unknown instances of those same categories [Yao, 1993; Lampinen and Vehtari, 2001].

A problem with the back-propagation algorithm is that in real neurons learning does not seem to proceed backwards [Lampinen and Vehtari, 2001]. Very often also, learning in real brains can take place without the presence of a supervisor, which in the case of the back-propagation algorithm is not possible.

The big advantage of artificial neural networks is that they can work with noisy input such as blurry pictures. This is precisely the type of problems were symbolic approaches struggle. Critics of artificial neural networks often point out neural nets' inability to learn something like logic.

The beginning of connectionist approaches to AI can be traced back to the 1940s, when [McCulloch and Pitts, 1990] published their work "A logical Calculus of ideas Immanent in Nervous Activity" where they presented an abstract model of a neuron in which the probability of it activating itself depends on the input signal and the connection strength. Also during the 1940s, [Hebb, 1949] published the work "The organization of the Behavior" where he describes how neurons can learn by reinforcing the connection strength between neurons when they are activated in close temporal proximity. In the 1950s [Rosenblatt, 1958] developed the concept of a perceptron. A perceptron is a type of ANN that performs recognition of abstract and geometric patterns. In the 1960s [Hoff, 1962] develops a model based on the ideas of [McCulloch and Pitts, 1990] and the perceptron from [Rosenblatt, 1958]. The new model is known as ADALINE and it is able to linearly separate input spaces. At the end of the 1960s [Minsky and Papert, 1969] published a book entitled "Perceptrons". On it, authors expose perceptron's biggest weakness, the famous XOR problem (or exclusive No). XOR is a logic operator in which the conclusion is true only if just one of the premises is true and false if both are true or false. The solution to this problem is not linearly separable and a normal perceptron can not learn to separate the class. In the 1980s, [Hopfield and Tank, 1985] developed the concept of a Hopfield net: a recurrent ANN that work as content-addressable associative memory system with binary threshold units. Hopfield nets guarantee to converge to a local minimum, but convergence to one of the stored patterns is not guaranteed. During the late 1980s [Kohonen, 1988] developed the concept of "self-organizing map" or (SOM). SOMs are a type of artificial neural network trained using unsupervised learning to produce low-dimensional, discretized representation of the input space of the training samples, called a map. SOMs differ from typical ANN by using a neighborhood function to preserve the topological properties of the input space. From then on, other ANNs models have been suggested [Haykin, 2008] that further develop and optimize the work from the aforementioned pioneers but without much emphasis on bioinspiration.

2.4 Some Thoughts on the Need for Bioinspired Approaches in AI

Through AI's history, AI programs have only managed to do the task for which they have been specifically designed. In general, AI programs lack the ability to generalize among different contexts or show flexibility in any way remotely similar to the degree that the human brain exhibits. AI researchers' initial optimism about the possibilities of AI in several fields (language comprehension, creativity, motor control, etc) have yielded no significant progress over the years. Making computers emulate human perception, language or behavior remains very challenging. There are still researchers who argue AI can successfully tackle any problem with faster computers, but there is a considerable amount of excepticism about that approach

[Hawkins, 2004].

Some scientists argue that computers should rather focus on simulating real neural networks in a computer. Theoretically, computer models could simulate all the neurons in the brain and their connections. If the simulation were to be realistic, it should be indistinguishable from a real brain and hence the system should show intelligent behavior. However, a brain can not be simulated without first understanding what it does. Furthermore, technologically it seems a long way off until scientists are able to precisely described the complete anatomy and physiology of a human brain, not even to model that system in a computer.

Over the last six decades, AI has grown from its humble beginnings in which only attracted a few researchers to nowadays thousands of engineers and specialists on AI; Algorithms have also progressed from programs capable of very rudimentary dialogs, to complex systems able to perform medical diagnostics. However, the goal of recreating real intelligence on a computer remains elusive. Advancements in the quest for AI though, have and will continue to affect our society [Hawkins, 2004].

The difficulties of AI to live up to its promises suggests that there exists a pressing need for more bioinspired approaches in AI in general and machine learning in particular.

Connectionist approaches that have been particularly successful in pattern recognition problems have not incorporated major significant bioinspired approaches since the 1950s. This is all in spite of the fact that knowledge about brain function has experienced an exponential growth over the last 6 decades. However, very few of this knowledge has been incorporated in artificial intelligence paradigms.

ANNs have often been accused of being too simplistic to be considered accurate models of brain function. While it is true that ANNs try to model neural like attributes such as connectionist strengths, inhibitory and excitatory activity, it is also true that they ignore important attributes of brain function such as temporal dynamics, conduction velocities, local recognition, discrimination of input signals, multicoding strategies and spatial locations which may be significant aspects of real brain functioning [Latorre et al., 2011]. So despite their flamboyant name, it seems highly likely that ANN are just oversimplified models of brain function.

Most ANN consider individual nodes functionally identical, possessing the same transfer function and without a mechanism for transient memory in each node [Latorre et al., 2011]. Furthermore, no ANN paradigms discriminate information as a function of the recognition of the emitter unit. The oversimplification of ANN facilitates the mathematical formalism subjacent to each paradigm, yet it fails to mimic biological neural networks in detail. Numerous ANN paradigms also have disregarded biological inspiration for topological organization and the hierarchical organization of the neocortex has not been fully incorporated in traditional ANN. Some ANN do incorporate the concept of a spatial hierarchy but they do not use a spatio-temporal hierarchy. Traditional ANN have also ignored the role that time could play during learning despite the fact that the time dimension has been proved critical in nervous systems learning [Hebb, 1949; Auyeung et al., 1987; Kandel et al., 2000].

Although for many problems ANN work very well, for several others their performance is not optimal. Given the lack of transmission from neuroscience research about brain information processing to machine learning paradigms, it could be argued that the incorporation of additional properties derived from biology could help ANN reach higher levels of performance. Bioinspired approaches can be applied to important aspects of ANN such as individual neuron dynamics, network topology, and learning functions. In this thesis we will explore the incorporation of biologically realistic topologies in ANN and the usage of time as a sort of supervisor to cluster temporally adjacent spatial patterns. Of course, additional knowledge from human neural information processing will be required to further develop bioinspired approaches in AI.

Chapter 3

Mammalian Neocortex and Memory

3.1 The Neocortex

Brain architecture can be very informative in terms of how the brain works. The outer surface of the brain is formed by a thin sheet of uniform neural tissue that surrounds the rest of the brain and it is called the neocortex, see Figure 3.1. Most high level cognitive functions (language processing, creativity, mathematics, music, executive control, perception) are highly based and dependent on neocortical activity.

The brain consists of many other structures besides the neocortex (brain stem, cerebellum, hippocampus, thalamus, etc). These structures are involved in regulating hunger, pulse, emotions, muscle coordination, sexual urges, etc. But it is the neocortex, the structure that is dominant in those behaviors what we often associate with "intelligent" behavior. Many of the subcortical structures in the brain are highly interconnected with the neocortex with often both parts requiring each other for proper function. This is especially true for the hippocampus [Kandel et al., 2000].

A detailed description of the mammalian brain exceeds the scope of this chapter, so we will briefly delineate the core aspects of its anatomy and functionality.

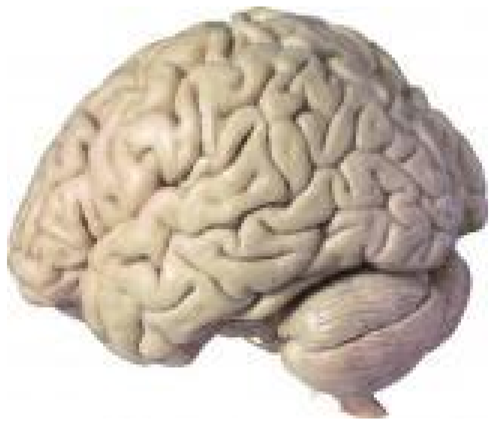


Figure 3.1: **Neocortex tissue.** Picture of a human brain. The outside structure is mainly composed of neocortex, with the other parts of the brain totally covered from our view by the neocortex except the cerebellum (at the lower right of the image). The convoluted structure of the neocortex is meant to maximize its surface area.

The neocortex is divided into six layers, altogether encompassing 2mm in thickness. Each layer has a set of features in terms of cells types on it and connectivity patterns that makes it different from the others. The surface area of the neocortex amounts to about a table napkin in size [Hawkins, 2004]. The neocortical area of other mammals is smaller but all of them also contain six layers. It can be argued that humans are "smarter" by having a larger "computational" area of neocortex folded up by evolution in order to maximize the amount of neocortical surface available.

The neocortex is packed with neurons and supporting cells such as astrocytes, microglia, and oligodendrocytes. It is estimated that the human neocortex contains around 30 billion neurons [Kandel et al., 2000]. It is the synchronous activity of those 30 billion neurons what construct who we are: our memories, hopes, knowledge, skills and accumulated life experience. At first sight, the neocortex does not present any particularly noticeable landmark except the giant fissure that separates the two cerebral hemispheres and the prominent sulcus that divides the back and front regions [Kandel et al., 2000]. But the rest of the surface looks surprisingly similar: convoluted. This fact is quite remarkable taking into account that different parts of the cortex are involved in different types of information processing and cognitive behavior: the occipital part of the brain is mainly in charge of processing visual information while the auditory cortex processes sound, the parietal cortex engages mathematical operations and the prefrontal cortex is involved, among other activities, in executive function.

Knowledge about the different functional areas of the cortex emerged initially from studies of lesions on the cortex: strokes, mechanical injuries, etc. Depending

on the localization of this lesions (Brocas area, fusiform gyrus,...), different functional deficits are apparent: compromise usage of the rules of grammar in language, inability to move a certain part of the body, incomprehension of a certain abstract concept, inability to recognize faces, etc. The regions of the lesions seem to operate semi-independently from each other and a certain degree of specialization in specific aspects of perception or thought is apparent [Hawkins, 2004]. The arrangement of these parts is remarkably similar among different humans although the functions are rarely clearly delineated. Topologically and functionally, they are arranged as a branching hierarchy [Douglas and Martin, 2004].

The hierarchical organization of the cortex is a fundamental principle underlying its topology and functionality. In an abstract categorization, a hierarchy specifies that some elements are above or below others. Here, above and below do not refer to their physical arrangement, rather to the way they connect between them. Lower areas feed information towards higher areas and higher areas provide feedback information to lower areas using a different connection pattern [Kandel et al., 2000]. There are also lateral connections between areas that are in separate branches of the hierarchy [Douglas and Martin, 2004]. A detailed map of the monkey visual cortex is shown in Figure 3.2.

The lower functional regions in the neocortex are the primary sensory areas [Kandel et al., 2000]. For instance, visual information enters the cortex through the primary visual area [Kandel et al., 2000], known as V1, see Figure 3.2. V1 is engaged in the recognition of low level visual features such as tiny bits segments, small scale components of motion and basic color and contrast information. V1 feeds information to higher areas such as V2. The study of higher areas has shown that they are involved in the recognition of objects of intermediate complexity such as for instance star shapes [Hawkins, 2004]. The highest areas in the visual pathway represent visual memories of complete objects such as faces, tools, animals and so on. Other senses posses similar hierarchies. Sensory information gets aggregated in association areas were two or more sensory pathways are added up and integrated into a combined representation [Kandel et al., 2000]. Another area of the cortex whose functionality at first sight may seem very different from the rest is the motor cortex, which is in charge of generating motor commands directing muscle movements. Yet, both the hierarchical structures present in the motor area and the sensory area look remarkably similar [Hawkins, 2004].

Most schematic descriptions of the brain oversimplify the nature of the hierarchies within. In general input such as sound, sight or touch enters through the primary sensory areas and moves up in the hierarchy towards the association areas from which they are channeled towards the frontal lobes of the cortex to finally reach the motor cortex and then the information flows out towards the motor neurons in the spinal cord [Douglas and Martin, 2004]. But this view overlooks the fact that information flow often travels in both directions with more projections feeding back down the hierarchy than up [Douglas and Martin, 2004].

Neurons possess a cell body and branching processes forming wired like structures. These processes are different depending on whether the information flows in (dendrites) or the information flows out (axons), see Figure 3.3. At the junction

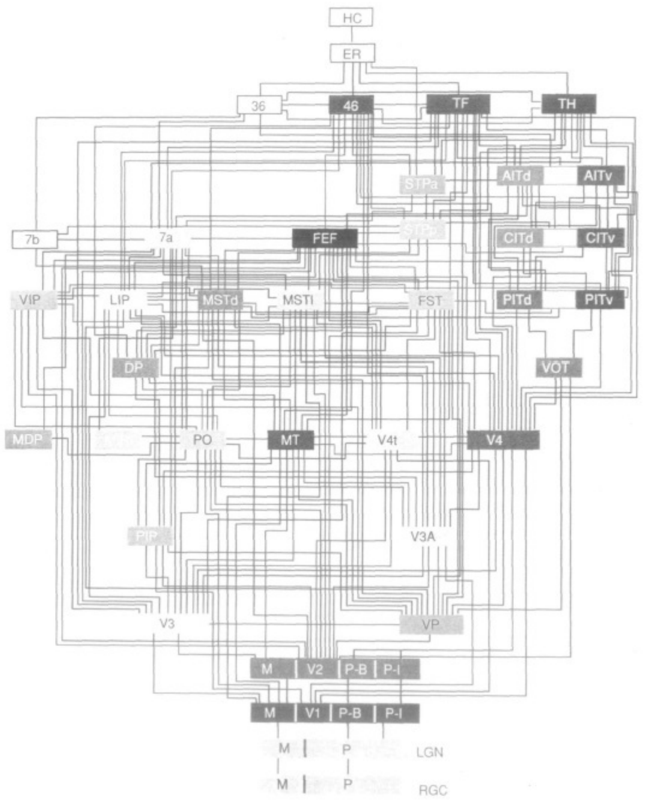


Figure 3.2: **Visual cortex' hierarchy.** Schematic diagram illustrating the hierarchy of the cortical areas involved in vision in a rhesus monkey visual cortex. The hierarchy rises from the primary visual cortex at the bottom of the figure all the way up to the highest levels of associative processing at the top. The "where" path is illustrated on the left of the figure and the "what" pathway is illustrated on the right [Felleman and Van Essen, 1991].

between the axon of one neuron and the dendrite of another neuron, a connection called a "synapse" is formed. This is how the activity of one neuron influences other neurons. An incoming neural spike at a synapse increases the likelihood for the recipient cell to spike [Kandel et al., 2000]. However, some synapses show the opposite effect, with incoming spikes making it less likely that the receiving cell will spike [Bear et al., 2002]. Hence, synapses are named according to the effect that they cause in the receiving cell as excitatory or inhibitory. Synaptic strength can change depending on the temporal behavior of the two cells [Bear et al., 2002]. This effect is apparent when two neurons generate temporally adjacent spikes, which increases the connection strength between the two neurons. This effect is called Hebbian learning [Hebb, 1949; Auyeung et al., 1987].

There are many different types of neurons in the cortex, but one type shows an overwhelming prevalence over any other, that is the pyramidal neuron. Pyramidal neurons compromise 80% of all the neurons in the cortex [Kandel et al., 2000]. They own their name to the shape of their cell bodies (that roughly approximate

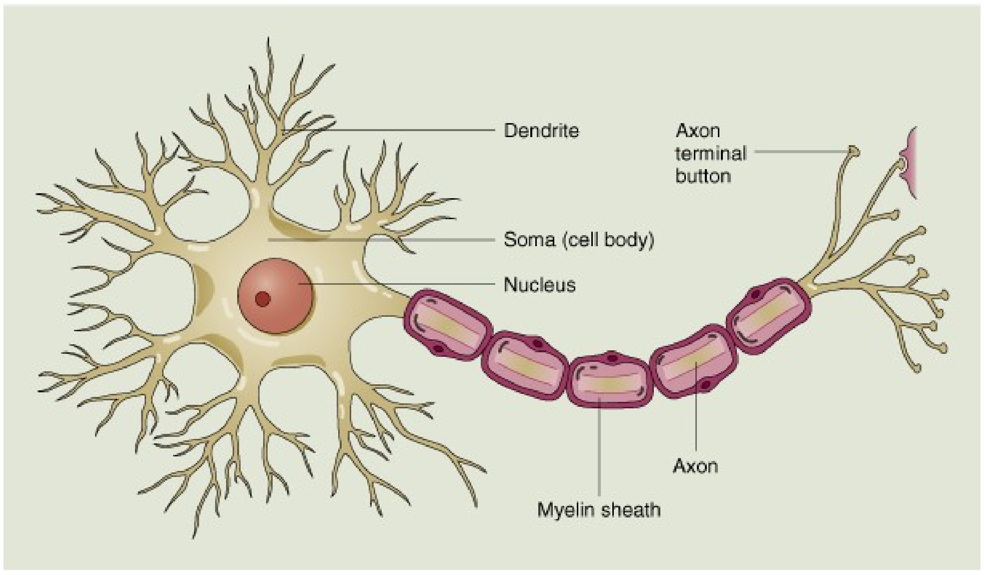


Figure 3.3: **Schematic diagram of a neuron.** A typical neuron is divided into three parts: the soma or cell body, dendrites, and axon. The axon and dendrites are filaments that extend from the soma. A synapse signal (excitatory or inhibitory) occurs at the contact between the axon of one neuron and a dendrite or soma of another. If the net excitation received by a neuron over a short period of time is large enough, the neuron generates a brief electrical pulse called an action potential through its axon.

the shape of a pyramid). All the six layers in the neocortex except layer 1 which contains mainly axons, contain pyramidal cells. Pyramidal neurons connect to many other neurons in the immediate neighborhood and to distant cortical regions or lower brain structures [Kandel et al., 2000].

In 1978 a neuroscientist named Vernon Mountcastle published an article “An organizing principle for cerebral function” [Mountcastle, 1978]. In this paper Mountcastle delineated a profound insight about the neocortex. Mountcastle argued that the neocortex is remarkably uniform in appearance and structure [Mountcastle, 1978]. That is to say that the regions in the cortex that handle auditory input appear remarkably similar to the regions that handle touch, vision, motor control or executive function [Hawkins, 2004]. Mountcastle revolutionary idea was the hypothesis that if all the regions look cytoarchitectonically similar, perhaps they are performing the same basic set of operations, “computational algorithm”, to carry out their apparently different functions [Hawkins, 2004].

Different regions of cortex do differ among themselves according to different metrics: layer thickness, layer cell density, cell types, horizontal connections’ length or synapses density [Mountcastle, 1978; Kandel et al., 2000]. However, given the association of these regions with a very diverse set of cortical functions, what stands

out is how relatively similar they are overall rather than the marginal differences that differentiate them. These differences are often so subtle that even trained neuroanatomist can not even agree on them [Hawkins, 2004]. Mountcastle hypotheses was that all regions of cortex are performing the same operations, and what makes them a vision or motor area is how these regions are connected to each other and to other parts of the nervous system [Mountcastle, 1978]. In fact, Mountcastle argued that anatomical differences among cortical regions stem from the different connectivity patterns of each area. Mountcastle was not arguing that vision is the same as listening or motor output, he was just suggesting that the cortex process signals from different sensory inputs in a similar way [Hawkins, 2004].

Mountcastle ideas have not enjoyed widespread support in the neuroscience scientific arena over the last decades. Often, efforts in neuroscience have mostly concentrated in the determination of where in the cortex a certain function takes place and not so much to when does it take place and how [Hawkins, 2004]. Furthermore, most studies published in the last decades using fMRI have favored the notion of a brain divided in a collection of highly specialized modules [Hawkins, 2004].

Although most of these studies have not placed a lot of emphasis on how the brain carries out a particular task, evidence exist to suggest that the brain uses a similar mechanism to carry out different tasks [Hawkins, 2004]. Any given brain, placed in the right context, can learn one among 100 different languages, sign language, mathematical language, written language, musical language, body language or a computer language [Hawkins, 2004]. The neocortex can learn to survive in very different environments, it can become an expert in fishing, hunting, farming, architecture, or engineering [Hawkins, 2004]. Evidence suggests that the brain has an incredible capacity to adapt to different environments by creating a model about them. This suggests the notion of a brain as a flexible system that is able to organize its knowledge representations according to the particular environment in which it is placed [Hawkins, 2004].

Physiological evidence also piles up to support this notion of flexibility. The wiring of neural networks is extremely plastic, since they are capable of rewiring themselves according to the spatio-temporal structure of electrical signals flowing in. The work from [Newton and Sur, 2005] proves that newborn ferrets brains can be surgically rewired to send visual input to regions of the cortex that normally handle auditory input and that in these conditions the ferrets developed normal visual capabilities. This shows that ferrets can develop functionally capable visual pathways in anatomical areas normally in charge or processing auditory information [Hawkins, 2004]. All this suggests that regions of cortex are not predetermined to specialized in vision, touch or hearing according to anatomical positioning but rather that connectivity creates the specialization [Shepherd, 2004].

The previously described properties apply to the human neocortex as well which also shows a high degree of plasticity. For example, humans that are born deaf, process visual information in what otherwise would be normal auditory regions and people who were born blind, use the occipital part of the cortex (which in non blind individuals is used to process visual information) to read Braille [Hawkins, 2004]. All this evidence seems to suggest that regions of cortex become specialized due to

the type information that flows in during nervous system development.

Additional evidence to support Mountcastle ideas for a common "cortical algorithm" arises from the fact that although information arriving in the cortex from different sensory inputs may seem very different at first sight: sound, vision or touch; in fact, all that information arrives into the cortex through bundles of nerve fibers transmitting spatio-temporal codes by means of electrical signals. These neural signals are called action potentials or spikes. The sensory organs that generate input signals travelling towards the central nervous system are different among themselves, but once these signals are turned into action potentials, they are just patterns of electrical activity codifying spatio-temporal data structures [Hawkins, 2004].

An easy way to visualize what is meant by spatio-temporal patterns is to think about vision. The visual pathway transfers both spatial and temporal information [Hawkins, 2004]. Spatial patterns are simultaneously occurring patterns: they are generated when multiple receptors in the same sensory system are activated simultaneously. In vision for instance, the co-activation of several receptors in the retina at the same time create a spatial pattern. This pattern is relayed to the brain. Vision also transfers temporal patterns. Temporal patterns are just the time dynamics of spatial patterns over time. These changes however, are not random but are defined by the constraints of the physical world. Sound information coming into the cortex is also composed of spatio-temporal patterns. We usually think of sound as just a temporal pattern but the auditory sensory receptors do react specifically to the different frequencies composing a sound and hence, that represents the spatial pattern of sound stimuli. Touch sensory input also has a temporal component. When recognizing an object just from touch information without visual information, we move the object around our fingertips or move the fingertips around the object in order to identify the object. Without these changing patterns of information it is very difficult to identify the object. Hence, sensory information coming from touch sensory organs are also composed of spatio-temporal patterns. To sum up, the cortex does not possess its own interface to sense the world directly, all that it operates with are electrical patters encoding spatio-temporal information through bundles of axon fibers.

3.2 Memory and Functions of the Neocortex

Often, an analogy has been traced about the similarities between brains and computers. However, this analogy can be severely misleading. Neurons are relatively slow to process synaptic inputs and generate action potentials when compared to the operating speeds of transistors in digital computers [Hawkins, 2004]. The basic operation of a computer is much faster than the basic operation of a neuron [Hawkins, 2004]. It takes a neuron about 5 ms to process an input and generate an output. Humans can recognize an object in a picture in about half a second. That means that visual information entering the brain transfers a chain of about 100 neurons at most during image recognition. A digital computer would require billions of serial steps to not even come close to the performance levels of the brain. The brain solves

the problem however in just a few steps, but using massive parallelization [Hawkins, 2004].

A hypothesis lied out in [Hawkins, 2004] postulates that the brain does not compute answers to recognition problems, since this would take a long time. It seems that rather the brain retrieves the answers from memory. The idea is that patterns were stored previously in neural networks and neurons just perform pattern matching to infer what information to retrieve from memory. Hence, neurons are not just relays of sensory information but rather they constitute memory itself. A memory that has been imprinted through life history in their structure and patterns of connectivity. According to this view, the entire cortex is therefore a probabilistic memory system not a computer at all [Hawkins, 2004].

We will now contemplate the difference between computing a solution to a problem and using memory to solve the same problem, for instance, catching a ball [Hawkins, 2004]. Solving this problem involves perceiving the ball travelling towards us and snatching it out of the air. The task does not seem specially difficult at first sight, yet it becomes cumbersome when trying to program a robotic arm to replicate it. Engineers tackle this problem by calculating the flight trajectory of the ball in order to determine the position in space that it will occupy when it comes close to the arm using a set of differential equations. The process has to be repeated iteratively as the ball approaches and sensors get more up to date information about the ball location and trajectory. A computer requires millions of serial steps to address the differential equations involved in solving this problem of catching the ball. It is worth recalling now that the brain solves the very same problem in just a few steps (recall the 100 step rule from above) [Hawkins, 2004] by using memories of muscle commands required to catch a ball [Hawkins, 2004]. When the ball is thrown, the appropriate memory is retrieved by the sight of the ball, a temporal sequence of appropriate muscle commands is generated and this sequence is adjusted to accommodate the particulars of the moment (ball's actual path, position of the body, etc). The memory of how to catch a ball was not pre-programed genetically into the brain; it was learned over several years of life history and not calculated by neurons [Hawkins, 2004].

The previous example has been used to dismount the deceiving view of the brain as a computer. Let us now elaborate on what it is known about how the brain is actually organized and how does it process information. Evidence emerging from neuroscience research has established a few principles that the neocortex uses to carry out its functioning tasks [Hawkins, 2004]:

-The neocortex stores sequences of patterns: Memories are stored in the cortex in the form of sequences. It is impossible for a human to retrieve from memory all the details of an event simultaneously. A walk through the temporal sequence is necessary to recall complex memories. The temporal aspect of the sequences is very important. If we try to recite the alphabet backwards, we usually fail because we are not used to experience that sequence backwards. We store in our memory the alphabet as a sequence of patterns [Hawkins, 2004]. That memory can not be recalled all at once, or in an arbitrary order. The same principle applies to other memories: days of the week, the months of the year, telephone numbers, songs,

and also low degree sensory memories [Hawkins, 2004]. The storage of memories as sequences of patterns is therefore an inherent aspect of the neocortical memory system [Hawkins, 2004].

-The neocortex recalls patterns auto-associatively: An auto associative memory system can recall complete spatio-temporal patterns when given only partial or distorted inputs [Hopfield and Tank, 1985; Kandel et al., 2000]. Our cortex is able to complete visual spatial patterns from partial versions of it. Our brain also completes temporal patterns. If we recall a fragment of a story, the entire memory sequence is subsequently retrieved [Hopfield and Tank, 1985]. When talking with other people, our brain is not able to hear all the words in the conversation, specially if we are located in noisy environments, but the brain fills in the missing parts with what it expects to hear, although we are not usually aware that our brain is constantly completing patterns [Hawkins, 2004].

-The neocortex stores patterns in an invariant form: Another characteristic of neocortical memory is how it is able to form what the neuroscience research community refers to as invariant representations. Digital computers store information exactly as it is presented. Bytes are stored with absolute fidelity, with single errors disrupting the whole memory [Hawkins, 2004]. The neocortex does not store complete-fidelity memories of things. It rather remembers significant associations in the physical world, independent of details [Hawkins, 2004]. Artificial auto associative memories fail to recognize patterns if they are significatively moved, rotated, rescaled or transformed in any of numerous different ways [Hawkins, 2004], yet our brains are able to handle robustly these variations. The cortex therefore has a remarkable ability to perceive things as belonging to the same ultimate cause although the input patterns flowing in are constantly changing [Hawkins, 2004]. This principle can be illustrated through an example: looking at a face. The pattern of receptor cells activated in our retina, as the face moves around us or we move with respect to it, is very different from time to time. Yet, our neocortex has no problem at perceiving an invariant representation of that face. The brain's internal representation of the face does not change even though the stream of light informing the neocortex of the presence of the face is in constant flux. Scientists refers to these stable representations as invariant. Every time, an image moves or our eyes make a new fixation, the pattern of activity in V1 changes, as a result of the changing sensory pattern arriving from the retina [Kandel et al., 2000]. However, if we monitor the activity of cells in the face recognition area, a functional region several steps higher than V1 in the cortical hierarchy, stability is apparent. Some set of cells in the face recognition area remain active as long as the face is anywhere in our field of vision regardless of its size, position, orientation, scale an expression. These stability of cell firing is an invariant representation [Hawkins, 2004] of that particular face. The problem of understanding how the cortex forms invariant representations remains one of the biggest challenges in neuroscience. Hence, it can be hypothesized that the way our neocortex understands the world is by finding invariant structure in a constantly changing stream of input [Hawkins, 2004]. However, invariant representations by themselves are not enough as a basis for making specific predictions and the brain has to combine knowledge of invariant structure with recent details [Hawkins, 2004].

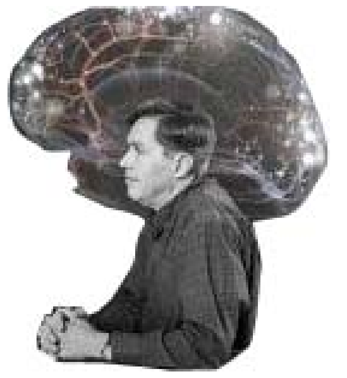
The brain uses stored memories to constantly make predictions about everything we see, feel and hear [Hawkins, 2004]. The vast majority of predictions occur outside awareness. Prediction in fact is so pervasive that we can derive the profound conclusion that what we perceive (how the world appears to us) does not come solely from our senses. What we perceive, is a combination of what we sense and of our brains memory derived prediction [Hawkins, 2004].

-The neocortex stores patterns in a hierarchy: Hierarchical organization leads to shared representations, generalization capabilities and storage efficiency. Causes at lower levels of the hierarchy are shared among higher level causes, significantly reducing the amount of memory and time required to learn new causes and providing a mechanism for generalization [Hawkins, 2006]. The hierarchical structure matches in fact the spatial and temporal hierarchy of the real world [Hawkins, 2006]. Objects in the real world do possess a hierarchical structure. A human is composed of legs, trunk, arms and head. A head in turn is composed of hair, eyes, mouth, ears and eyes. Last but not least, a hierarchical representation affords a mechanism for attention [Hawkins, 2004].

We can summarize the previous discussion by concluding that intelligence can be defined as the capacity to remember a variety of patterns in the world and make predictions about their likely evolution. To make predictions of future events, the neocortex needs to be able to store sequences of patterns. The correct recalling of memories requires the ability to retrieve patterns based on their similarity to past patterns (auto associate recall). Memories also have to be stored in an invariant form, efficiently, so that abundant knowledge from past events can be applied to new situations that are similar but not identical to the past [Hawkins, 2004]. The hierarchical structure contributes to the efficient storage and recalling of information. The physical neocortex accomplices all these requirements and others not discussed here or known yet. Hierarchical temporal memory tries to encapsulate those known concepts into a set of algorithms to carry out pattern recognition and sequence prediction in-silico.

Chapter 4

Hierarchical Temporal Memory



4.1 HTM Overview

This chapter provides a succinct overview of Hierarchical Temporal Memory, HTM, a machine learning algorithm bioinspired on some of the principles that govern neocortical topology and information processing. HTM was originally developed by Jeff Hawkings and Dileep George. The authors stated purpose is the far fetched goal of replicating on a piece of hardware or software some of the fundamental principles that the human brain uses to carry out its functions [George and Hawkins, 2009].

The main justification for the emergence of HTM theory is the abundance of cognitive functions that humans perform with apparent ease and that computers are not yet able to replicate at similar levels of performance [Hawkins, 2006]: visual pattern recognition, navigation in a 3-D environment, comprehension of language, executive planning and manipulation of objects just to name a few. In humans, these abilities reside largely in the neocortex. Hierarchical Temporal Memory proposes a hypothesis of neocortical function that can be formalized on a set of algorithms ready to be implemented on a computer.

HTMs can be thought of as a memory system that can carry out classification and prediction based on spatio-temporal statistics learned during training sessions

[George and Jarosy, 2007]. HTMs are organized as a tree-shaped hierarchy of computational nodes. Each node implements a common learning and memory function [Hawkins, 2006]. The HTM topology creates a hierarchical spatio-temporal model of the data it has been exposed to during training. The fact that HTMs try to capture spatio-temporal characteristics within a training set makes this technology applicable to a vast array of problems. Any problem composed of a hierarchical structure in which there are spatial relations between different variables and non-random fluctuations through time is amenable for HTM to create a model of it. Yet, the capabilities of an HTM are largely limited by what kind of data it has been exposed to during training [Hawkins, 2006].

HTMs can be instantiated in different forms. In this thesis, we develop a simplistic binary HTM that uses binary vectors for information transfer between nodes as explained in this Chapter and Chapter 6. HTMs can also use more sophisticated learning models employing Markov Chains to model temporal groups and belief propagation to shape information transmission in the network in a probabilistic fashion. This is the implementation developed by HTM original inceptors at Numenta[1] and employed in some of the experiments shown in Chapters 7, 8, 9 and 10. Finally, to deal with multivariable time series, HTMs can be implemented with an extended top node as described in Chapter 7 using a binary HTM as a primitive. That type of HTM is the specific contribution of this thesis to the field of HTM and its inner workings will be specified in Chapter 7. In the next section and in Chapter 6 we will illustrate the inner-workings of a binary HTM on a simple problem of image recognition, since it is easiest to visualize the main properties of HTMs in this type of problem.

4.2 HTM Theoretical Principles

HTMs could be theoretically used not only to mimic mammalian sensory systems, but also to create artificial senses that sample a wide array of data structures: weather forecast, stock-market analysis, or surveillance systems among others. Nonetheless, in the following description we will often illustrate HTM theory in terms of a visual pattern recognition task, since it is a well studied field of research, and it is easy to visualize and grasp fundamental HTM concepts when being shown in the particular task of vision [Hawkins, 2006].

Although the neocortex performs a wide array of functions going from pattern recognition of sensory input to executive control of behavior, as explained in Chapter 3, cytoarchitectonically all different parts of the neocortex look surprisingly similar [Douglas and Martin, 2004; Mountcastle, 1978] and HTMs theory lies its roots from the theoretical prediction that all of the neocortex works on a common neocortical algorithm [Mountcastle, 1978].

HTM theory speculates that the neocortex functions by creating a model of the particular input patterns to which it is been exposed through the sensory organs. The system uses then that model to make predictions about the future based on

[1]http://www.numenta.com

the spatio-temporal statistics learned during training and the current state of the model as it senses its surroundings in real time. These principles are simple to replicate at a low level in a computer. However, replicating them at higher levels has proved extremely complicated. If it were to truly mimic neocortex functionality, HTM algorithms should be able to [Hawkins, 2006]:

- Discover causes in the world
- Infer causes of novel input
- Make predictions
- Direct behavior

Figure 4.1 shows a conceptual view of an HTM system. Basically, an HTM system divides the world in a hierarchy of so called “causes”. A cause is just a persistent and recurrent pattern in the world [Hawkins, 2006]. Causes can be physical such as humans, animals, buildings or not physical such as ideas, words or songs. The common feature to all these causes is that they are persistent, i.e. they exist over time [Hawkins, 2006]. Therefore, causes are the ultimate culprits for the flow of electrical information in real neural networks [Hawkins, 2006].

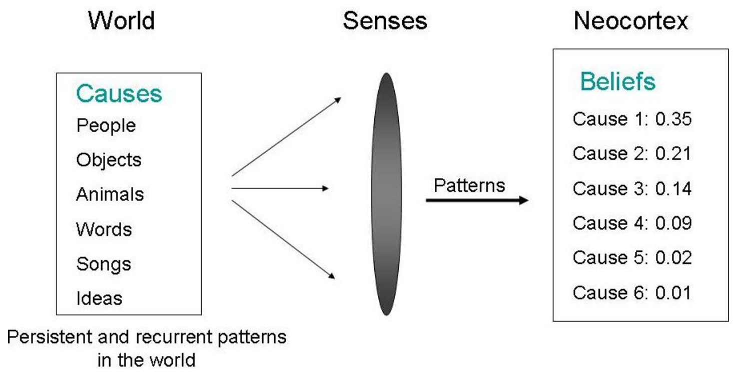


Figure 4.1: **HTM model of the world.** The world is composed of a wide array of causes. The function of an HTM is to cluster different input data into invariant representations of different causes categories and being able to perform inference on unseen instances based on this clustering performed during training.

Discover causes in the world:
As indicated in Figure 4.1, an HTM interfaces the world through a set of senses. The senses sample attributes of the world, (for instance light intensity or sound frequency). Based on the pattern of data flow that the system senses at any moment in time and using the model previously created during training, the system creates a set of beliefs about what it believes are the ultimate causes of the novel pattern to which it is being exposed.

HTMs use time as a key cue to cluster different incoming patterns into the same ultimate cause, therefore, it is essential during learning that the sensory data

changes through time. At the inference stage, time changing data is not required, but it can improve the efficiency of the system. It is important to point out that eyes saccades also improve the performance of humans during vision recognition tasks [Hennessey et al., 2008]. Therefore, an HTM receiving spatio-temporal image patterns should receive them over time in order to create a model of the ultimate causes likely to generate current input patterns. In an HTM, causes are represented by vector elements [Hawkins, 2006].

HTM theory argues that discovering causes is at the heart of any intellectual activity. According to this view, the world is composed of a hierarchy of causes, low level causes are relatively easy to comprehend, e.g. learn to cluster all types of dogs that can be encountered into the category "dog". A middle level cause, would be to grasp the relationship between a person's behavior and the possibilities that that behavior impulses a dog to bite us. Higher level causes would involve for example the understanding of the high level causes governing the behaviour of markets or any causal relationships among multivariable systems. Through the hierarchical HTM system, low-level causes are learned at the bottom of the hierarchy while high level causes with more subtle spatio-temporal dynamics are learned higher up into the hierarchy and are composed of spatio-temporal relations between the low level causes [Hawkins, 2006].

Infer causes of novel input:
As in human sensory systems, a particular pattern of sensory input to an HTM system at a given time instant will likely always be novel. The visual system in humans is composed of millions of receptor cells in its retina. Similarly, a camera may have up to several million pixels in its optical sensor. It is unlikely that precisely the same pattern of input pixels will be presented to both systems twice. Therefore, after an HTM system has been trained, the system should be able to perform inference on novel sensory input using the internal model of the sensory world generated during training. That is, the system should be able to assign one of the learned causes as the most likely ultimate culprit of the particular sensory input to which the system is being exposed. More specifically, the result of the computation should be a discrete probability distribution among all the causes the system was exposed to during training with each probability-cause pair indicating the likelihood that a specific cause is the ultimate cause of the present input pattern. If the sensory input data would be highly non-ambiguous, the distribution should be peaked at just one or a few causes, otherwise the distribution would probably be flatter.

It has been mentioned before that time-varying sensory input is absolutely essential for HTM training, since the discovery of causes requires continuously changing inputs. This is not the case for inference, yet the performance of the system should improve when given data that varies over time. It is important to point out again that the same happens to humans. If a human subject is exposed to a vision recognition test, the performance of the subject will improve if the images are present in the subject's retina long enough for his eyes to perform saccadic movements over the images [San Agustin, 2010].

Make predictions:
Once an HTM has built a model of its world, it can perform predictions about how

a current state of the variables might change over time using the model stored on memory. Theoretically, each node should store sequences of patterns that are likely to follow each other. This is more easily said than done, and particular shortcuts are used on the implementation of this theoretical requirement to avoid an exponential explosion on memory requirements. The sensory data being perceived by an HTM system at any time point can be compared with the stored sequences to predict what might happen next. This ability is highly useful to create prior probabilities to what may happen next based on the current state of the nodes. For example, an HTM system designed to process spoken language would automatically create prior probabilities about words and phonemes that are likely to follow a particular word or sentence. Using this prior can help enormously to interpret noisy or missing data [Hawkins, 2006].

Direct behavior:
Last but not least, all the capabilities recently described can be used by a piece of software to produce and control the behavior of robotic machines or agents in virtual worlds.

Our implementation of HTM in this thesis has only focused on being able to perform one of the 4 predicted capabilities, namely, our HTM system is able to discover causes in the world and to infer causes of novel input.

4.2.1 Discovery and Inference of Causes in HTM Theory

An HTM system can be thought of as a hierarchy of nodes as shown in Figure 4.2. Each rectangle in the hierarchy represents a node, the computational heart of an HTM network. Each node performs the same function all the way through the hierarchy except the top node which requires a supervisory signal during training. Specifically, each node can carry out two distinct phases, one of learning in which it memorizes recurrent spatial patterns and the sequences of spatial patters arriving to it over time (causes) and one of inference, in which the node passes up the hierarchy a vector of "beliefs" in terms of what it believes are the causes (learned during training) to which it is being exposed at a particular time instant [Hawkins, 2006].

The output beliefs of several nodes become the input patterns of nodes higher up in the hierarchy. Nodes at the lowest level of the hierarchy receive data from sensors that sample a certain value of the cause to which the system is being exposed. Therefore, bottom level nodes discover causes from just a small area of the overall sensory input area. In neurobiology, this small area is known as receptive field. Nodes higher up in the hierarchy receive input from nodes underneath and store the sequences of the most common spatial patterns that tend to follow each other. Obviously, the receptive field of these nodes is bigger than the ones from the nodes below. High level nodes should capture and discover causes over the entire input sensory area. In a problem of image recognition, nodes at the bottom of the hierarchy just learn sequences of vertical bars, horizontal bars, or corners and their likely translational sequences, nodes higher up in the hierarchy memorize more complex geometries by combining lower level causes [Hawkins, 2006].

Nodes cluster the space of stored spatial patterns by clustering them into groups,

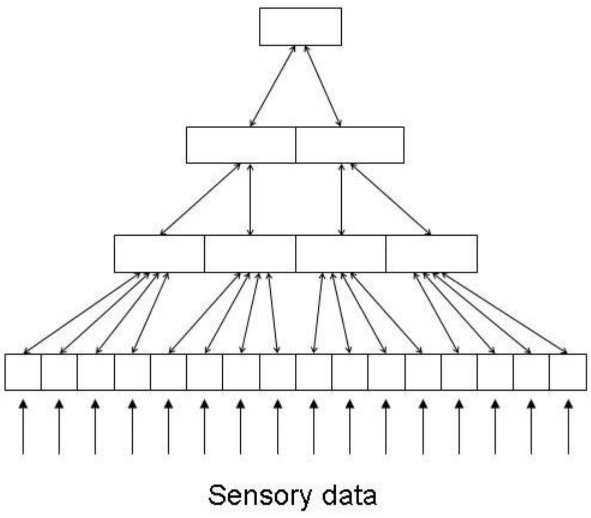


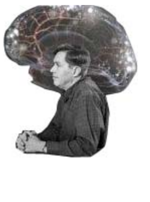
Figure 4.2: **HTM as a hierarchy of nodes.** HTM hierarchy with 4 layers. Each layer is composed of several nodes (represented by a rectangle), except the top layer which contains just one node

not in terms of geometric distances but rather according to their temporal proximity (whether they tended to follow each other during training). During inference, a node exposed to sensory input, carries out a set of operations converging towards a definite and discrete probability distributions over the stored set of invariant representations (each invariant representation being composed of several spatial patterns). The value of this probability represents the likelihood of each cause being the ultimate culprit of the particular sensory input information that the node is receiving. The node passes then this information higher up into the hierarchy in form of a vector, with each component representing the particular probability that a certain cause is being sensed at that particular time by the node [Hawkins, 2006].

Training proceeds in a sequential manner, since nodes higher up in the hierarchy can not start their learning/training phase until their children nodes, from which they get their input patterns, have gone from the learning phase to the inference phase.

The basic operation of each node during inference is divided in two steps [George and Jarosy, 2007]. First the nodes assigns its input pattern to one of a set of quantization points, which represent common spatial patterns that entered the node often enough during training. The node assigns a probability to each one of the learned spatial patterns quantifying the likelihood that the particular instance being perceived at the present time instant belongs to a certain stored spatial pattern. The node then observes sequences of spatial patterns arriving over time, and assigns a probability that they belong to one of several stored sequences of spatio-temporal patterns learned during training [Hawkins, 2006].

A node can also send information to its children in terms of what spatial pattern it believes it is being exposed to, this information can be used by children nodes to



disambiguate noisy situations during inference [George and Jarosy, 2007]. Beliefs from the upper nodes can help to disambiguate flat probability distributions over the set of clusters of spatial patterns. The parent node is able to do this, since by means of the hierarchical structure of the system, possesses a broader receptive field and therefore can neutralize noise over a limited spot within its receptive field [Hawkins, 2006].

Nodes at the bottom of the hierarchy are exposed to fast changing spatial patterns. Nodes higher up in the hierarchy experience changes in the patterns received upon their receptive fields over a longer time scale. Therefore, nodes possess different time-dynamics along the hierarchy of an HTM system.

Parameters of the system such as the number of levels through the hierarchy, number of nodes at each level, or connectivity metrics (fan-in, fan-out parameters) are not critical to the functionality of the system. Yet, tweaking of the parameters can lead to vastly different results in terms of performance. Nonetheless, most arrangements will work to a certain extent. Also the lost of a few nodes through the hierarchy should not have dramatic results in the overall performance of the system. In this sense, one property of the HTM theory is its robustness [Hawkins, 2006].

4.2.2 The Importance of Hierarchy

Shared representations lead to generalization and storage efficiency: Computational approaches that perform pattern recognition usually have to deal with the problem of scale. As the number of features the system should be invariant to (translation, rotation, noise, size) grows, the memory and computational requirements of the system grow exponentially, leading to a quickly overflow of computational resources. HTM theory tackles this scale problem by means of a hierarchy. Since causes at lower levels of the hierarchy are shared and combined by nodes higher up, HTM systems are able to generalize by means of using previously learned causes and combining them in novel ways to represent new types of unlearned causes at the top nodes.

Let's use a simple visual example for clarification. During a visual recognition task, a node at the bottom of the hierarchy would have as its receptive field a small patch of an artificial retina that light could activate. Let's say the size of this patch is of size 4X4, that is, 16 pixels. If each pixel has two states (on/off), the amount of possible patterns that the system could theoretically perceive would be 2^{16} or 65536 different patterns. In HTM, this node would only store a small fraction of the total number of possibilities (e.g. 150). The node would only store the most frequent patterns that occur over time within its receptive field during training. These stored patterns will be called from now on, quantization points. Additional training should not increase any further the number of quantization points, but it could change them. During inference, the node would look at a new pattern, one out of the 65536 different possibilities, and decide using a distance metric how close it is to each one of the 150 stored quantization points.

It is important to point out, that a node at this stage of the hierarchy is just able to recognize, vertical or horizontal bars, crosses or corners nor houses, or faces. But

the former are the building elements that can be combined by nodes higher up into the hierarchy to form more complex causes (houses, or faces). Notice again, that the memory used to store and recognize low level features (causes) will be shared among high level causes [Hawkins, 2006]. It is also important to point out that a node at any stage through the hierarchy can only learn causes that are combinations of the causes that its children nodes are passing up the hierarchy.

HTM Hierarchy matches the hierarchy of the physical world: The physical world is composed of a hierarchical structure. Any object we perceive through our eyes is just a combinations of sub-objects. The ultimate sub-objects whose combinations give raise to any complex shape are just lines, corners, curves and intersections. Combinations of objects give rise to higher-level objects: A set of curves, lines and intersections can be combined to draw a finger. Five fingers and a palm give rise to a hand. A hand, forearm, elbow and shoulder give rise to an arm. Putting together lips, eyes, mouth, nose, forehead, hair, chin and cheeks give rise to a head. A set of arms, legs, truck and a head give rise to a human body [Hawkins, 2006].

HTM systems not only capture spatial hierarchical relations but also temporal relations. Nodes at the bottom of an HTM find temporal correlations among patterns that occur close together both in space and in time [Hawkins, 2006; George and Jarosy, 2007].

In summary, causes that occur close together in time and space are more likely to be correlated than causes presented farther away in time and space. This is the result of the forces of physics that tend to be stronger for objects close in space and time.

4.2.3 The Importance of Time

Nodes through an HTM hierarchy learn common sequences of patterns, therefore, time changing input is necessary for learning to occur. HTMs use time as a way to create many-to-one maps or pooling. Spatial pooling is performed on each node by means of a distance metric: if two spatial patterns have sufficient overlap, they are assigned to the same cause. Temporal pooling functions by means of assigning recurrent sequences of spatial patterns to a group that functions as an invariant representation of all the spatial patterns clustered as they change over time. Time functions, therefore, as a de facto supervisor during learning. By pooling largely different spatial patterns to the same cause whenever they follow each other in time often enough, a time-invariant representation of a set of spatial patterns is achieved, this group is referred to in this work as a “cause”. Therefore, time has provided the cues to pool together very different spatial patterns into a common cause. An HTM can also be extended by adding an artificial supervisor to the system, this is biologically plausible, since even though any mammalian brain learns to pool together the very different patterns arriving to the retina and belonging to a certain physical object to the same cause, we often need an external cue (e.g. a teacher), that help us assign a common cause to very different spatial patterns. For example, someone that inform us that patterns belonging to the cause “bear” and “wolf”

can be pooled together to the higher level cause "animal". Often, learning correct categorization can be made faster by using supervised training. In HTMs this is done at the top node during training, when the top node is informed to which category the pattern being sensed belongs.

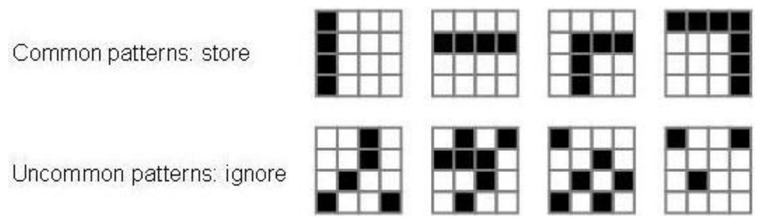
4.2.4 Discovery and Inference of Causes by each Computational Node

The main function of a node is to assign causes to persistent or recurrent patterns of input, that is, patterns that happen repeatedly over time. A node distinguishes and stores two type of patterns: spatial patterns and temporal patterns. If a node at the bottom of the hierarchy has 16 potential inputs and a set of 4 of those 16 inputs become active at the same time over and over again (with a frequency greater than what would be expected by chance). The node infers that they must share a common cause or part of it, and the node stores these spatial patterns in its memory. The amount of memory of the node is limited, so it only stores a fixed amount of spatial patterns through the learning stage. When a new input arrives at the node, the node calculates how close this new pattern is to each one of the stored quantization points on its memory and depending on whether it is close enough to some of the stored patterns or whether it occurs often enough it decides whether to memorize the pattern or ignore it [Hawkins, 2006].

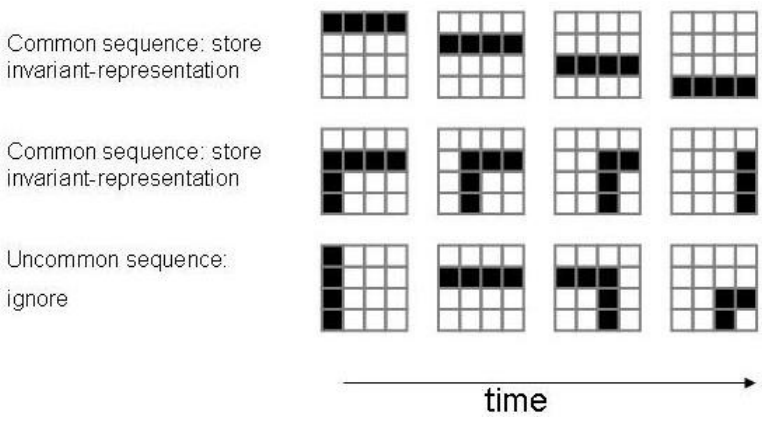
Now let's suppose that over time, the node observes that a certain pattern, say "a3" is often followed by pattern "c5", which itself is followed by " f1". If this happens a far greater number of times than chance would allow, the node creates and invariant representation of the patterns arriving over time in the form of a "temporal pattern" (cause). A temporal pattern is a cluster of temporally adjacent spatial patterns that represents an invariant representation of all of them. The rationality for this approach is that spatial patterns that tend to follow each other in time are likely to share a common cause [Hawkins, 2006].

Figure 4.3 shows the two basic operations of a node in terms of a visual example. The 4x4 grid, represents the receptive field of a node in the bottom of and HTM vision hierarchy. A black box, means light being projected at that point and a white box represents the lack of light. Subfigure 4.3(a) shows a node storing common and recurrent spatial patterns that fall over its receptive fields while uncommon patterns of light arrangements are discarded. Furthermore, the node memorizes sequences of spatial patterns that often follow each other as seen in Subfigure 4.3(b). The node stores the sequence of a horizontal line moving down and a corner moving towards the right, while the uncommon third row of spatial patterns that often do not follow each other when trained with normal images are discarded [Hawkins, 2006].

In summary, each node through the hierarchy learns to store the most likely spatial patterns of its input and the most likely sequences of those spatial patterns which are also stored. The node's output that percolates up the hierarchy represent a distribution of probabilities over the set of all learned common sequences indicating the likelihood of each one being present at that particular time instant.



(a)


Common sequence: store invariant-representation
Common sequence: store invariant-representation
Uncommon sequence: ignore
time

(b)

Figure 4.3: **Node cause discovery**. The node stores common (repetitive) spatial patterns in the particular input space it is exposed to. Panel *a* shows that common patterns in an input space of 2 dimensional images formed by segments would be formed by horizontal and vertical lines, corners, etc. Uncommon patterns for that type of input space, would be disregarded by the node. Panel *b* shows common temporal sequences being clustered and stored while uncommon sequences are discarded.

4.3 Comparison of HTMs with other Machine Learning Algorithms

The no free lunch theorem [Ho and Pepyne, 2002] states that no learning algorithm is inherently better than any other algorithm for all types of problems. What determines the performance of an algorithm is the set of assumptions that the algorithm exploits in order to learn the model of the training data to which it is expose to during training. Generalistic models are usually harder to train while very specific algorithms tend to over fit the model to the training data [Numenta, 2006a]. An ideal learning system that can be applied to a variety of situations needs a set of assumptions general enough to apply to a large class of problems but specific enough to match the characteristics of real world data [Numenta, 2006a].

Evolutionary processes have shaped a set of assumptions in the human brain that make it a very efficient learning mechanism for a wide variety of problems. Some

of these assumptions are reflected in its hierarchical organization both in time and space. Machine learning algorithms that exploit the hierarchy in time and space existing in the physical world should be able to exploit that structure to learn about the world [Numenta, 2006a]. This is what the neocortex does and what HTM also tries to replicate.

We delineate next some of the similarities and differences between HTM and existing machine learning algorithms:

4.3.1 General Purpose Probabilistic Models

Different types of probabilistic models that analyze statistical relationships among sets of variables exist [Numenta, 2006a]. The variables can represent states of a category or counters of a variable. These models do not require or specify a particular relationship among the variables but they work better when they can exploit conditional independence between the variables.

- **Bayesian networks [Pearl, 1988]:** Use acyclic graphs for topological organization. These graphs encapsulate the independence assumptions about probability distributions.
- **Energy based models [Lecun et al., 2006]:** Although technically they are not probabilistic models, there is a rough equivalence between energy based models and probabilistic models. Energy based models are more convenient for certain large class problems than probabilistic models.
- **Hierarchical hidden Markov models [Fine, 1998]:** model time in a similar way to HTM. However they only use the hierarchical topology in a temporal way. HTM use hierarchy in both space and time. Additionally, HTM provides a framework where action and attention can be implemented.
- **Boltzmann machine and Hellhmholtz machine [Hinton and Sejnowski, 1986; Dayan et al., 2010]:** Abstract energy based models with a neural instanciation. This models use stochastic sampling to learn a probability distribution. Historically they have struggled with the issue of getting stuck in mediocre solutions. These approaches do not use the temporal structure of data and do not incorporate any assumptions about hierarchy. They employ the assumption that determining the probability distribution of a problem is the essence of learning.

General purpose probabilistic models are very good tools for simple probabilistic analysis, but they are too broad for several real world problems. HTMs do not conflict with the fundamental principles of these models but HTMs make additional assumptions about the nature of the world.

4.3.2 Non Generative Models

Several approaches in machine learning skip the phase of building a model about the training data that can describe the input and directly try to map inputs to the correct answer, using some sort of supervised learning for this mapping during training. These models are collectively called discriminative models. These models are mainly heavily supervised while HTM is fundamentally unsupervised (although supervision can be applied at the top node of the hierarchy but this is not strictly required, and almost all learning is performed in an unsupervised fashion). Furthermore, the HTM can generate data and it is able to predict the next stage of a temporal sequence.

- **Support vector machines [Burges, 1998]:** SVMs are very efficient algorithms to find boundaries in high dimensional space to separate several instances according to their corresponding categories. Support vector machines do not employ any assumptions about the hierarchical structure of the training data they are exposed to or any type of temporal organization in the world and therefore do not take advantage of these properties. Since the underlying models are discriminative and not generative, they can not predict forward in time.
- **Artificial neural networks [Ripley, 1996]:** Such as the multilayer perceptrons. These are supervised learning models that are typically trained with an algorithm known as back propagation. Some types of artificial neural networks make use of space and time but they do not exploit temporal coherence. Neural networks require rather heavy sampling of the training data and often struggle with overfitting.
- **Slow feature analysis [Wiskott and Sejnowski, 2002]:** It is not a discriminative model per se but it lacks generative semantics. Slow feature analysis learns invariant features in a hierarchy using temporal slowness as the underlying principles. HTM use the same principle for learning in a hierarchy and therefore they share many features with slow feature analysis. However, slow feature analysis can not generate data and can not be used to predict forward in time.

4.3.3 Empirical Neuro-Biological Models

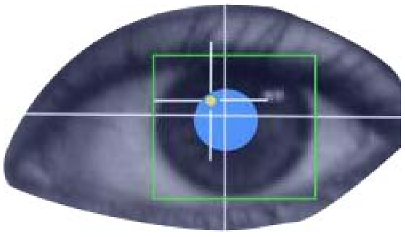
Some models try to implement observed neuro-biological behavior in a learning algorithm. A good example of this approach is:

- **HMAX [Riesenhuber and Poggio, 1999]:** is a biological a realistic model of vision. This model shares many similarities with HTMs. HMAX uses a hierarchical structure similar to HTM. Furthermore, the feature set employed in HMAX at different levels resembles the quantization points and the temporal groups in HTM. But the HMAX model obtains these basic features and the feature groups by human intervention so they possess a high degree of supervision and do not self learn the features models. Also, the HMAX

model does not implement a generative model and the inference mechanism is not probabilistic. Additionally, HMAX disregards feedback connections (that biologically are very relevant) in the model and can not predict forward in time.

Chapter 5

Example Applications Of Hierarchical Temporal Memory



5.1 Introduction

Hierarchical Temporal Memory, HTM, can be applied to a variety of problems. In this chapter, we present representative problems in which we will study the performance of HTMs. Namely, problems where the spatial representation of a pattern is complete at any give time instant (image recognition) and problems where changing spatial patterns over time conform a particular instance (sign language recognition and gaze gestures recognition).

In subsequent chapters of this thesis, we will analyze in detail how HTMs performs on image recognition, using a data set of 2 dimensional binary images subjected to different degrees of translation, distortions, occlusions and noise. We will also explore HTM performance on Australian sign language recognition using data gathered with an electronic data glove. Finally, we will employ HTMs to recognize gaze gestures in real time using a low cost eye tracking system in both head-mounted and remote setups.

In this chapter specifically, necessary background information will be provided on the basic characteristics of image recognition, sign language recognition and gaze gesture recognition. Also, we will introduce eye tracking technology and gaze based interaction as this information is a pre-requirement to understand the issues involved in gaze gesture recognition, to be treated in Chapters 8, 9 and 10.

5.2 HTMs in Image Recognition

The original developers of HTM, [George and Hawkins, 2009] first tested the HTM algorithms in the problem of image recognition. HTM proved to be very efficient at solving this type of problem as reported by [George and Hawkins, 2005]. We start the *Results* section of this thesis with a replication of the results obtained by the original authors of HTM using our own implementation of HTM.

Vision is for mammals the most fundamental sensory modality to sense the world. In humans about 30% of the neocortex is devoted to vision related areas in the occipital lobe of the brain [Hawkins, 2004]. Light rays projected into our retina depolarize receptive cells which send patterns of action potentials to the lateral geniculate nucleus from which information is relayed to area V1 of the occipital lobe. The brain deciphers and interprets these patterns of electrical activity and relates them to previously stored causes in the world or memorizes the new patterns in order to be recalled later.

A critical feature that the mammalian brain is able to performs is that of invariant visual pattern recognition. That is, the ability to recognize images despite changes in location, size, light conditions and in the presence of deformations and large amounts of noise. When a mammal perceives a certain object, usually the patterns of light rays entering the retina changes continuously through time, either the animal moves around the object, the object moves by itself, or the ligh conditions change. There are virtually an unlimited number of possible images for each specific object. Nonetheless, mammals have no apparent trouble recognizing those very different patterns of light entering our retina and the very different signals of electrical activity encoding the visual information as belonging to the same cause or object. That is, the mammal has created an invariant representation of the object in its brain that remains stable, besides changes in light conditions, shape, position in space or deformation. No computer is currently able to match the degree of performance of the mammalian brain for a visual pattern recognition task. Tackling the invariant pattern recognition problem on a computer has not been easy [Russell and Norvig, 2002]. In part because similarity metrics at the pixel level will not achieve the desired result since often images from different categories have more overlap in the pixel space than images from the same category[George and Jarosy, 2007].

Most computer science approaches towards tackling the problem in visual pattern recognition try to learn low-level statistical patterns of the images and use a supervisor method to map those features to the correct categories. A typicial approach is to store a prototypical representation for each object to be recognized. Unknown patterns undergo then a set of transformations (scaling, rotations, translations) to get them to match the prototypes. Finally a distance metric is used between the transformed unknown and the prototypes to determine the best match [Hawkins, 2006]. With these techniques, a certain degree of invariance is achieved, yet they often are not able to generalize at the degree of performance that the human brain does.

It can be concluded, that the main issue in image recognition problems is that of invariances. That is, how to train a system with a set of prototype images, and then having a recognition engine able to recognize categories properly and independently of translations in space, rotations, size changes, deformations, occlusions, changes in light conditions or contrast and the presence of noise. In Chapter 6, we show how our HTM algorithm is able to create a certain degree of invariance over the categories learnt and it is able to recognize unseen instances from those same categories that are considerably different to the training instances in terms of translations, size variations, presence of noise, occlusions and distortions.

5.3 HTMs in Sign Language Recognition

As we will discuss in detail in the *Results* section, HTMs work very robustly in problems such as image recognition in which the complete spatial structure of an instance is presented at any given time instant. We will show how HTMs are somehow limited to this type of problems and struggling on problems where the spatial structure of a sign is composed of a number of spatial states that develop over time. We explore this limitation in the problem of Sign language recognition, SLR. We addressed SLR as a good example of a problem in the physical world with a hierarchical structure, that an HTM should be able to model. Due to the limitations of traditional HTMs to robustly deal with the SLR problem, we propose an extension of HTMs for them to work better on this type of problem.

Sign language uses hand gestures for conveying information by means of sequences of hand/arm/finger shapes, position and movements as well as facial expressions instead of speech [Sarkaleh et al., 2010].

Several techniques for signal capture can be employed for automatic sign language recognition. A video camera can be used to capture frames from a signer, this approach requires considerable pre-processing of the input data to segment the hands from the rest of the image. Image processing techniques have to be sophisticated and robust to tolerate occlusions of the hands, recognize different shapes, etc. A more modern approach is to use 3D scanners which interpret 3D scene information using continuously-projected infrared structured light. The commercial success of Microsoft's Kinect is a hardware development going in that direction, although whether that cheap technology is able to discriminate fine finger movements, sometimes essential in robust SLR, remains to be seen. An alternative approach to capture data for SLR involves the usage of electronic data gloves[1] such as the one shown in Figure 5.1.

Although HTM algorithms could work independently of the methodology used to capture the sign language instances, in the work presented in the *Results* section we used a data set consisting of several instances from Australian sign language captured using a pair of electronic data gloves containing accelerometers and sensors to track 11 channels of information for each hand: x, y and z spatial coordinates of the hand, the roll, pitch and yaw rotation angles of the wrist and a bend coefficient for each finger.

[1]Data Gloves from 5DT Fifth Dimension Technologies (`http://www.5dt.com`).

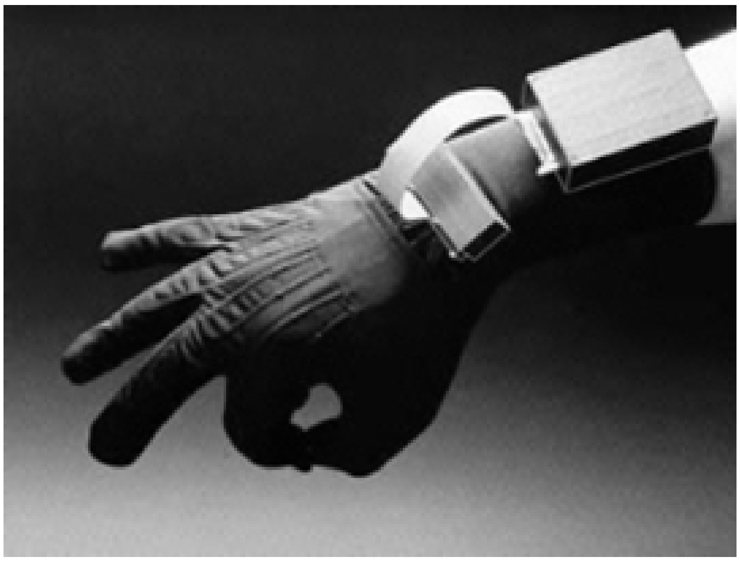


Figure 5.1: **Electronic Data Glove.** A glove similar to the one used to capture the sign language instances used in the *Results* part of this thesis. The glove provides dynamically data about the spatial position of the hand, the roll, pitch and yaw rotation angles of the wrist and the degree of bendiness of each finger.

Sign Language Recognition, SLR, can be carried out in isolated signs or continuous speech. Continuous SLR presents the advantage of being able to incorporate language models based on grammatical and semantical heuristics to improve performance. SLR can also be classified into recognition of signs from a single signer and signer independent recognition [AL-Rousan et al., 2009]. Several sign languages dialects have been employed in machine learning experiments: American [Starner et al., 1998], Australian [Rozado et al., 2010b; Kadous, 2002], Persian [Sarkaleh et al., 2010], Arabic [Mohandes et al., 2007], Italian [Infantino et al., 2005], Taiwanese [Liang and M., 1998], Polish [Kapuscinski and Wysocki, 2009] and Chinese [Fang and Gao, 2002]. Some authors have used electronic gloves and magnetic trackers conveying information about hand position in space and shape[Kadous, 2002; Dipietro et al., 2008] while others have opted for video based recognition with or without visual marking devices [Zhou et al., 2010; Liang and M., 1998; Starner et al., 1998; Holden et al., 2005; Gupta and Suwei, 2001; Lavee et al., 2009]. The computational methods traditionally used in SLR have been fuzzy logic [Holden et al., 1999], wavelet transform [Sarkaleh et al., 2010], neural networks [Vamplew, 1998; Fang and Gao, 2002], Hidden Markov Models, HMM, [Starner et al., 1998; Liang and M., 1998; Holden et al., 2005; Mitra and Acharya, 2007] and more recently HTMs [Kapuscinski and Wysocki, 2009].

5.4 HTMs in Gaze Gesture Recognition

The study of the interaction between humans and computers is known as Human Computer Interaction or HCI. The aim of HCI is to make interaction easier, more intuitive, and more efficient. Nowadays, interaction with computers is not limited to keywords and mouse. Different kinds of pointing devices such as touch sensitive surfaces, wide resolution displays, microphones and speakers are becoming common place devices for HCI purposes. Hence, new modalities for HCI have emerged: speech interaction, gesture based interaction or tangible sensor based interaction. An emerging input modality is gaze interaction which nowadays finds its main application in the field of accessibility and user studies. Accessibility systems typically use gaze as the sole input for HCI purposes. Outside the field of accessibility, eye gaze can be combined with other input modalities to form multimodal interaction paradigms.

In Chapters 8, 9 and 10 we explore the usage of traditional HTMs and extended HTMs on the recognition of gaze gestures, both offline and in real time. Gaze gestures are a new form of gaze interaction with a computer consisting on using a sequence of gaze movements recognized by a machine learning algorithm to generate a command. In order to fully understand the possibilities and limitations of gaze gestures a brief overview of eye tracking technology and gaze based interaction is provided next.

5.4.1 Eye Tracking

Eye trackers have been traditionally expensive partly because they require specialized hardware setups and customized software. Currently, a precise and robust eye tracking system costs usually between 10,000 and 50,000 €. Therefore, it is accessible only to a relatively small market segment. There is a great deal of interest in developing low cost eye trackers using off the shelf components [San Agustin et al., 2010]. However, trade-offs have to be reached between robustness, accuracy and price. A system can be developed which is cheap enough for most people to buy, but not precise or robust enough for all types of applications.

Eye tracking system must find a compromise between intrusiveness and precision. By placing different lenses or suction devices on the cornea, or by placing electrodes around the eyes, very precise readings of eye movements can be made. However, these systems are generally viewed as being too intrusive for everyday use. Video-based eye tracking systems are preferable for every-day use. Moreover, since many computer devices such as mobile phones, laptops and flat displays already contain built in cameras and processor power continues to increase steadily, video based eye tracking is technically feasible in most electronic devices at reasonable costs [San Agustin, 2010].

Video based eye tracking uses an infrared led pointed at the user's eye. The infrared light causes a reflection on the cornea known as glint. Due to the eye spherical shape, the glint remains at the same position over the surface independently of where the eye is directed. Using a video camera, image analysis software libraries detect the reflection spot, or glint, and measure its distance to the center of the pupil as the user moves its gaze around, see Figure 5.2. Gaze can be derived from the glint to pupil center distance through a 2 degree polynomial model generated during a previous calibration procedure [San Agustin, 2010].

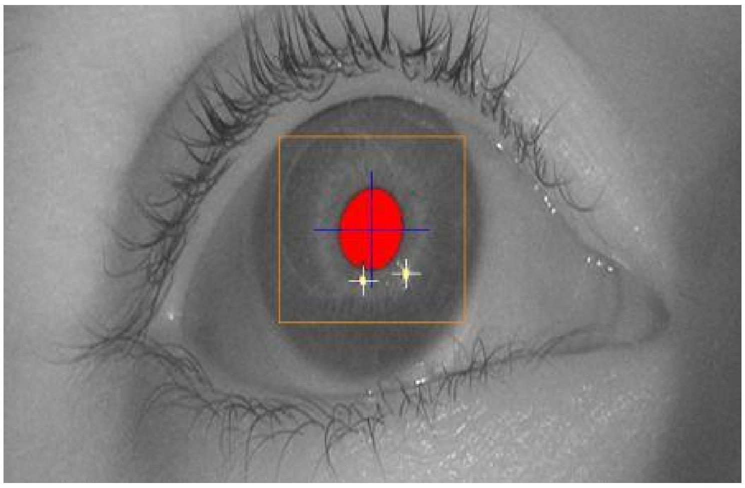


Figure 5.2: **Eye tracker.** An open source gaze tracker [San Agustin et al., 2010] tracking one eye. The features detected in the image are the pupil center and two corneal reflections.

Some people interact with a computer all day long, for their work and in their leisure time. Most interactions are done via keyboard and mouse. Both types are dependant on heavy use of the hands. As a result of this trend, some people suffer from repetitive strain injuries, or RSI, in particular parts of the hands, arms, elbows, shoulder, neck or back. With the vision of ubiquitous computing becoming a reality, the amount of interaction with computers will increase further and hence, there exists a pressing need for interaction techniques which do not cause physical problems. The eyes are good candidates for this because they move by default anyway when users interact with computers. Using the information of eye movements, a lot of hand interaction could be spared [Drewes, 2010]. Finding interface techniques that take advantage of the intuitiveness and speed of eye movements can offer users some real benefit in terms of HCI since by being able to track a person's eye movements, the focus of its attention can be deduced [San Agustin, 2010].

5.4.2 The Nature of the Eye

The eyes constitute the primary sensory organ of humans. The eye is a light sensitive organ, which absorbs light patterns and directs them to the brain via the visual pathway using nerve impulses.

Light is a form of electromagnetic radiation created from the oscillation of electrically charged particles. Electromagnetic radiation is abundant in our surroundings and this is probably why mammals have evolved sensors to register electromagnetic radiation in the environment. Visible light only constitutes a small part of the actual spectrum of electromagnetic energy (other forms being radio waves, infrared, ultra violet, microwaves, etc). However, physical light ability to interact with the surface of objects through the physical concepts of reflection and refraction allows for a great amount of detail to be conveyed [Duchowski, 2007].

We generally use the eyes to gather information about our surroundings. However, the eyes are also used to convey information. When we talk with other people, we normally address the person by looking at her or him. This small example show how we use the eyes beyond vision for communication purposes.

The fact that we use the eye for human to human communication leads to the following conclusions for the field of HCI:

- We have the ability to control our eyes and use them as output effector organs.
- For natural communication with computers, the computer should know user's gaze direction.
- Analyzing eye gaze reveals information on the person (mood, intention and life experience [Chua et al., 2005]). This information would be very useful for smart computer interfaces which are able to assist the user on a personal level.

Figure 5.3 shows a schematic view of the human eye. The eyeball is approximately a spherical object. Three important structures relevant for our discussion here are: the iris, the pupil and the sclera. The pupil and the iris are covered by the cornea, a transparent layer that refracts light before it enters the eye. The pupil is the opening through which light enters the eye. The pupil regulates the amount of light reaching the retina by expanding and contracting. The retina in the rear part of the eye contains photo receptors able to transform light impulses into electrical signals [Davson, 1990]. Most of the volume of the eye is occupied by the vitreous body, which consists of a transparent fluid through which light is filtered. The light entering the eye is refracted by a lens (a transparent structure located behind the pupil) that changes shape in order to focus the rays of light into the retina.

The process of receiving and transforming light waves into electrical signals that convey visual information takes place in the retina. Photoreceptors in the retina are neuron-like cells that convert light into electrical impulses that are then sent to the brain through the optic nerve. The optic nerve carries the impulses until they reach the optic chiasm [Davson, 1990]. At this point, impulses that come from the nasal area of the retina pass over to the opposite side of the cerebral hemisphere. From this point on, the optic nerve becomes known as the optic truck and relays information to a subdivision of each cerebral hemisphere called the lateral geniculate nucleus (LGN) [Davson, 1990]. From here, information moves through the visual cortex hierarchy and its various processing centers: V1, V2, V3, V4, MT, PIT, PP, etc.

Photoreceptors are divided into two types: rods and cones. Rods are very sensitive to light but can not discriminate colors. They are able to provide visual information in dim light conditions and they are mainly responsible for our peripheral vision. Cones on the other hand, can discriminate among different colors but

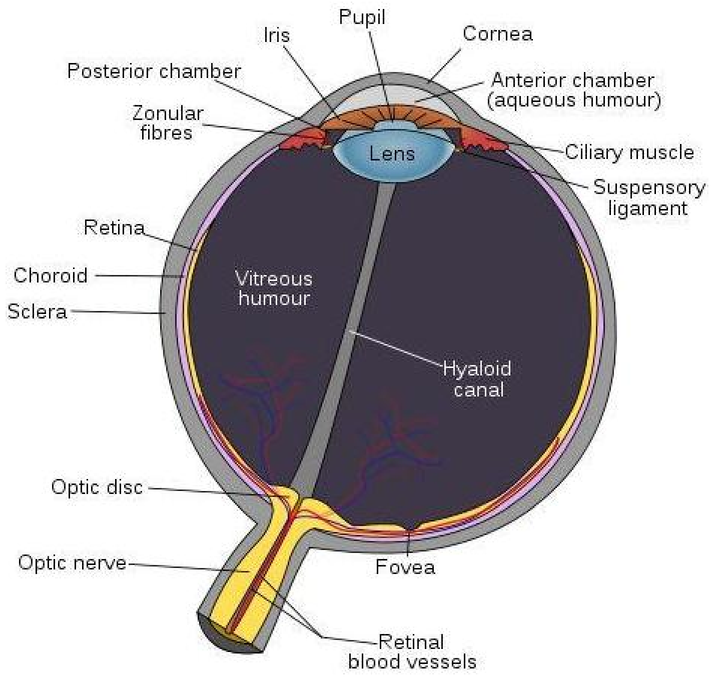


Figure 5.3: **Anatomy of the human eye.** Schematic view of the human eye showing its main anatomical parts. Light enters through the pupil and gets focused on the retina by the cornea and the lens.

are not as sensitive to light as rods. The distribution of rods and cones across the retina surface is irregular. The highest concentration of cones corresponds to the fovea. The fovea is located in the retina precisely at the center of the macula region. The fovea is responsible for sharp central vision, which is necessary for any activity where visual detail is of utmost importance. The angle of vision covered by the fovea is around 2^o. Visual acuity is approximately constant within those 2^o and drops linearly from there to the 5^o border of the fovea, and exponentially beyond 5^o [Davson, 1990]. The eye is not completely symmetric; however, an approximated symmetry axes can be considered as the line joining the different lenses of the eye. This is called the optical axes, and it does not coincide with the actual line of sight or visual axes. The visual axes is the line that connects the fovea with the object we're looking at. Optical and visual axes intersect at the lens center, with an angular offset that is subject dependent. There is a high variability among humans. Typical values are 5^o in the horizontal direction and 1.5^o in the vertical direction [Davson, 1990].

5.4.3 States of the Eye

Eye movements consist of rotations of the eye ball. These movements are possible by six muscles known as the extra ocular muscles. The concept of saccadic suppression refers to the fact that while our eyes are moving from one fixation point to another, the visual cortex is not receiving information. The states on which a human eye can be are three:

- **Fixations:** The temporal state in which the eye is relatively still and fixed on a feature of interest. They have a duration longer than 100ms. It is during fixations, that we gather data from our surroundings. The use of the concept "relatively still" comes from the fact that the eye is constantly moving. Although we perceive the image as being completely still, three different types of eye movements (micro-saccades) take place during fixations: drifts, tremors and micro-saccades. This micro movements are unconsciously and autonomously performed by the brain stem in order to keep the photo receptors in the retina constantly excited, and to bring the different parts of an object of interest to the center of the fovea. Experiments made with the eyes artificially held still while observing images show that objects appear to fade away or disappear entirely if the pupil is held still. This micro-saccades jerky movements of the eye scattered around 1° over the visual angle. Where fixations occur and how long they remain at the specific point is usually an unconscious decision, based on factors such as light conditions and objects of interest [Davson, 1990; San Agustin, 2010].

- **Saccades:** A saccade is a fast ballistic eye movement that prepositions the eye so that the object in which we are interested is projected on the fovea. Saccades usually occur between two fixations. No information is gathered by the visual system during a saccade. The velocity of a saccade is often in the order of 700^o/s for large amplitudes saccades. Saccades are reactions, forced or unforced, to light changes perceived in areas of the retina with lower resolution than that of the fovea [Davson, 1990; San Agustin, 2010];

- **Smooth Pursuit:** Smooth pursuits take place when our eyes track a moving object. Smooth pursuits require a moving target and the process of following the target requires a combination of both saccadic and small eye movements. The object in motion requires the eye to perform minor corrections during small saccadic movements to maintain the object in the fovea while it moves around. The vision system is able to track objects moving at a velocity in the range of 1 to 30^o/s. Smooth pursuit movements are not subject to endogenous control, and require a moving stimulus [Davson, 1990].

5.4.4 Eye Tracking Techniques

The movements of the eye can be tracked using different technologies, and the accuracy and invasiveness of each will depend on the method employed. Often, a trade-off needs to be made between the cost of the eye tracker, the precision of gaze estimation and the intrusiveness of the equipment. Eye tracking methods are commonly divided into three categories:

- **Electrooculography (EOG):** Based on the existence of an electric field that changes its potential as the eye moves in its orbit. To detect these changes in electric potential, electrodes are placed on the skin around the eyes. The system is regarded as invasive [San Agustin, 2010; Duchowski, 2007].

- **Special purpose contact lenses:** The most accurate method to track the users' eye gaze is achieved by means of special purpose contact lenses. The accuracy is as high as 10 arc-seconds. However, this method is extremely invasive with the lenses being connected to wires, making the system very uncomfortable to use and constrained to laboratory research [San Agustin, 2010; Duchowski, 2007].

- **Video oculography (VOG):** Uses a camera to record the eye movements of the user, and extracts the information of different eye features to determine the point of regard or the line of sight. The great advantage of video based eye trackers lies in their inclusiveness. The user does not need to wear any hardware, and generally, systems are tolerant to head movements, at least to a certain extent. Depending on the hardware configuration of the different components, it is possible to classify this tracking systems as remote or head mounted. In remote systems, the camera and light sources are placed at a distance from the user, usually below or around the computer screen, making for a non intrusive configuration. Head mounted systems places the camera and light sources on the head of the user, usually mounted on a helmet or a pair of glasses. Such systems make mobile gaze interaction possible [San Agustin, 2010; Duchowski, 2007].

5.4.5 Gaze Estimation

The main purpose of a gaze tracking system is to determine where a person is looking at by gathering information from eye position and movement. Several methods exists to do this that can be best categorize in two main categories [Hansen and Ji, 2010]:

- To determine the person's eye gaze as a 3D vector in space (the line of sight).

- To determine the person's eye gaze as a 2D point on a plane (the point of regard or PoR).

The process of determining gaze direction can be divided into two sub-processes: eye tracking and gaze estimation. Eye tracking consists of following the eye movements as it moves around in its orbit. This is often achieved by detecting and tracking eye features such as the pupil or the iris overtime. Gaze estimation uses these features to estimate the users' eye gaze. Eye tracking methods can be divided into three categories:

1. Shape base methods rely on a prior model of the eye structures, which can be constructed from features of the eye or from contours (deformable shape models). The pupil, iris and eye corners are commonly used features. The image data are fitted to this predefined eye model [Hansen and Ji, 2010; San Agustin, 2010].

2. Appearance based methods rely on building a model of the eye region that tries to imitate as much similarity in the template to the target feature [Hansen and Hammoud, 2005]. Usually they are built from a large set of training images with different subjects, illumination conditions that try to cover all the classical situations that depict the eye features in any given image [Hansen and Ji, 2010; San Agustin, 2010].

3. Hybrid methods combined different techniques, such as shape, appearance and model, to exploit the benefits of each method [Hansen and Ji, 2010; San Agustin, 2010].

The method employed to track the eye depends on the type of objective wanted to be achieved. Most systems make use of active infrared light, although passive systems that use natural light exist. Methods that rely on infrared illumination are the most popular in both commercial systems and in research. The pupil becomes a salient feature when infrared light is available. Outdoor eye tracking is challenging, and efforts are devoted to improve the reliability of eye tracking under varying light conditions [Hansen and Ji, 2010; San Agustin, 2010].

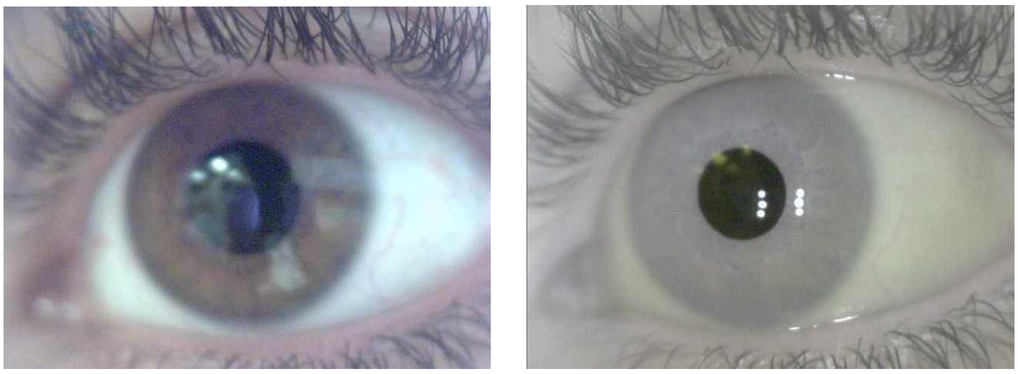
Depending on the use of the gaze tracking system, two different applications can be considered: diagnosis and interaction. Diagnostic applications analyze eye movements to gather information on the person's cognitive processes and attention when performing specific tasks such as inspecting a visual scene, listening or reading. These analyses are usually carried out not in real time, and therefore the eye movements do not have an effect on the scene being observed. Diagnostic applications are widely employed in different fields, such as usability research, marketing and psychology [Duchowski, 2007].

Gaze tracking can also be used to facilitate interaction with computers. Gaze information provided by the gaze tracker can be used as an input to control a GUI on a screen. Often the PoR coordinates estimated by the system are usually noisy, mainly due to inaccuracies in the extraction of the eye features. When the PoR is used as a pointer substituting the mouse the jitter in the pointer can be distracting for the user, so some degree of smoothing is usually applied to the signal to make the cursor more stable [San Agustin, 2010].

Gaze tracking systems use infrared light to improve the pupil/iris contrast, see Figure 5.4 and to create the glint used as a reference to estimate gaze. Infrared illumination can be used in two different modes. A light source located close to the optical axes of the camera is called on axes light. The captured image shows a bright pupil since most of the light falling on the retina reflects back to the camera [Hansen and Hammoud, 2005]. When a light source is located away from the optical axes of the camera, which is called off axes, the retrieved image shows a dark pupil [Hansen and Ji, 2010].

5.4.6 Eye Movement Detection

Analyses of eye movements while users perform different tasks (reading or driving) allows the extraction of information regarding the visual and cognitive processes



(a) (b)

Figure 5.4: **Improved contrast with usage of infrared illumination**. Panel *a* shows an image capture of an eye without infrared illumination. Notice the difficulty for the image recognition routines to precisely identify and delimit the edge between pupil and iris. Panel *b* shows the same eye with infrared illumination. Increased contrast allows the image recognition routines to easily segment the pupil from the iris.

taking place. Different metrics are employed in the analyses: fixation duration, saccades amplitude, saccades velocity, etc. This analysis is usually carried out *a posteriori* [Duchowski, 2007].

Fixation detection algorithms have been divided into three main categories: velocity based, dispersion based and area based. Each method presents its own set of limitations [San Agustin, 2010].

Smooth pursuit detection has not been fully explored in real time gaze control applications. Although most interaction in graphical user interfaces is based on point and click, applications that display moving objects might benefit from a smooth pursuit detection algorithm [San Agustin et al., 2010].

A task that has been substantially explored using eye tracking is reading patterns. Generally, fixations occur at every word although small function words are often skipped and saccades size depends on the specific nature of the task at hand. In tennis, it has been shown that advanced players loom fixations and saccades together differently from novices. These examples further substantiate the premise that cognition affects the way scenes are viewed or vice versa [Chua et al., 2005].

5.4.7 Gaze Interaction

HCI deals mainly with moving information between the computer and the brain of a user. How to increase the useful bandwidth across that interface, making it more natural and convenient is one of the main goals of HCI.

Gaze interaction deals with using the direction of gaze to navigate and/or make selections in information interfaces. This interaction can be considered in the context of an addition to existing input modalities in HCI or in the context of a sole input. The reason for this distinction is that the design parameters and applicability of these two input strategies are very different. Gaze as an addition mainly deals with the enhancement of navigation processes. Gaze as a sole input needs to be able to handle navigation and selection processes [Mollenbach, 2010].

Current day selection strategies use for sole input can be divided into two main approaches; dwell time activation and gaze gestures, both of which seek to minimize accidental selections [Mollenbach, 2010]. Dwell time activation consists of prolonged fixations on onscreen targets. Gaze gestures consist of recognizable and repeatable sequences of eye movement patterns that can be distinguished from natural search patterns.

There are many situations where conventional input devices are inadequate or inappropriate and alternative input devices are necessary. When researching interaction techniques several parameters affect each other and should be taken into consideration when determining the most appropriate input. The three basic parameters to be considered are: task, user context, and feedback.

Physical impairment constitutes a context in which the choice of input device can be a life changing necessity. In other cases, the user might be limited in its movement possibilities to interact with a system (a doctor performing surgery or a soldier managing weapons) [Drewes, 2010].

As previously mentioned this interaction has some inherent issues at its core, the fundamental one being the need to make a system aware of the intentions behind gaze. In gaze controlled interfaces, additional information is needed to inform the system whether eye movements are an indication of us wanting to interact with something or simply examining it.

Users' gaze can be used as an input to control a computer by using the PoR provided by a gaze tracker. As such, the users' gaze becomes a pointing device that substitutes or complements the mouse. Eye movements can reveal our intention to interact with an object in the physical world. Interactive applications can use eye movement as an input to control an interface. In these applications, the user looks around to select items and activate objects, and the interface reacts in real time by modifying the information displayed. However, it must be noted that even though our eyes can show our attention to interact with real world objects, these objects to not react to gaze.

The use of eye movements as a direct input to control an interface has been extensively studied. The main issues when using exclusively eye movements for interactions is the Midas touch problem. Our eyes can be used for pointing after some training but they lack an activation mechanism, a necessity in the case of controlling an interface. Hence, there is the risk of issuing an unintended activation. Ideally, an external switch would allow the user to perform selections, in the same way as the bottom of a mouse. However, some individuals might be unable to use an external switch, and in these cases gaze is the only selection technique available. Dwell time clicking or long blinks are the most commonly used solutions to circumvent this problem. The former requires the user to stare at the desire interactive object for a predefined amount of time in order to ease an activation, while the later requires a long blink to generate a selection.

Different studies have shown the speed advantage of gaze over manual input devices. This pointing can be faster than mouse pointing for very short dwell times. However, short dwell times are unrealistic since they produced a high number of undesired selections in everyday interaction. Provided good tracking conditions and some training, the eyes can be used for pointing, although not as efficiently as with the mouse. Nonetheless, an eye gaze interface might offer several potential benefits [Drewes, 2010]:

- **Ease of use:** Less stress for hands and arms.
- **Interaction speed up:** The combination of eyes with another input modality can provide even faster interaction.
- **Maintenance free:** Gaze based interaction works contact free which means that no maintenance is necessary.
- **Hygienic interface:** Eye gaze interfaces allow interaction without the need to touch.
- **Remote control:** High resolution cameras make eye gaze detection possible over several meters of distance.
- **Safer interaction:** Eye-tracking not only warranties the presence of a person but also his or her attention.
- **Gain more information on the user's activity and whereabouts:** Tracking the eye provides useful information for context aware systems. In the simplest form an eye tracker tells where the attention is focused. [Drewes, 2010].

Of course eye interaction also possesses drawbacks [Drewes, 2010]:

- **Ability to control:** The eyes perform constantly unconscious movements and this might disturb their use as computer input.
- **Conflict of input and vision:** Using the eye for computer input might result in conflicts since the eye primary function is to provide vision.
- **Fatigue of the eye muscles:** It is conceivable that extensive use of muscles groups involved in eye movement could cause physical problems such as repetitive strain injury.

5.4.8 Gaze Gestures

We explore in this thesis how our extended HTM paradigm can successfully solve the problem of real time gaze gesture recognition. Hence, we provide a brief introduction to this innovative HCI paradigm.

The idea of using gaze gestures for HCI purposes stems from a pressing need to find novel methods to interact with computers through gaze beyond dwell times

[Drewes, 2010]. Numerous electronic devices nowadays possess a built in camera within them. Furthermore, the processing power of the standard computer makes processing of real time video media stream feasible. It is hence possible that in the near future eye tracking technology could be built in most computing devices for no extra cost. Therefore, gaze interaction could become an additional input modality for the mainstream user [Drewes, 2010].

Gestures can consist of any repeatable and recognizable body motion that can be robustly recognized as separated from normal physical action. Gestures have traditionally been explored in the field of HCI. Unistroke [Mankoff and Abowd, 1998] for instance consists of gestures formed by sequences of elements, typically strokes, which are performed in a sequential time manner [Drewes and Schmidt, 2007]. The amount of gestures commands available in gesture interaction can can be arbitrarily chosen. In GUI command based interaction, commands are usually selected from a list [Drewes, 2010]. Large commands lists sizes results in large lists that cause real state problems in the screen. For these reason, gestures have recently started to be used as a way of interaction with electronic devices.

Stylus based text entry is an area where gestures have been greatly explored. Unsurprisingly, stylus based text entry has served as an inspiration for gaze gesture based text entry. More recently, gestures have become a mainstream interaction principle with the introduction of multi-touch finger gestures for mobile devices, laptops and pads. Some of the most common finger gestures are: taping, sweeping and pinching [Mollenbach, 2010].

Various types of physical gestures have also been explored when considering users with motor skill impairments because they allow for individually adapted interactions. For example, head gestures consisting of simple horizontal and vertical motions have been used as switch control gestures.

Gaze gestures are a recent addition to the set of gaze interaction modes. A gaze gesture can be described as: “a definable path of eye movements performed within a limited time period, which may or may not be constrained to a particular range area, which can be identified in real time, and used to signal a particular command or intent” [Drewes, 2010; Mollenbach, 2010].

Strokes are the foundations of gaze gestures. A stroke is the motion between two intended fixations; it is not necessarily the same as a saccade, which is the eye movement between any two fixations. A stroke can be completed even if jitter causing small saccades did take place between the two intended fixation points [Mollenbach et al., 2009]. Potentially, gaze gestures hold many advantages as an interaction method:

1. **Speed:** Gaze gestures can potentially be very fast. Saccades reach over 1^o and 40^o of visual angle and last between 30 - 120ms.

2. **Screen real state:** As shown in [Rozado et al., 2011b] gaze gestures do not need to take up screen space.

3. **No Midas touch problem:** If gaze gestures can be easily distinguishable from navigational eye movements, the Midas touch problem pervading dwell

interaction can be avoided.

Gaze gestures have nonetheless their own set of innate problems. The main one being accidental gesture completion, due to the potential overlap between natural gaze patterns and the eye movement patterns needed to complete a given gaze gesture. Accidental gesture completion is the equivalent of the Midas touch problem for dwell activation. An additional drawback of gaze gestures is the cognitive load that they might induce on the user. In standard HCI, activation is generally a direct response to information presented on the screen, whereas gaze gestures often rely on the user to be able to remember specific eye movement patterns under consideration with no immediate feedback.

5.4.9 Eye Gaze Based Interaction - Limitations

Gaze based interaction is mainly used in the domain of accessibility and user studies [Drewes, 2010]. Eye tracking based accessibility systems often used a keyboard on the display through which users enter characters by gazing for a predefined threshold time (dwell time), on the target key. The time saved by the fast nature of eye movements is lost through the dwell time. Reducing the dwell time threshold however causes unwanted selections. Furthermore, gaze interaction is limited by the accuracy limits of eye tracking technology, that coincidentally hast not much to do with hardware limitations but rather with aspects intrinsic to the physiology and anatomy of the eye. As a result, gaze interaction requires big size layouts which leads to space issues [Mollenbach et al., 2010].

Most of the limitations of gaze based interaction can be traced back to the nature of the eye as mainly an input sensor and not an output actor [Drewes, 2010]. Usually, the eyes move to see something, not to trigger actions. Using the eyes for both, input and output, often results in conflicts as a results of overloading of the visual channel [Mollenbach et al., 2009; Drewes, 2010]. The question of how much intentional output activity the eyes are able to handle is not clear yet.

5.4.10 Stroke Complexity

Strokes are the foundations of many types of gaze gestures. A stroke is defined as the motion between two intended fixations. The reason why fixation points are interesting for gaze gestures lies on the fact that they ensure a distinction between inspection eye movements and selection eye movements.

A single stroke gesture is the simplest form of finite gestures and is defined as the motion between two intended fixation points. A complex gaze gesture is the motion between three or more intended fixation points. Intermediate complexity gaze gestures are the main focus of the experimental research conducted and subsequently presented in this thesis.

Complex gaze gestures have the advantage of increasing the vocabulary size of gaze interaction and minimize the likelihood of accidental gesture completion. However, complex gaze gestures bring with them both cognitive and physiological loads on the end user. Cognitively it is difficult to remember a large number of gestures and physiologically it is stressful to create and complete them. There is a need for determining how demanding it is to complete a number of strokes that can be both remembered and repeated in a sustainable manner.

5.4.11 Finite Gaze Gestures

Finite gestures are eye movement patterns in which a predefined shape is completed. These can be based on abstract shapes in which the eye movement follows a discernible shape, but one with no innate meaning. It could be a triangle, a square or a circle. Symbolic path gestures also follow a specific shape; however these stand for or suggest something else by reason of the relationship, association, convention or accidental resemblance. An obvious example is that of gestures that resemble a letter in an alphabet.

One of the main issues concerning the implementation of a gaze based interface is the fact that simple gestures are easy to repeat and there are only a limited number of them available. Complex gestures on the other hand are plentiful and can construct a huge alphabet. However, they require more precision when being completed and increase the cognitive load on the user.

Several studies have shown the limited potential of gaze gestures on text input using a mapping from a specific gesture to a letter of the alphabet [Wobbrock et al., 2008]. The main advantage of this principle is that a large number of these gestures could become easier to remember because they have become contextualized and build on the familiar representations of the alphabet. The disadvantage is the high cognitive load they required, which can overwhelm many users.

Path based gestures have also been explored on several locations [Heikkilä, 2009]. Studies have shown the potential of gaze gestures to be used in a drawing program [Heikkilä, 2011]. This represented one of the first examinations of how gaze behaves in situations where the user follows unintended shape path with the eyes. The overall conclusion of the research was that there was not a significant difference between the time it took to complete the path of large shapes compared to the time it took to complete the path of smaller shapes. This was found to be the case because the eye movements of participants would reach higher speeds when completing the contours of larger shapes. Overall, the study found that it was very difficult for users to follow the paths properly, specially curved ones.

In another example of abstract gaze based gestures [Drewes, 2010], the gestures were relative to the eye movements rather than on screen fixed points. A set of these gestures was developed based and a combination of an existing set of mouse gestures available for the Firefox browser [De Luca et al., 2007]. A short dwell time was implemented to help differentiate between eye movements. The researchers asked participants to complete the gestures on different backgrounds with text, tablets or web pages. Their results hinted that both gaze gesture complexity and the background over which they were performed did not have a major impact on completion time [Drewes, 2010]. Gesture completion time only depended on the number of segments, in other words the number of strokes that needed to be completed. They recorded a time of only between 550-600ms per stroke.

In [Vickers et al., 2008], authors proposed a novel approach to gaze interaction named "Snap clutch". This method was implemented as a model interface that allowed the user to control an avatar in the computer virtual scenario of Second Life. They initially tried to use gaze to substitute mouse control. Snap clutch was implemented so that the user could move between different modes by looking off screen in different directions. The results showed that selection times almost reached that of the mouse and had fewer errors compared to dwell time.

Inspired by this work, the same authors have analyzed the usage of abstract paths base gaze gestures to control an avatar in the virtual game of "World of Warcraft" [Istance et al., 2009]. The results of this work show surprisingly small differences between the input modalities of Mouse/keyboard and gaze respectively. Their main motivation for their work was to allow users with severe motor impairments to be able to participate in online communities on equal footing with other players. They found out that all participants involved in the study could use gaze gestures to navigate or perform actions in the game. However, locomotion proved quite difficult during attacks, but accidentally completed gestures were relatively low.

5.4.12 Continuous Gaze Gestures

Continues gestures are not based on a specific shape, rather they are derived through our response to either a dynamic or a static visualization. A lot of research on gaze gestures deals with the strokes being tight to some kind of visualization on the screen. These gaze gestures are defined and shaped by the features laying out the information, dynamic or otherwise, on the screen. This type of gestures will be referred to as continuous gaze gestures. Path based gestures are often finite patterns, whereas continues gestures rely on continuously performed movements.

The main example of software using continuous gaze gestures is dasher which is also one of the first gaze gesture interfaces [Mackay, 2003]. Dasher uses motion as feedback to indicate which letter is being selected. Initially 27 characters are placed in a column to the right of the screen. To select the letter T, the user looks at the letter T in the right column. The size of the letter begins to increase as it moves toward the left. Once the letter crosses the line dividing the screen, it is selected. The user can reverse and then select a chosen letter at any time by looking at the left side of the center line. Other than being a continuous magnification base gaze contingent typing application, Dasher is known for having a very well integrated letter probability prediction model [Mollenbach, 2010]. Probabilities are dynamically visualized by increasing the size of the most probable subsequent letter in a sequence. If *"T"* is the first selected letter, then the letter *"H"* is presented as the largest one of the whole set of letters. After that, *"E"* becomes the most prominent letter and so forth, see Figure 5.5.

Continuous gaze gestures use eye movements to guide users through the interface by providing feedback that corresponds to consequences of a given action. The disadvantage of these techniques is that they are very task and application specific.

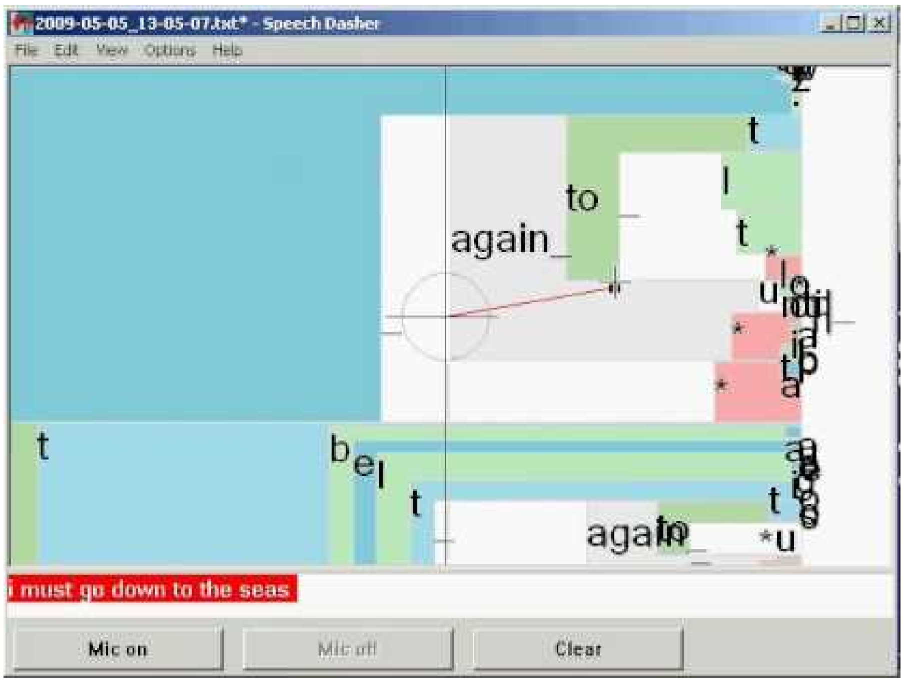

2009-05-05_13-05-07.txt* - Speech Dasher
File Edit View Options Help
again_
to
i must go down to the seas
Mic on
Mic off
Clear

Figure 5.5: **Gaze typing with Dasher.** Letter selection is done by gazing at the incoming letters from the right. Letter size is correlated to its likelihood to follow the text written so far.

5.4.13 Final Thoughts on Gaze Gestures

Traditionally gaze tracking technology has used user's gaze as a pointing device. Little research has focused on other types of gaze interactions such as gaze gestures.

Gaze gestures hold considerable promise as an innovative input modality in HCI. Gaze gestures address some of the problems of eye gaze interaction [Drewes, 2010]. Some gaze gestures modalities use only relative eye movements and consequently do not require calibration[Drewes, 2010]. Accuracy is in general not a big issue for gaze gestures, because gaze is not used for pointing [Drewes and Schmidt, 2007]. Hence, low cost eye trackers can be employed to detect gaze gestures. Also, the usage of gaze gestures circumvents the Midas touch problem. Finally, gaze gestures do not need to take real state on the screen freeing up space for the display of information.

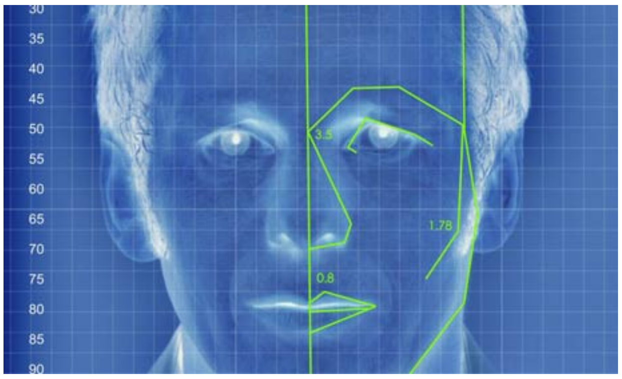
Research on gaze gestures is still at its infancy. It remains to be seen whether users will accept gaze gestures as a new input modality to control an interface. In the field of accessibility however, gaze gestures offer obvious advantages to users with disabilities since they provide an additional interaction possibility to a rather interaction constrained population. Gaze gestures can also be advantageous in highly hygienic environments, such as operating rooms, were surgeons or nurses may wish to interact with computers using gaze alone. Possibilities in the realm of mobile devices also hold considerable promise since traditional input methods are inconvenient due to the small display size of the devices.

Part II

Results

Chapter 6

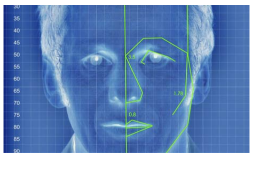
HTMs in Image Recognition



6.1 Introduction

In this section, we outline our own implementation of the traditional HTM algorithms and test it on a simple 2-dimensional binary image recognition task. This implementation served the purpose of my experimental setup on which to explore HTM theory[1]. The results shown in this section will illustrate some of the features of the HTM paradigm such as the emergence of invariance capabilities and its tolerance to noise. Not all principles of HTM theory have been implemented. Ideas such as using probability distribution over a set of beliefs to pass information up the hierarchy or the usage of prior probabilities by means of feedback from higher nodes to lower nodes as described in Chapter 4 have not yet been implemented. Nonetheless, our in-house version was intended as a proof of concept and serves the purpose of showing HTM robustness, noise tolerance, generalization and invariance capabilities.

[1]A freele available version of my software is available at `http://www.ii.uam.es/~gnb/drfthesis/htmimagerecognition.rar`)



6.2 HTM Algorithm - Methodology

6.2.1 Task, Training Data and Network Structure

HTM algorithms can theoretically be applied to a wide array of problems. Yet this chapter focuses on the usage of an HTM system on a visual pattern recognition task. The goal of the system is to be able to create invariant representations of a training set consisting of the 48 symbols shown in Figure 6.1. The system was trained on 4 different representative sizes for each symbol. The symbols were projected and moved during training over an artificial retina of size 32 by 32 pixels. Afterwards, the system was able to perform inference on unseen symbols similar to the training symbols but subjected to variable amounts of translation, scaling, distortion and noise.

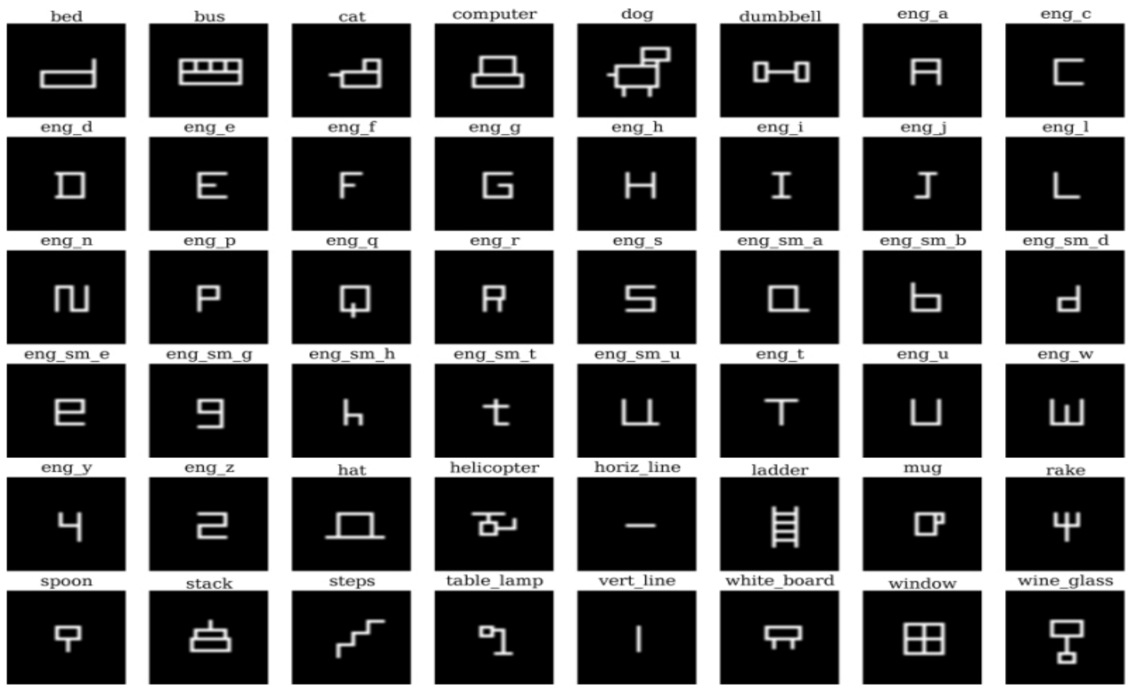
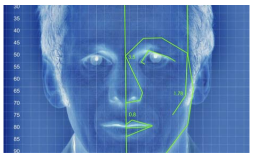


Figure 6.1: **Images training set.** Training images used by our HTM visual pattern recognition system. Each image was shown to the system over 4 different sizes going from small to large.

The training data consisted of a series of "movies" created by taking different scale variations of the 48 symbol set and moving then horizontally and vertically across the whole receptive field of the system. These movies determined and constrained the kind of transformations the system was invariant to.

As Figure 6.2 illustrates, we used an HTM network consisting of a 3 layers layout. The bottom layer (layer 1) consisted of 64 nodes organized in an 8x8 matrix, layer 2 consisted of 16 nodes organized in a 4x4 matrix, finally layer 3 consisted of just 1 top node. The nodes located on layer 1 receive input from the artificial



retina. In Figure 6.2 an “A” has been projected on the artificial retina with black pixels indicating “light on” and white pixels indicating “light off” over a particular receptive field. The different span and mappings of the receptive fields of the different nodes along the hierarchy have been marked in the figure.

HTM technology is not constraint to specific fan-in/fan-out parameters (the specific connectivity pattern from nodes in one layer to higher level nodes), we used the particular configuration of 4 nodes in layer 1 projecting to a node in layer 2, and the whole 16 nodes of layer 2 projecting to the single node in layer 3.

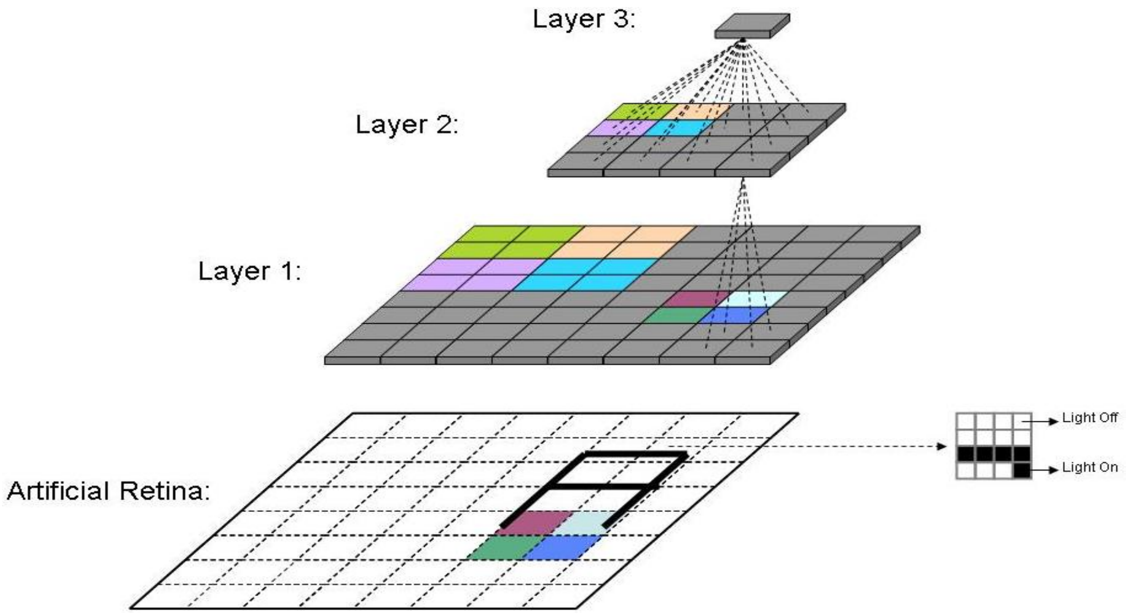
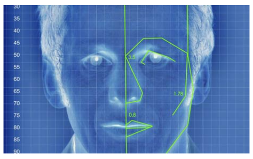


Figure 6.2: **HTM topology.** Architecture of the HTM Topology used for our image recognition task.

The receptive fields of nodes at different layers of the hierarchy span distinct areas of the artificial retina depending on where in the hierarchy they are located. Nodes at the bottom of the hierarchy span relatively small receptive fields directly over the artificial retina. Receptive fields of nodes higher up in the hierarchy span indirectly wider areas as can be seen in Figure 6.3. A node in layer 1 possesses a receptive field of 4x4 in the artificial retina. A node in layer 2 possesses a receptive field spanning the output of 4 nodes in layer 1. Finally, the only node in layer 3 receives as input data the outputs of all nodes in layer 2.

6.2.2 Node Operation

The nodes in an HTM hierarchy are the main computational elements of the network. Each node receives as input a concatenation of the output vectors from the nodes immediately underneath. The function of each node is to create invariant representations of vectors presented over its corresponding receptive field. That is, clustering together different inputs that are slight temporal or spatial variations of



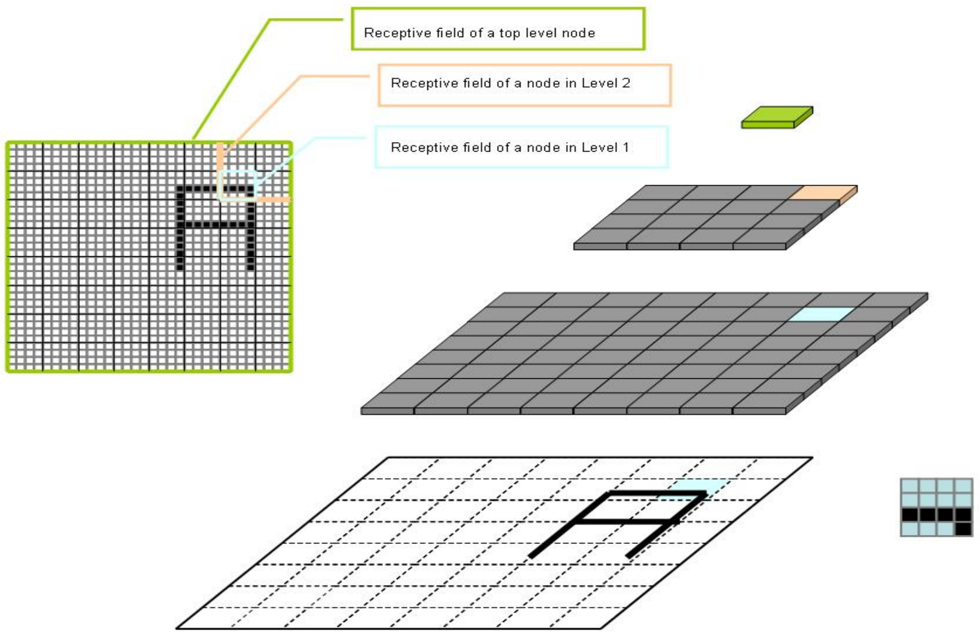
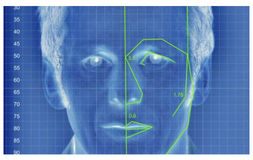

Receptive field of a top level node
Receptive field of a node in Level 2
Receptive field of a node in Level 1

Figure 6.3: **Receptive fields.** The span of node's receptive fields in an HTM hierarchy depends on their spatial position within the topological hierarchy.

each other. Each node goes through two stages during its lifetime: a learning stage, after which, the node switches to an inference stage.

Each node carries out two types of pooling mechanisms: spatial pooling and temporal pooling, see Figure 6.4. The spatial pooler pools together patterns according to their dot product similarities. The node stores in memory only spatial patterns that it has not observed previously and that are sufficiently dissimilar to stored spatial patterns. During training, when a new input vector pattern arrives to a node, the node performs a dot product between the new vector and each one of the already stored vectors. If any of these dot products is high enough[2], the new vector is discarded, since this means that a sufficiently similar vector has already been stored by the node. We refer to this vector which represents a bunch of similar spatial patterns as a "quantization point". If the node does not have in its memory a pattern similar enough to the incoming pattern, it stores the incoming pattern as a new quantization point. Spatial pooling works well in noise removal from patterns, but it is not useful in solving temporal invariances. That is, spatial pooling is unable to detect the similarity of two spatial patterns representing a horizontal bar moving up or down from time t to $t + 1$ since their pixel-similarity metric is very low. Yet

[2] "High enough" is a parameter set-up before training by the supervisor of the system. This parameter defines a trade-off: The bigger the parameter is, the more specific the system becomes, since more quantization points are stored by each node, yet the higher the computational requirements in terms of memory and processing power and the less generalization capabilities of the system



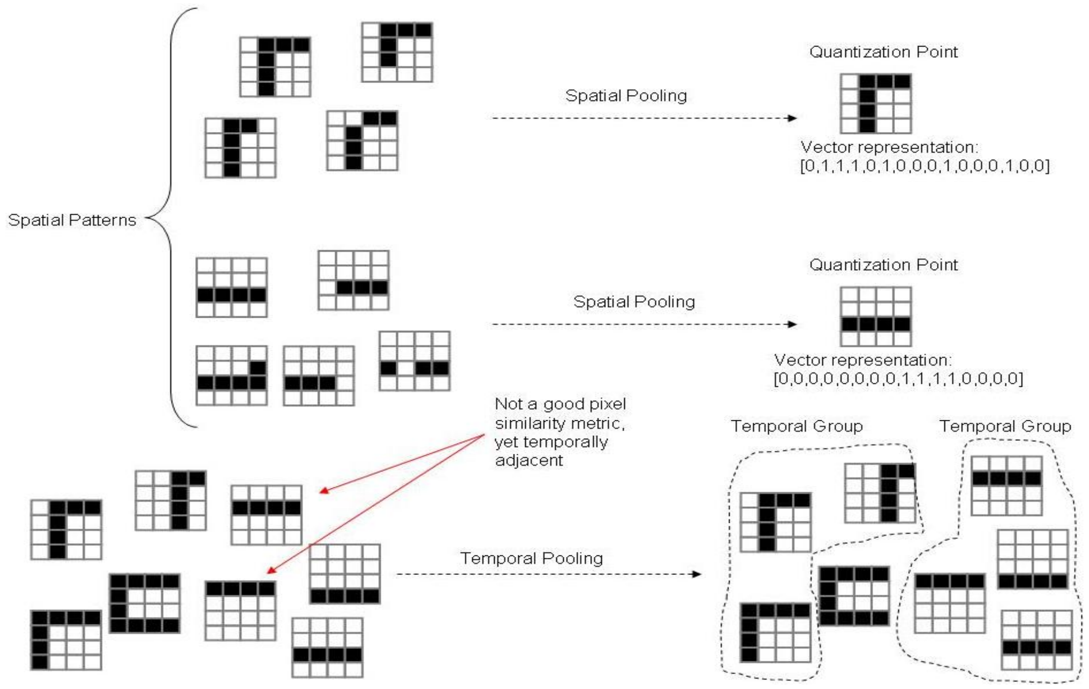


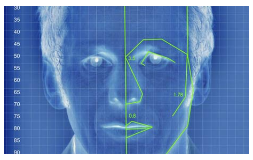
Figure 6.4: **Spatio-temporal pooling.** Each node carries out during learning two operations: spatial pooling and temporal pooling.

both spatial patterns represent a horizontal bar. We need a system therefore that would cluster both spatial patterns into a group which represents the ultimate cause that generates this spatial patterns (a horizontal bar).

Each node uses a second pooling mechanism according to temporal adjacency. This pooling mechanism uses temporal proximity as distance metric. If a certain pattern $sp4$ follows another pattern $sp12$ often enough, the two are cluster into an invariant representation formed by several spatial patterns with significant temporal proximity but not necessarily dot-product similarity. This system is able to pool together patterns that do not possess a high pixel-to-pixel similarity, but that nonetheless are caused by the same cause as it varies through time over the receptive field.

Summarizing, a node can be conceptually thought of as a computational box with two separate "boxes", a spatial pooler that stores "quantization points" representing the incoming input spatial space and a temporal pooler that pools together "quantization points" that are temporally adjacent into groups, Figure 6.5. The node generates as output a vector that indicates what temporal group is most likely being sensed at a particular instant in time.

Nodes throughout the hierarchy go through a learning mode before they switch to an inference mode. When all the nodes at a particular layer in the hierarchy have completed their learning process they are all switched to inference mode, and nodes in the next upper layer start their learning phase. The learning process of an HTM node can be visualized in Figure 6.6.



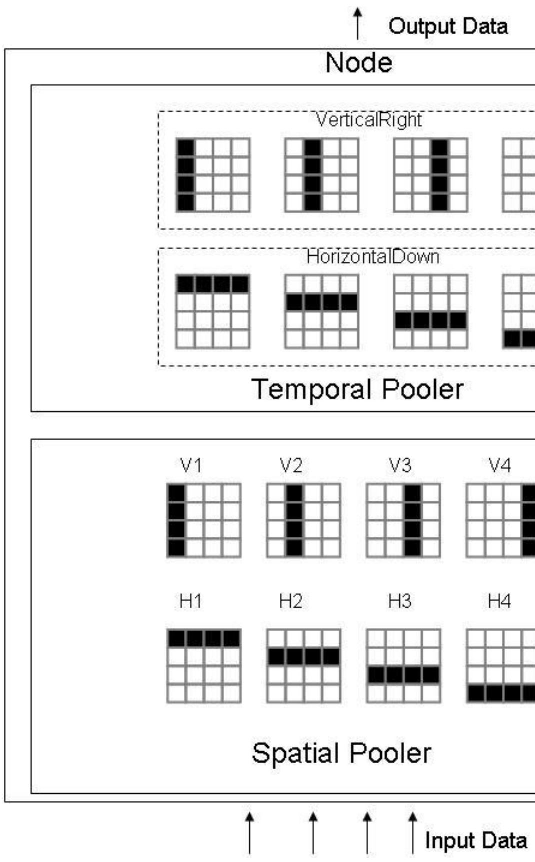


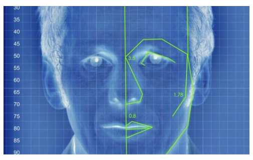
Figure 6.5: **An HTM node.** Conceptual representation of a Node in an HTM system.

Operation of the Spatial Pooler

During training, a node is exposed to different input patterns. The node stores only patterns which are novel enough. That is, when new patterns are compared to previously stored patterns using a dot product operation, only patterns with a sufficiently low dot product threshold parameter are considered different and stored. Figure 6.7 shows some of the "quantization points" stored by a node in layer 1 of our HTM system with a 4x4 receptive field over the artificial retina. Notice these spatial patterns consist mostly of vertical and horizontal segments, corners and intersections.

The node keeps learning patterns of "quantization points" until it fills its spatial memory. The process unfolds over time in a logarithmic fashion with a rapidly increasing number of "quantization points" being stored by the spatial pooler early on during learning, followed by a logarithmic-like pattern of growth later on. Figure 6.8.

Once the learning stage for a level of an HTM system has been completed, all the nodes in that layer are switched to inference mode. In this mode, a node compares the incoming input patterns to all the quantization points stored in its spatial pooler



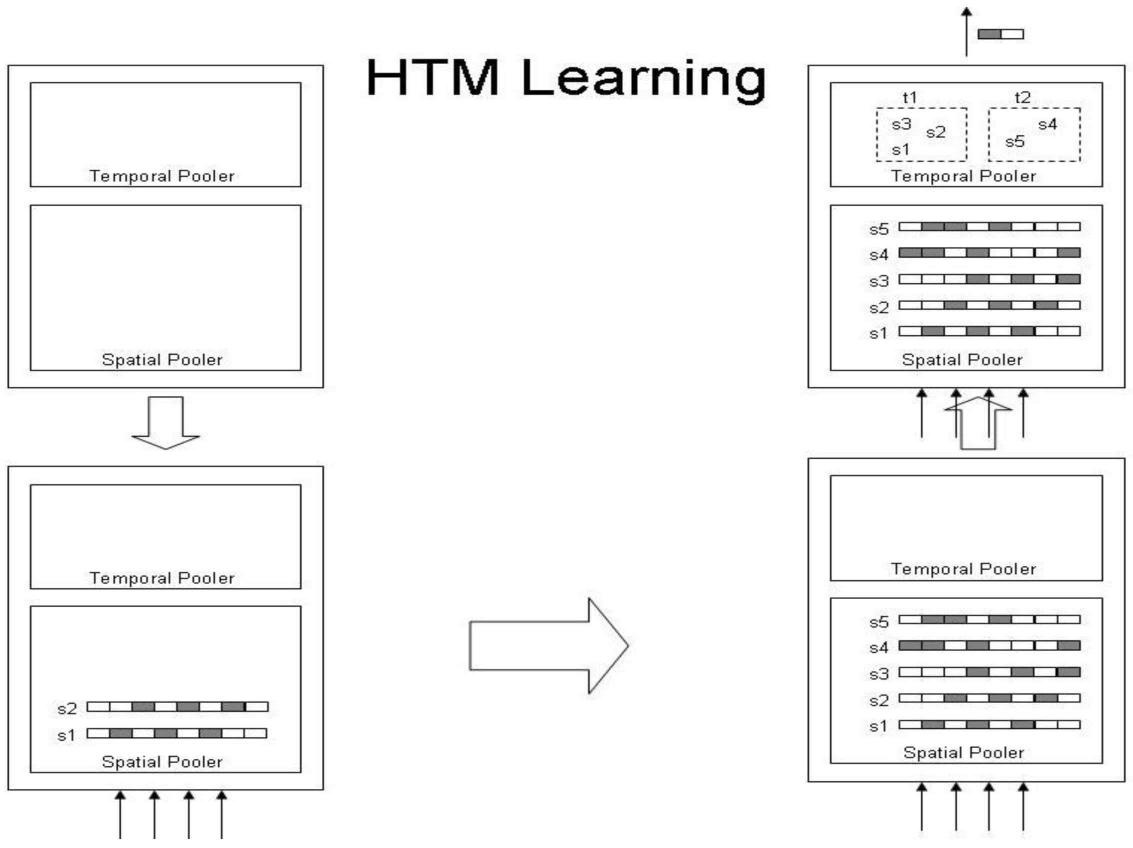


Figure 6.6: **Schematic representation of HTM learning.** The spatial pooler stores input vectors that are dissimilar enough. A node in layer 1 of the hierarchy receives binary input vectors of the type [0,0,1,0,1,0,1,0].

and determines which one of the stored patterns resembles the input pattern the closest. This approach is a shortcut of the theoretical guidance, which requires that at this stage the node produces a distribution of likelihoods in terms of how close the input pattern is to each of the stored quantization points in the spatial pooler. We have chosen a winner-take all simplification of this theoretical requirement. Once this operation has been completed, the spatial pooler then sends the current active "quantization point" to the temporal pooler.

Operation of the Temporal Pooler

The operation of the temporal pooler is partitioned in three stages. First, the temporal pooler learns the statistics of temporal transitions between the spatial patterns (quantization points) stored by the spatial pooler [George and Jarosy, 2007]. Then, using the frequencies learned in the previous step, the temporal pooler groups quantization points according to their temporal adjacency [George and Jarosy, 2007]. The temporal pooler produces then the output of the node by means of generating an output vector where all components are "0", except one component, which has the value "1". Each element of the output vector represents one of the groups of

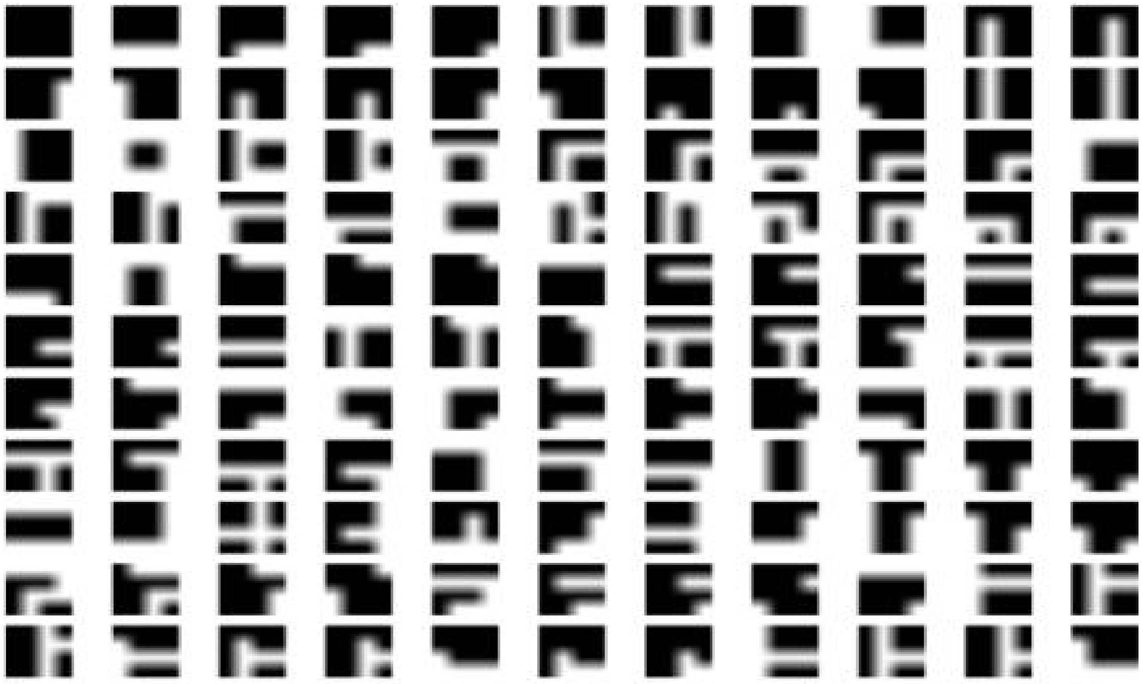


Figure 6.7: **Spatial pooler quantization points.** Quantization points stored by a node in layer 1 of the HTM image recognition system. Patterns correspond to parts of images over a 4x4 receptive field.

quantization points clustered together by the temporal pooler. In a winner take all fashion, "1", represents the temporal group of quantization points with high temporal proximity among themselves that the node believes its is being currently exposed to. It is important to understand that each component of the output vector does not represent a quantization point, rather a cluster of quantization points which have been determined to lie within temporal proximity of each other.

Learning the temporal transitions of quantization points proceeds by creating a zero matrix whose rows and columns represent all the quantization points stored by the spatial pooler. Through training, each time a node was perceiving a quantization point $a1$ at time t and a quantization point $b3$ at time $t+1$, a unit was added to the intersection between the row representing quantization point $a1$ and the column representing quantization point $b3$. Figure 6.9 shows how the time adjacency matrix looks like after training in one of our simulations. In the Figure, some of the row-column intersections indicate high-values by using a color code that simply reflects that a "quantization point" in the specific column of the matrix often followed a "quantization point" in the corresponding row. The figure also shows how a theoretical zoom into the matrix could look like.

Clustering "quantization points" into temporal adjacent groups consists on partitioning the transition matrix once sufficient training has been provided to the system. The goal is to partition the matrix in temporal coherent subgroups, that is, subgroups of spatial patterns that often enough during training tend to follow

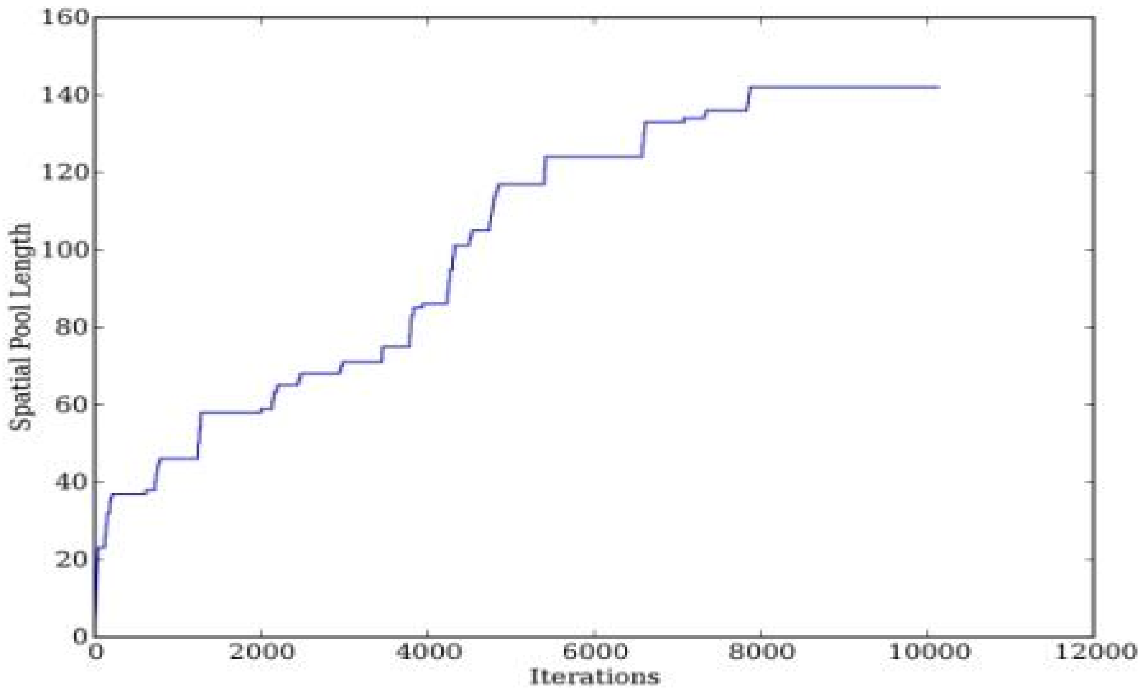


Figure 6.8: **Filling of the spatial pooler of a layer 1 node over time.** During the first stages of the learning phase, the node adds new quantization points to its spatial pool at a fast pace. Over time, the rate of growth of the spatial pooler substantially decreases.

each other in time. There are several ways of carrying out this partitioning. Here, we implemented a graph partition approach in which each quantization center represents a node in the graph and the values of the edges between the nodes represent the frequency with which one node transitions into another node. The idea is to maximize the objective J function [George and Jarosy, 2007].

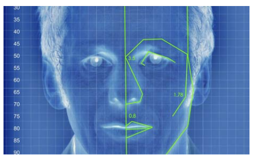
$$J = \frac{1}{2}\sum_{i=1}^{N_g} n_i * t_i \tag{6.1}$$

where n_i is the number of elements in the partition and t_i can be defined as a time-adjacency measure defined over D_i (disjoint subset of the set of quantization points $c_1, c_2, \ldots, c_N$) [George and Jarosy, 2007]:

$$t_i = \frac{1}{n_i^2}\sum_{k\in D_i}\sum_{m\in D_i} T(k,m) \tag{6.2}$$

where $T(k,m)$ denotes the $(k,m)^{th}$ entry of the time-adjacency matrix T. Therefore, t_i can be thought of as a measure of the average temporal similarity of the quantization points within cluster D_i [George and Jarosy, 2007].

The goal is to find the set of partitions $D_i = 1, ..., n$ that maximizes this objective function [George and Jarosy, 2007]. Unfortunately this is not computationally fea-



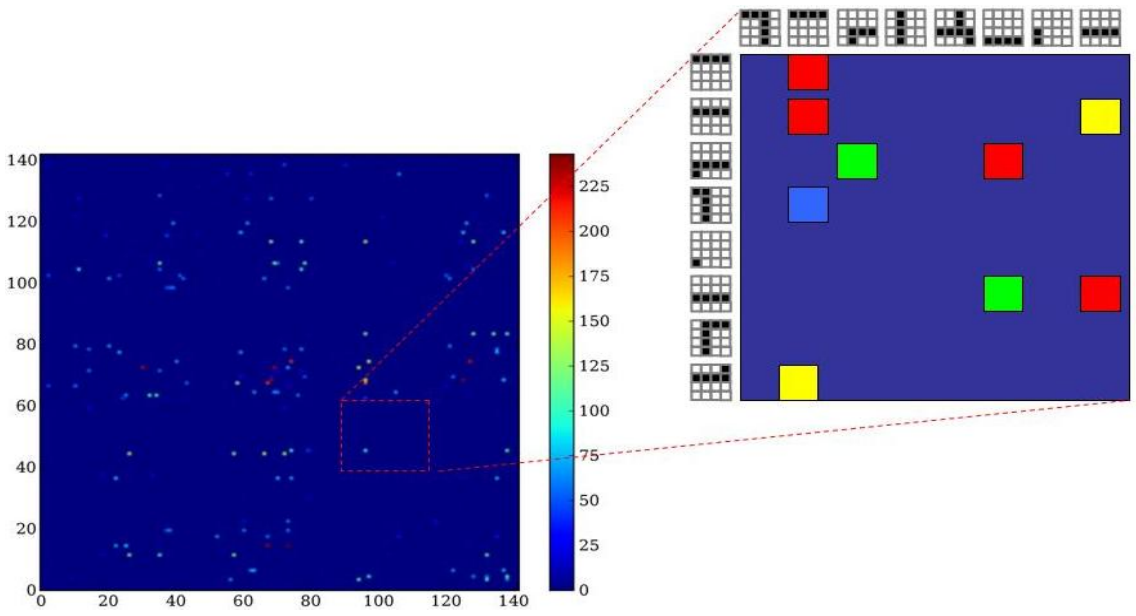


Figure 6.9: **Time adjacency matrix.** Actual time adjacency matrix produced in one of our simulations. On the right side, an interpretation of how a theoretical zoom into the matrix could look like.

sible, so greedy approaches must be used with the risk of reaching local optima that miss the global optimum. In our HTM implementation, we used the same greedy method employed by the authors of the HTM theory to partition the temporal matrix. The algorithm goes through the following steps as described in [George and Jarosy, 2007]:

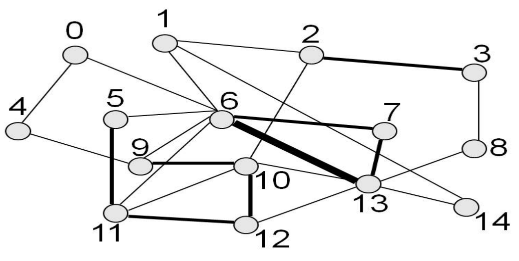
1. Find the most-connected quantization point in the temporal matrix that is not yet part of a group (i.e. the quantization point whose row in the time-adjacency matrix has the greatest sum).

2. Pick the N_{top} most-connected quantization points to this quantization point (we use N_{top} of 2, only quantization points that are not already part of a group are considered).

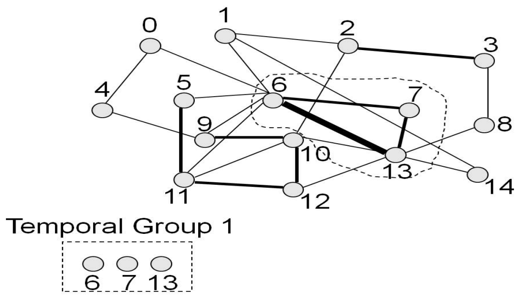
3. Repeat step 2 for each of the newly-added quantization points. With well-behaved groups, this process will terminate automatically, because all of point X's closest N_{top} neighbors are also likely to have point X as one of their N_{top} closest neighbors. If the group size does not automatically close off by the time the group reaches a certain size, the parameters "MaximumGroupSize" sets an upper bound by which the process is terminated and no new points are added).

4. The resulting set of quantization points is added as a new group. Return to step 1 until all quantization points have been grouped.

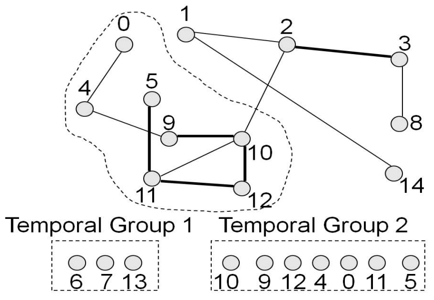
Figure 6.10 shows how the algorithm partitions the graph over time and creates temporal coherent clusters. The nodes of the graph represent quantization points and the width of the edges that join them, represent the frequency of transitions stored in the time adjacency matrix.



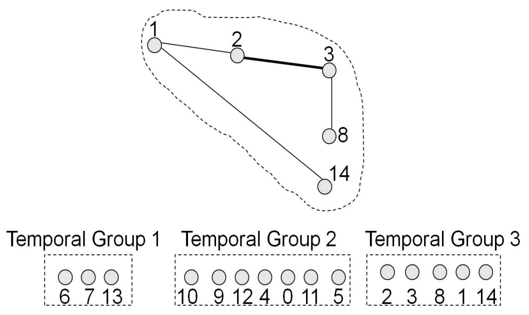
(a) Each node in the graph represents a "quantization point" and the edges joining the nodes represent the frequency by which one "quantization point" transitions into another.



(b) The first temporal group has been founded starting of from the most connected quantization point



(c) The second temporal group has been determined using the next top most connected quantization point

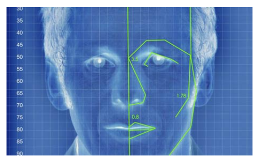


(d) The partioning of the graph has been completed and we have found 3 temporal coherent groups

Figure 6.10: **Formation of temporal groups.** The figure represents the sequential steps carried out by the temporal pooler (as described in the text) as it proceeds in finding temporal groups using the time-adjacency matrix's data. Explanatory graph taken from [George and Jarosy, 2007]

Figure 6.11 shows some temporal groups created by the algorithm just described on a node in layer 1 during one of our simulations. Notice how each temporal group is represented by a row and that each of the quantization points within a temporal group does not necessarily share a high pixel-to-pixel similarity with other quantization points in the group, they all do show a high degree ot temporal coherence though.

Once the temporal pooler has used the time-adjacency matrix to partition the quantization points into temporal coherent subgroups, the node can be switched to inference mode, where the node analyzes an input spatial pattern and determines to which one of all the quantization points stored in its memory is most closely related. A simple dot product operation [George and Jarosy, 2007] is used as a similarity



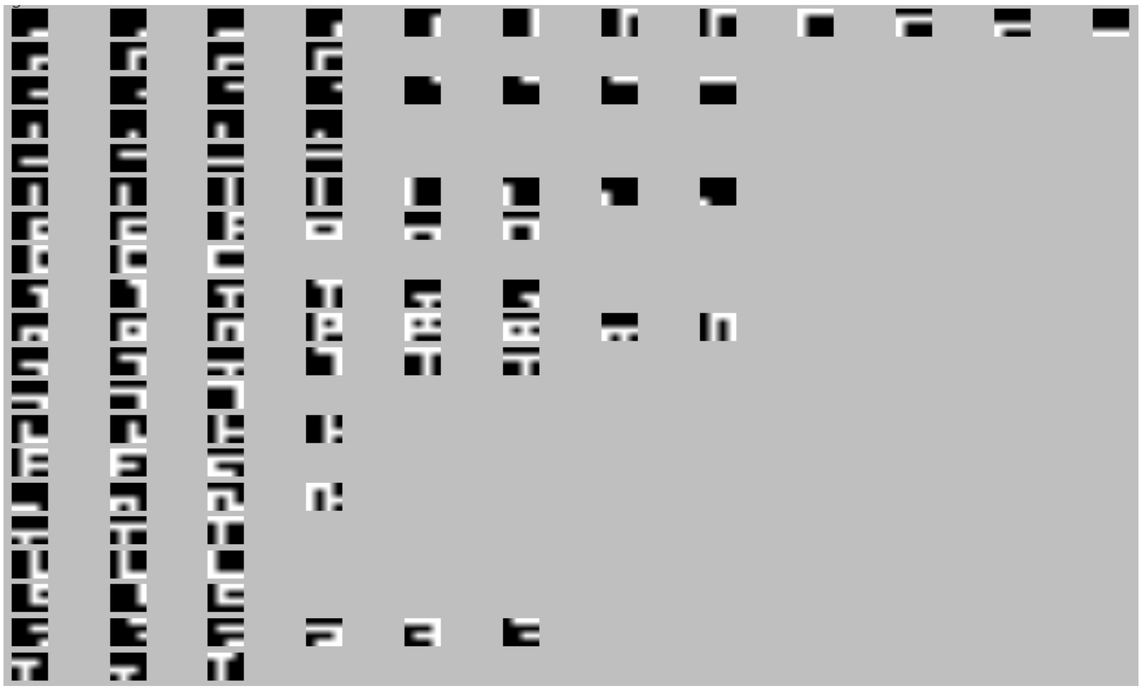
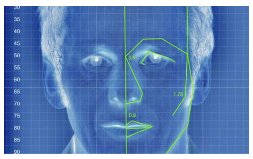


Figure 6.11: **Temporal groups created by a layer 1 node of our image recognition HTM.** Each row represents a temporal coherent group of spatial patterns that tend to follow each other in time.

metric. The node then determines to which temporal group this input spatial pattern belongs by just mapping which of the disjoint temporal groups contains the quantization point with the highest similarity to the current input spatial pattern. The node then, produces an output vector in the following way. A parameter determines how many temporal groups a node is able to report, if let's say this parameter is "6", the node fills all of the 6 elements of the output vector with "0" except the element representing the temporal group in which it is included the quantization point that the node believes it is perceiving at the current time instant. On that specific element, the vector contains a value of "1". So the output of the nodes consists of a vector with 6 elements, in which only one of the elements has the value "1" and the rest are "0s", see Figure 6.12.

Supervised Pooling at the Top Level

Each node of an HTM system uses time as a sort of supervisor to cluster together quantization points into temporal coherent groups using temporal proximity information. It is possible though, to use an external signal to pool quantization points into clusters. This approach makes sense at the top level node when an HTM system is trying to solve a pattern recognition task. In our experiments, the system was exposed to several drawings and while all the nodes carried out an unsupervised approach towards learning, the top level node used an external signal to map input patterns to external category names provided by the designer of the system. This



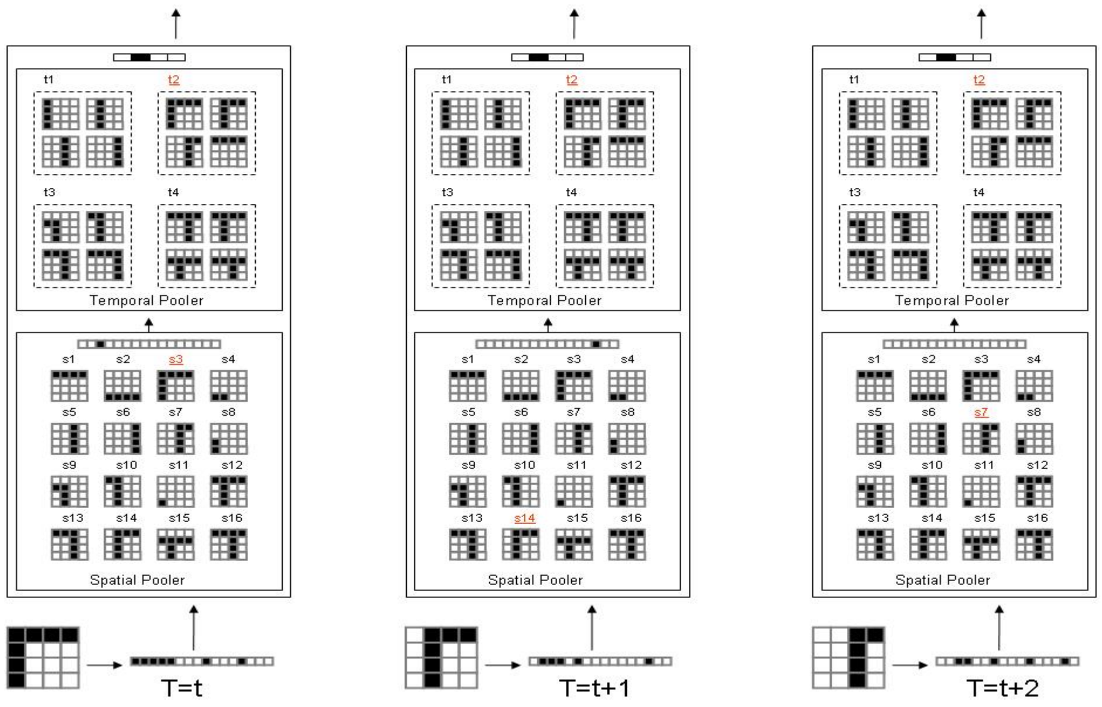


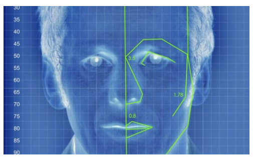
Figure 6.12: **Node operation over time**. The current active "quantization points" and "temporal groups" of each node as a result of its input data are indicated in red and underlined. Notice that although the input pattern varies over time, due to the existence of temporal coherent groups formed during training, the output of the node remains stable. Invariance has been achieved over this simple input space.

type of clustering, replaces the temporal pooler of the top node.

We used a very simple matrix that associated each vector learned by the spatial pooler of the top node (this vector is a row of the matrix) with an index category (a column of the matrix). The matrix is initialized to 0 at the beginning of the training process [George and Jarosy, 2007]. At each instant in time during training, the particular intersection of current active quantization point and current active index category was increased by one.

6.2.3 Hierarchical Node Operations

Training of an HTM system proceeds in hierarchical steps. At first only the nodes at the bottom layer of the hierarchy are trained. While on training, these nodes produce no output. When these nodes have been fully trained and switched into inference mode, the nodes in the next lever higher up in the hierarchy switch on their learning stage, and use the output generated by the nodes immediately underneath as training data. Meanwhile, the nodes higher up in the hierarchy are silent, waiting to start their learning process when the nodes immediately underneath have completed their learning stage and have been switched to the inference stage. This whole process



is repeated all the way up the hierarchy until all the nodes are fully trained and switched to inference mode. Once a node is fully trained, it can be switched to the inference mode. Inference is about identifying the temporal group to which the input spatial pattern is more closely related and passing this information in the form of a vector to the parent node above [George and Jarosy, 2007].

At the bottom level of the hierarchy, nodes receive their input from some artificial sensor or file. Higher-level nodes form their inputs by concatenating the outputs from their child nodes [George and Jarosy, 2007]. For any given parent node, its input is just a concatenation of the outputs of its children, Figure 6.13.

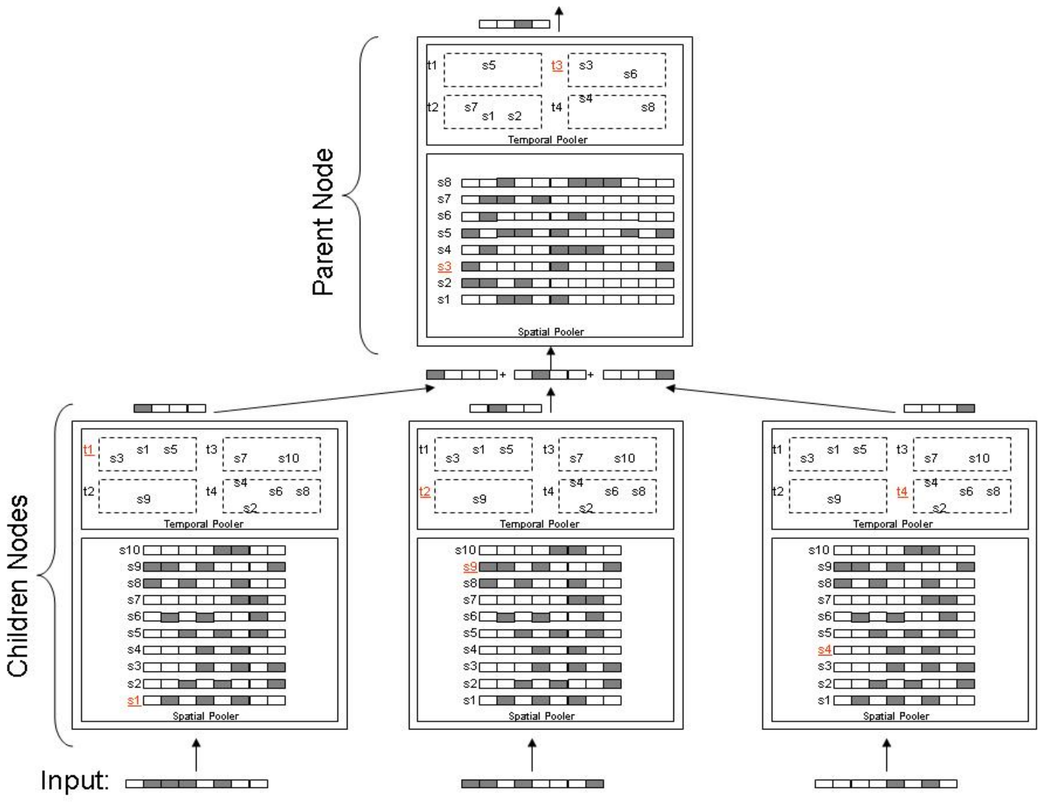
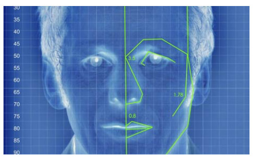


Figure 6.13: **Relationship between the output vectors of children nodes and the input vectors of parent nodes.** The input vectors from parent nodes are formed by aggregating the output vectors of children nodes.

6.2.4 HTM Implementation of Coincidence Detection

It is important to point out that a pervasive principle of neocortical function, that of coincidence detection, is present in the node operations of an HTM system. Basically, each quantization point stored by the spatial pooler is a co-occurrence of several temporal groups from the node's children nodes.



6.2.5 Usage of Probability Distributions in Message Passing

We already explained that the input vector to a parent node is the concatenation of the output vectors of its children nodes. The children nodes outputs carry information about the temporal groups they believe they are perceiving. A child's node output is just a vector of 0s and one 1, with the 1 element corresponding to the temporal group the node believes it is being exposed to 6.13.

The performance of the system could improve by using probability values representing the likelihood of a temporal group as elements of the output vectors instead of the current winner-take-all approach. This way, the node would not be force to settle on a particular temporal group if the input pattern is noisy or ambiguous. Using this approach, the node could generate as output a probability distribution of the likelihood that the input pattern it is being exposed to corresponds to each one of the temporal groups it has learned.

6.3 Results

Our HTM system consisted on an arrangement of 3 hierarchical layers (Figure 6.2). The bottom layer, layer 1, was composed of a total of 64 nodes (8x8) that completely tiled the grid input space. Each of these nodes had a receptive field of 4x4 pixels in the artificial retina's input grid. Layer 2 consisted of 16 nodes, each node had a receptive field composed of the 4 nodes immediately underneath in layer 1. Finally, layer 3 was composed of just one node, whose receptive field was the whole 16 nodes in layer 2. Learning starts at layer 1 and continues all the way up.

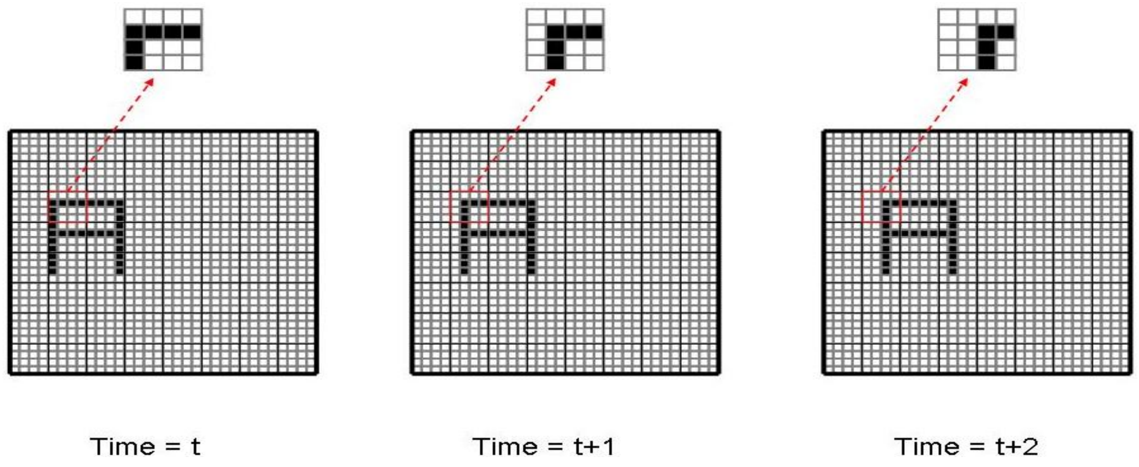


Figure 6.14: **Operation over time of the system.** The red square indicates the receptive field of a node in layer 1, and how the spatial arrangements that serve as input to this node change over time.

We formalized the algorithm described in the previous section and train it on a set of movies of line drawings. The movies consisted of horizontal and vertical displacements and flips along the Y axis for each one of the categories shown in

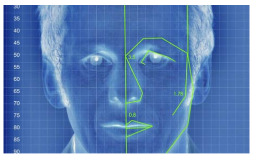


Figure 6.1. Figure 6.14 shows how training proceeds over time. There was a total of 48 categories with 4 representations of different sizes for each category. Each image was 32 by 32 pixels. The movies were created by placing one of the categories at 1 of 4 sizes and moving it vertically and horizontally with a flip through the Y axis included across the whole grid (until the edges of a figure reached the limits of the grid). A movement was produced from time step t to time step t+1.

6.3.1 Recognition

The system exhibits robust recognition performance on sets of images that had been distorted with relation to the training images, considerable amounts of noise where also included in the simulations as shown in Figure 6.15. Note that some categories are recognized besides a flip about the Y axis, translations and sizes oscillations. This is the result of the system having being trained with flipped and un-flipped image categories of different sizes and that have moved along the whole artificial retina. This shows the ability of the system to optimally learn the particular type of invariances under which it was trained.

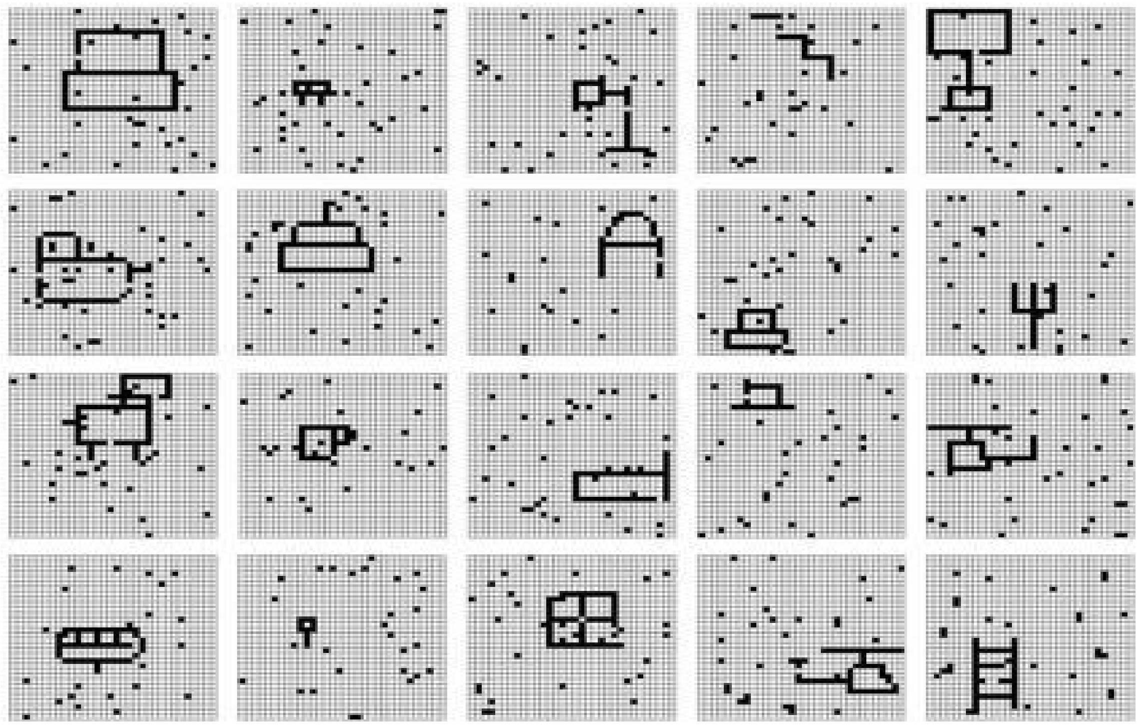


Figure 6.15: **Unseen images correctly recognized by HTM system (from left to right and top to bottom).** Computer, white board, table lamp, steps, wine glass, cat, stack, letter A, computer, rake, dog, mug, bed, hat, helicopter, bus, spoon, window, helicopter and ladder. If these images are compared with the training images in Figure 6.1, it is apparent that the trained HTM system has developed generalization and invariance capabilities to translation, distortion, noise, flips along the Y axis and size changes.

Different combinations of the algorithm parameters alter the specificity and sensitivity of the system. So far we have achieved a best simulation with a recognition accuracy of 93% over training images subjected to 3% noise ratio and a distortion chance of also 3% along and around the edges of the drawings. Figure 6.15 shows some images from the training library that underwent the described levels of distortion and noise and which were correctly recognized by our system.

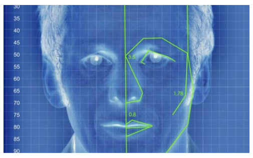
6.4 Conclusions and Directions for Future Research

This chapter of the thesis has described our HTM system in the context of a visual example. The system uses time as a supervisor by means of using proximity in time to group diverse spatial patterns into temporal coherent invariant representations of a certain cause. The hierarchical topology of the systems allows it to build increasingly complex configurations by using rearrangements of simpler parts: edges, corners and lines that give rise to more complex geometric figures when combined.

The key computational actor of the HTM system is the node. Each layer in the hierarchy is composed of several nodes. Each node is composed of two parts. A spatial pooler reduces the potentially infinite input space into a finite set of input patterns and a temporal pooler uses time as a supervisor to group quantization points into temporally-adjacent sets. Each parent node concatenates the output vectors of its children nodes to create its own input vectors. The pooling of several quantization points into a temporal coherent cluster produces an invariant representation of a feature. The spatial pooler of a high-level node learns coincidences of the invariant representations of its children nodes. The spatial-pattern learning process learns in the spatial dimension by storing particular coincidences of the invariant representations from lower level nodes [George and Jarosy, 2007]. The whole hierarchical system can be thought of as alternatively learning invariance (determining temporal-adjacent clusters) and selectivity (storing particular spatial arrangements within the input space) along the whole network to enable the system to recognize different objects besides their transformations. The alternation of these two techniques prevents the combinatorial explosion often associated to scalability in the pattern recognition problem.

The ability of the human brain to solve the invariant pattern recognition problem has been the subject of intense research for decades. Several systems have tried to emulate the capabilities of the human visual systems using various approaches. However, the performance of those systems has so far been limited and their ability to generalize questionable. HTM theory postulates that using continuity in time is the key cue to how the brain solves the problem. HTMs are able to capture multiple temporal and spatial scales at the same time.

HTM theory tries to encapsulate the algorithmic properties of the neocortex. The main capability of an HTM system is being able to discover the ultimate causes of persistent and recurrent patterns in the world, underlying sensory data. A cause can be a human face, a species of animal, a word, or an idea. An HTM system tries to create a model of the recurrent causes that persist in its input data by creating



internal representations of those causes. The learning period requires repeated exposure through the sensory input to causes that vary over time and that yet are persistent.

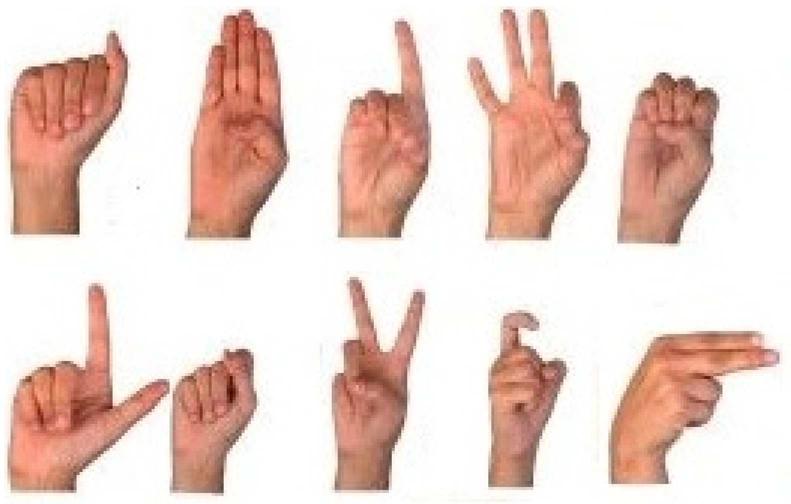
The physical world consists of a hierarchy of causes that an HTM tries to model. The HTM builds representations of lower level causes and higher level causes. Higher level causes are formed by combinations of lower level causes. Lower level causes in the hierarchy span shorter time dynamics and smaller areas of the input space [George and Jarosy, 2007]. The temporal dynamics of higher level causes is slower and they occupy wider regions of the input space (receptive field).

Discovery of causes is a pre-requisite towards later recognition. Once an HTM system has built a model of the world, it can infer what may be the most likely culprit of novel sensory input using the statistics stored throughout the hierarchy. During inference, information flows through the hierarchy all the way up to the top node that settles on the likeliest cause for its receptive field current input.

The described implementation of an HTM system has not yet fulfilled all of the requisites that the theoretical HTM postulates. In the following chapters we will examine the performance of a simple binary HTM (as the one described in this chapter), an HTM implementation from the authors of the theory containing all the theoretical requirements of the HTM paradigm and an extended version of our own binary HTM implementation to improve the performance of the algorithm in multivariable time series data sets.

Chapter 7

Extended HTMs in Sign Language Recognition - Comparative Study



7.1 Introduction

In this chapter we present an application of Hierarchical Temporal Memory (HTM) to the recognition of Sign Language [Rozado et al., 2010b]. In particular, we develop an extension of traditional HTM in order to adapt the algorithm to problems where instances possess a temporal structure that develops over time. We use as a proof of concept of our ideas, the specific problem of Australian Sign Language Recognition. We also compare the performance of our extended HTM system to well established machine learning approaches used in the literature [Rozado et al., 2011c] for Sign Language Recognition, SLR.

The theoretical aspects of the HTM paradigm have been thoroughly described in [Hawkins, 2006], [Hawkins, 2004] and in Chapters 4 and 6. An up to date version

of the theory with a probabilistic model of temporal aggregation and feedback information flowing from parent nodes to children nodes to disambiguate noisy inputs can be found in [George and Hawkins, 2009]. As the generative models that they are, HTM algorithms can be used to solve problems on different domains: pattern recognition, control or forward prediction [Numenta, 2006b]. In this Chapter as in the rest of the thesis, we center our attention to HTMs applied within the realm of temporal pattern recognition.

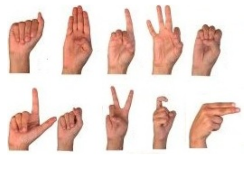
The main objective of an HTM network trained for pattern recognition is the development of generalization capabilities [George and Hawkins, 2005]. That is, given a set of categories, each one of them represented by a set of instances, the system should learn to properly separate the category-space using a small subset of training instances from each category set. After training, the system should be able to generalize and properly assign the correct categories to unseen instances from the category space.

HTM algorithms perform robustly in traditional machine learning tasks such as image recognition [George and Hawkins, 2005]. Problems where HTM excel are those with an inherent spatio-temporal structure and whose instances are fully represented spatially at any given time instant. For problems where an instance is composed of a time series of spatial arrangements, HTMs performance is not as robust.

In this paper we develop a feature for HTMs to perform better on learning tasks whose instances unfold over time: we modify the HTM's top node by enabling it to store sequences of spatio-temporal input patterns arriving at the top node over time. This top node also performs similarity measurements among incoming sequences in order to map unseen instances to known categories. The rationale for using sequences to map stimuli to categories has been justified in [Abeles, 2004], [Rabinovich et al., 2006], [Rodríguez and Huerta, 2004].

We illustrate the performance of our modified HTM system in the problem of Sign Language Recognition. Sign language is used by deaf people to communicate by means of using sequences of hand movements instead of speech. Sign Language constitutes a good fit for the type of problem that we wanted to tackle: category spaces whose instances are composed of an ordered sequence of spatial arrangements. Therefore, we chose a data set containing time series of hand movements representing signs from Australian Sign Language (ASL) as a proof of principle that our extended HTM system can perform well on this type of problems.

In [Starner et al., 1998], authors use a four-state HMM for the recognition of a 40 categories set of American Sign Language. The recognition accuracy of this video-based approach was 91%. The authors used a language model to improve recognition accuracy in a sentence-based context, without which, the performance of the algorithm in isolated SLR dropped to 74%. The work in [Liang and M., 1998] shows another HMM being applied to recognize a set of 250 categories of Taiwanese Sign Language (TSL) using a DataGlove. This work divided the input data in several predefined and supervised categories: posture, position, orientation and type. Authors achieved high recognition accuracies for categories embedded within sequences. However, the recognition accuracies for isolated gestures, without the usage of language models, also dropped the recognition accuracy to 70% for 250



categories and to 84% for 71 categories. The work from [Fang and Gao, 2002] used a combination of Simple Recurrent Networks (SRN) and HMM and a high number of dimensional features: 48. Their methods were computationally demanding for real time usage and oriented towards identifying proper partition of sentences and signer independent recognition and achieved a high recognition rate of Chinese Sign Language. The work in [Holden et al., 2005], also used HMMs for video based SLR of sentences, pulling heavily on a language model and predefined features to be recognized as well as face tracking. The authors of the data set [Kadous, 2002] that we used for our work, achieved SLR accuracy of up to 93% using a symbolic decision-tree like approach. However, their method is intrinsically of a highly supervised nature, since the features to be matched by the algorithm were previously defined by the authors. Recently, authors in [Kapuscinski and Wysocki, 2009] have also used HTMs for Polish SLR using video images. They achieved a 96% recognition rate for isolated words on a vocabulary set consisting of 101 categories. In their work, authors used an elaborated and innovative pre-processing of the data to divide gestures into their spatial structure (position, shape and movement of hands) and their temporal structure (visemes) and then fed the HTM algorithm with this data representation.

As shown previously, state of the art SLR reaches robust levels of performance [Liang and M., 1998]. Yet often, methods in the literature make use of pre-defined features to be search for by the recognition algorithms, pre-processing of the data, large feature input vectors and language models for additional support. When language models are not used, performance decreases substantially [Starner et al., 1998]. This chapter shows that our extended HTM system outperforms traditional HTM algorithms and that it reaches levels of performance similar to those in the existing literature using low dimensional features and without the usage of a language model, pre-defined features to be detected, pre-processing of the input data, nor a customized vocabulary set. Our approach has potential applications in simultaneous translation of Sign Language to natural language or between different dialects of Sign Language as well as applicability to other multi-variable time series recognition problems since it is not optimized nor specifically designed to handle the type of input data used in this work.

7.2 Methods

For our analysis, we compare under different contexts the SLR accuracies of binary HTMs (our own in-house version of traditional binary HTMs as explained in chapters 4 and 6), Markovian HTMs (a more sophisticated implementation using Bayesian belief propagation), our extended HTM system, the Metafeatures TClass algorithm and Hidden Markov Models.

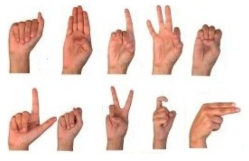
7.2.1 Data

The data set used to measure the performance of our algorithms was a publicly available data set of the 26 letter-alphabet Australian Sign Language (ASL) [Kadous]. It consists of a vocabulary of 95 signs categories with 27 instances per category [Kadous, 2002]. The data was recorded from a single native signer volunteer using ASL. This data set was chosen for its open accessibility, easiness of experimental reproducibility and the appropriateness of the data structures contained within to the type of problem we wanted to tackle. The list of categories in the training set is shown in Table 7.1.

God	I	Norway	alive	all
answer	boy	building	buy	change mind
cold	come	computer PC	cost	crazy
danger	deaf	different	draw	drink
eat	exit	flash-light	forget	girl
give	glove	go	happy	head
hear	hello	his/hers	hot	how
hurry	hurt	innocent	is true	joke
juice	know	later	lose	love
make	man	maybe	mine	money
more	name	no	not my problem	paper
pen	please	polite	question	read
ready	research	responsible	right	sad
same	science	share	shop	soon
sorry	spend	stubborn	surprise	take
temper	thank	think	tray	us
voluntary	wait not yet	what	when	where
which	who	why	wild	will
write	wrong	yes	you	zero

Table 7.1: List of all the Australian Sign Language categories contained in the data set and used in the simulations.

The data set was captured at 100 Hz using a pair of electronic data gloves containing accelerometers and sensors to track 11 channels of information for each hand: x, y and z spatial coordinates of the hand, the roll, pitch and yaw rotation angles of the wrist and a bend coefficient for each finger, see Figure 7.1. A particular configuration of all the channel variables at one instant in time is referred to in this paper as a frame. The average length of each sign was 57 frames. As illustrated in Figure 7.1, when performing a particular sequence of hand movements, the gloves provide dynamic data representing the spatio-temporal transitions of the tracked variables as they oscillate over time while the hands “utter” a particular sign.



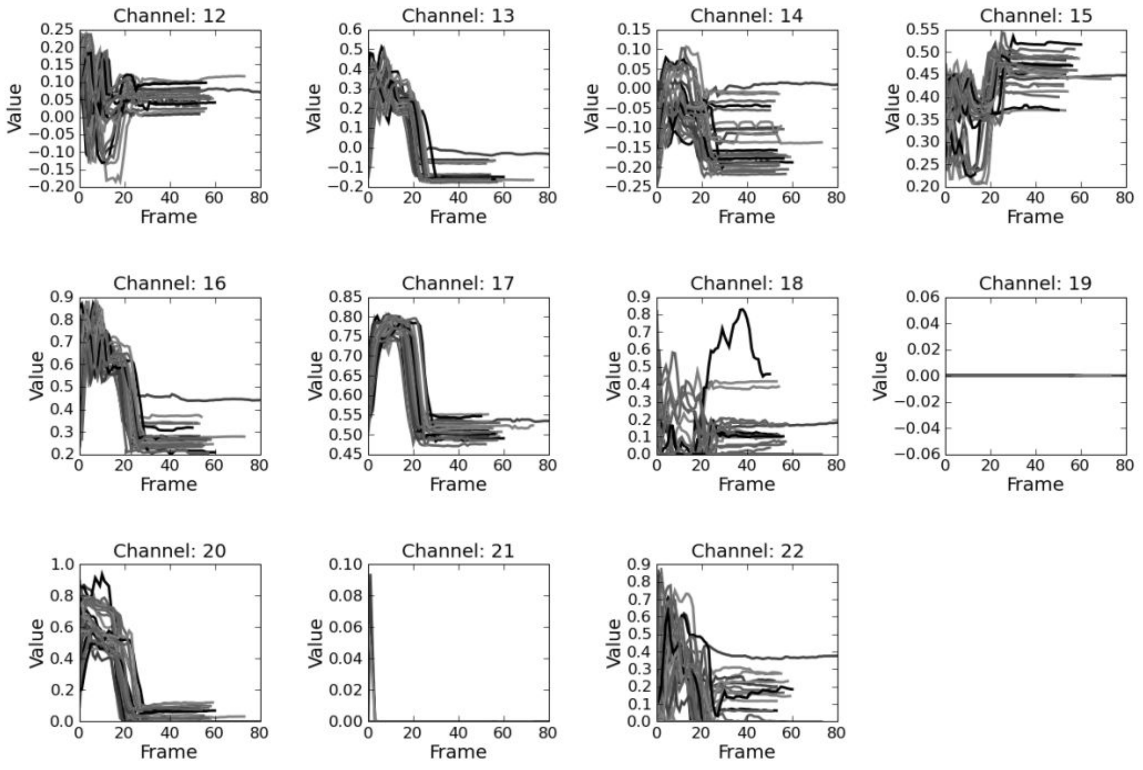


Figure 7.1: **Channel dynamics of a sign.** Visual representations of how channel dynamics of the right hand unfold overtime for several instances of the sign "danger".

7.2.2 Markovian HTM

A more sophisticated implementation of an HTM system than simple binary HTMs involves the use of a probabilistic generative model (Figure 7.2), and Bayesian belief propagation, (Figure 7.3). We refer to this type of HTM network as a Markovian HTM. In such an implementation, each node contains a set of coincidence patterns or CPs: $c_1, c_2, .., c_n \in C$ and a set of Markov chains or MCs: $g_1, g_2, .., g_n \in G$. CPs represent co-occurrences of sequences from their children nodes. Each MC is defined as a subset of the set of CPs in a node. CPs capture the spatial structure of nodes or sensors underneath in the hierarchy by representing vectorially the co-activation of MCs in a node's children. A MC activated in a parent node concurrently activates its constituent MCs in the node's children. The MCs capture the temporal structure of a set of CPs, i.e., the likelihood of temporal transitions among them. The incoming vectors to an HTM node encapsulate the degree of certainty over the child MCs. With this information the node calculates its own degree of certainty over its CPs. Based on the history of messages received, it also computes a degree of certainty in each of its MCs. This information is then passed to the parent node. Feedback information from parent nodes toward children nodes takes place by parent nodes sending to children nodes their degree of certainty over the children node's MCs.

We used Numenta's Nupic package (v1.7.1) [George and Hawkins, 2009] which is an implementation of a probabilistic Markovian HTM to run simulations with this

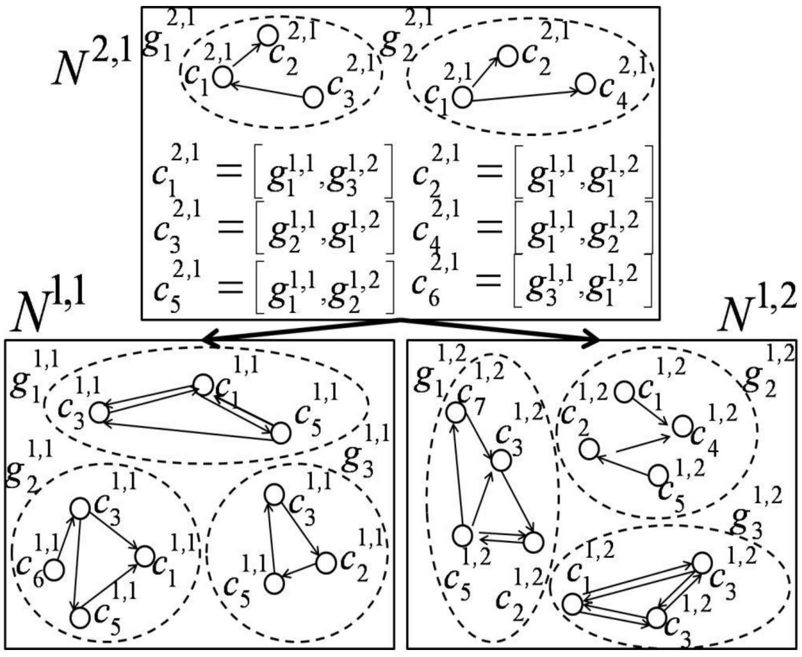

$N^{2,1}$
$c_1^{2,1} = [g_1^{1,1}, g_3^{1,2}]$
$c_2^{2,1} = [g_1^{1,1}, g_1^{1,2}]$
$c_3^{2,1} = [g_2^{1,1}, g_1^{1,2}]$
$c_4^{2,1} = [g_1^{1,1}, g_2^{1,2}]$
$c_5^{2,1} = [g_1^{1,1}, g_2^{1,2}]$
$c_6^{2,1} = [g_3^{1,1}, g_1^{1,2}]$
$N^{1,1}$
$N^{1,2}$

Figure 7.2: **HTM as a generative model.** A simple two level hierarchy consisting of a parent node and 2 children nodes is shown. Each node contains a set of CPs, c's, and a set of MCs, g's, defined over the set of CPs. A CP in a node represents a co-activation of a subset of MCs in its children nodes.

type of HTM implementation.

7.2.3 Metafeatures TClass

TClass is a supervised learner for multi-variable time series created by the author of the sign language data set used in this work [Kadous, 2002].

TClass works by first looking for sub-events in the training streams - in forms that the user has previously specified. Then, TClass analyses the extracted sub-events and selects those that are important, typical or distinctive. For each of the training instances, the algorithm looks for the sub-events found before. The presence or absence of these temporal patterns are used as attributes for propositional learning. The propositional learner built in this fashion is then used to perform inference on test data.

TClass has been used in classification tasks such as electrocardiographs classification and SLR [Kadous]. TClass runs on any platform that supports Java 2 and has been released under the Gnu General Public License.

7.2.4 Hidden Markov Models for SLR

Hidden Markov Models have been often been used in speech recognition and SLR [AL-Rousan et al., 2009; Mohandes et al., 2007; Fang and Gao, 2002; Liang and M.,

Likelihood over CPs:	$y_t(i) = P(-e_t \mid c_i(t)) \propto \prod_{j=1}^{M} \lambda_t^{m_j}(r_i^{m_j})$ where CP c_i is the co-occurrence of $r_j^{m_1}$'th MC from child 1, $r_i^{m_2}$'th MC from child 2,..., and $r_i^{m_M}$'th MC from child M.
Feed-forward likelihood of MCs	$\lambda_t(g_r) = P(-e_0^t \mid g_r(t)) \propto \sum_{c_i(t) \in C^k} \alpha_t(c_i, g_r)$ $\alpha(c_i, g_r) = P(-e_t \mid c_i(t)) \sum_{c_j(t-1) \in C^k} P(c_i(t) \mid c_j(t-1), g_r)\alpha_{t-1}(c_j, g_r)$ $\alpha_0(c_i, g_r) = P(-e_0 \mid c_i(t=0))P(c_i(t=0) \mid g_r)$
Belief distribution over CP	$Bel_t(c_i) \propto \sum_{g_r \in G^k} \beta_t(c_i, g_r)$ $\beta_t(c_i, g_r) = P(-e_t \mid c_i(t)) \sum_{e_j(t-1) \in C^k} P(c_i(t) \mid c_j(t-1), g_r)\beta_{t-1}(c_j, g_r)$ $\beta_0(c_i, g_r) = P(-e_0 \mid c_i(t=0))P(c_i \mid g_r)\pi_0(g_r)$
Message to be sent to children nodes	$\pi^{m_i}(g_r) \propto \sum_i I(c_i)Bel(c_i)$, where $I(c_i) = \begin{cases} 1 & \text{if } g_r^{m_i} \text{ is even} \\ 0 & \text{otherwise} \end{cases}$

Figure 7.3: **Belief Propagation Equations for HTM Nodes.** The reader is encouraged to take the Node $N^{2,1}$ from Figure 7.2 as reference. $N^{2,1}$ contains 6 CPs and two MCs. Each MC is composed of 3 CP. In this table c_i is the i^{th} coincidence in the node. g_r is the r^{th} MC in the node. $-e_t$ indicates the bottom up evidence at instant t. $-e_0^t$ indicates the evidence sequence from time $0...t$. $+e$ stands for top-down evidence. λ is the feed-forward output of the node. λ^{m_i} represents the feed-forward input to the node from its child node m_i. π is the feedback input to the node. π^{m_i} is feedback output of the node to its child node m_i. y is the bottom-up likelihood over CPs in a node. α is a bottom-up state variable for the MCs in a node. β is a state that combines bottom-up and top-down evidence for a MC in a node. B_{c_i} represents belief in the i^{th} CP in a node.

1998; Starner et al., 1998]. HMMs assumes that the modeled process is determined by a finite number of states and that states change randomly once per time step in a statistically predictable way (Markovian assumption)[AL-Rousan et al., 2009].

In our work, gesture categories are modeled as a single HMM with N states per gesture $(s_1, s_2, \ldots, s_N)$. A compacted notation for an HMM is: $\lambda = (\pi, A, B)$. $A = a_{ij}$ is the state-transition probability distribution with $a_{ij} = P(q_t = s_j \mid q_{t-1} = s_i)$; the index runs over $1 \leq i, j \leq N$. The observation symbol $B = b_j(k)$ is the probability distribution in state s_j, with $b_j(k) = P(v_k t \mid q_t = s_j)$; the index runs over $1 \leq j \leq N$ and $1 \leq k \leq M$. M is the number of distinct observation symbols per stated denoted by $V = v_1, v_2, \ldots, v_M$. $\pi = \pi_i$ is the initial state distribution with π_i [AL-Rousan et al., 2009].

HMM based pattern recognition involves three separate stages:

- The training stage consists on determining the (π, A, B) triplet that most probably described a set of observations of a category. That is, estimating

the parameters $\lambda = (\pi, A, B)$, given a set of observations O. The Viterbi algorithm solves this problem by iteratively adjusting the parameters π, A, and B. At each iteration, the most likely path through an HMM is calculated. This path produces the current assignment of observation vectors O_t to states s_j [AL-Rousan et al., 2009].

- Determining the most likely sequence of states $S = s_1, s_2, \dots, s_T$ that could caused the observation sequence O corresponds to the decoding stage. The Viterbi algorithm is used to determine the best state sequence.

- Evaluation consists of finding the mot likely HMM (π, A, B) from a set that generated a given sequence of observations (in our work: hand movements). The probability of an observation given a particular HMM is calculated with the forward algorithm. That is, given the observation sequence $O = O_1, O_2, \dots O_T$ and the model $\lambda = (\pi, A, B)$, the $P(O \mid \lambda)$ probability that the observed sequence is produced by the model is calculated.

For our simulations, we used the General Hidden Markov Model library (GHMM) a freely available LGPL-ed C library implementing algorithms, data structures and training routines for HMMs[1].

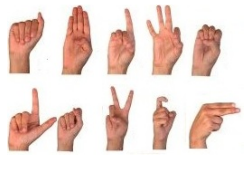
7.2.5 Extended HTM

We have developed a binary version of the HTM theory as proposed in [Hawkins, 2006]. Forward (up the hierarchy) message passing from children nodes to parent nodes is implemented using binary vectors, containing just 1s and 0s in each element without feedback messages from parent nodes. A binary vector indicates which temporal group is active at a time. Our extended HTM formalism has been developed in order to adjust the system to the specific needs of multivariable time-series.

The fundamental modification of our local HTM system with respect to traditional HTM networks is the modification of the network's top node whose task in original HTM algorithms is simply to map incoming vectors from children nodes to categories. The newly defined top node stores instead complete sequences of spatio-temporal input vectors in its *sequential pooler* and maps those sequences to categories.

We have tested our approach in the problem of sign language recognition. Sign language recognition is fundamentally different from previously tried out problems within the HTM community, [George and Hawkins, 2005]. Most problems undertaken by HTMs, [Numenta, 2006b], consist of instances whose spatial configuration is fully represented at any time instant. Sign language is composed of sequences of spatial arrangements over time that together constitute a sign. At any given time t, the complete representation of the sign is not available, just a particular spatial arrangement of the hands. It is the particular temporal sequence of hand arrangements what constitutes a sign. The fundamentally different nature of this kind of

[1]Available at: `http://ghmm.org/`



problem and the poor performance of traditional HTM networks to deal with it justified the undertaking of modifications within the HTM inner-workings.

Topologically, our extended top node sits at the top of the HTM network, receiving its inputs from a traditional top node underneath serving the purposes of its unique children node, see Figure 7.4.

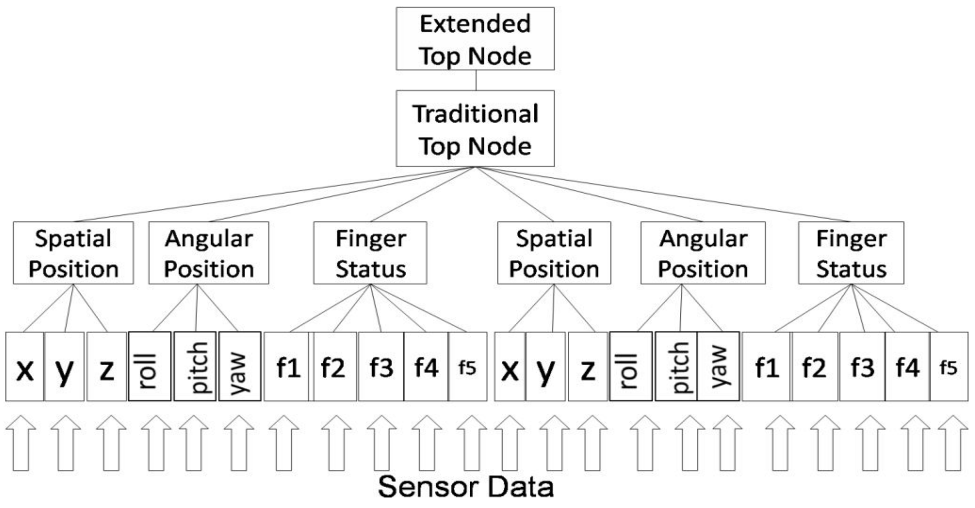


Figure 7.4: **Optimal Topology of the HTM Network used for Sign Language Recognition.** The extended top node sits at the top of the topology receiving its inputs from a traditional top node. The reasons of the topological arrangement of the sensor channels responds to the HTM theoretical requirement of clustering in the bottom levels of the hierarchy channels with high correlation, leaving the top node to figure out more distant or elusive dependencies, see Figure 7.11.

To deal with the sign language data structures, the extended HTM top node stores sequences of incoming vectors from children nodes in an abstraction that we referred to as *"sequential pooler"* and maps the whole sequence to a specific category, not just the individual elements. Each element of the sequence is a spatio-temporal configuration pattern of the hands at any given instant. The aggregation of this elements form sequences that encapsulate the whole spatio-temporal structure of a sign as it evolves over time. Figure 7.5 illustrates how the learning of a sign comes about over time in our modified top node by storing the sequence of spatial coincidences that follow each other in time during the "utterance" of a sign. A whole sequence captures the entire spatio-temporal structure constituting a sign from beginning to end, see Figure 7.6.

Since the spectrum of all possible sequences (to store during training and to recognize during inference), would quickly overflow the memory available to the node, a means to cluster different sequences into the same category was needed. This clustering was carried out by performing similarity measurements between an incoming sequence and previously stored sequences and classifying the incoming sequence as belonging to a certain cluster of similar sequences according to a threshold.

The need for a measurement of similarity is two-fold: It is needed during train-

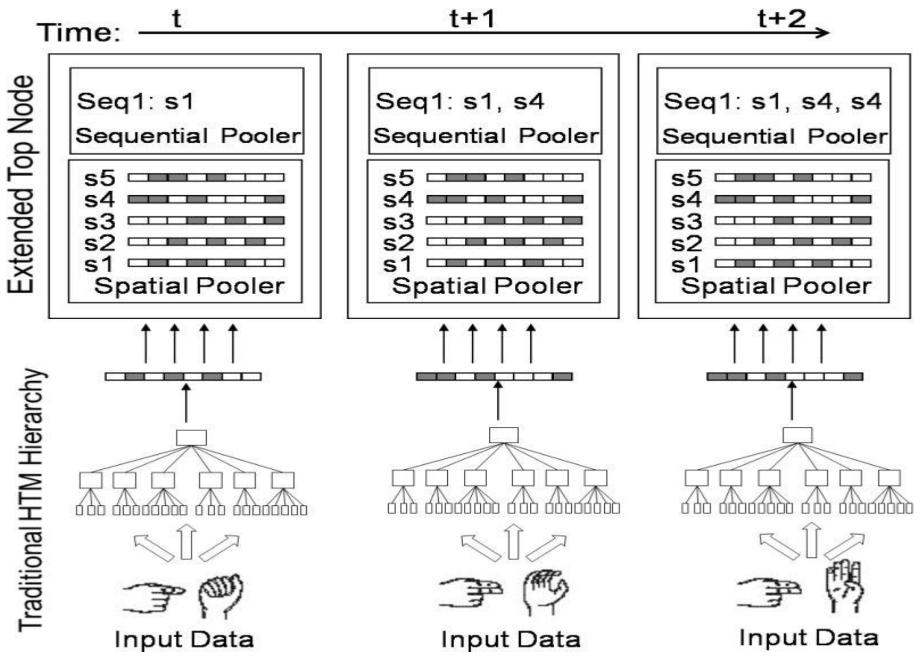


Figure 7.5: **Extended HTM formalism - A new Type of Top Node.** Traditional top nodes receive binary vectors representing the temporal groups active in the nodes underneath in the hierarchy and map this incoming vectors to categories. Our extended top node instead stores sequences of incoming vectors in an abstraction we refereed to as *sequential pooler* and maps whole sequences to categories. These sequences capture the "utterance" of a sign over time.

ing in order to determine which sequences to store and which ones to disregard, in case of high similarity to an already stored sequence. A similarity measurement is also needed during inference to determine which sequence, from the set of stored sequences in the sequential pooler of a top node has the highest degree of similarity to an incoming input sequence, see Figure 7.7. The similarity measurements between sequences representing signs were carried out using the Needleman-Wunsch algorithms [Needleman and Wunsch, 1970] that employs dynamic programming for sequence alignment. Dynamic programming has been successfully used by the bioinformatics research community to calculate the degree of similarity between genetic sequences. The inner workings are shown in Figure 7.8 where a matrix, D, representing the alignment between two candidate sequences, SeqA and SeqB, is calculated using a scoring scheme that scores matches, mismatches and gaps between both sequences. Each element, $D(i,j)$, in the matrix row i and column j is calculated using the function:

$$|D(i,j)| = max \begin{cases} D(i-1,j-1) + s(x_i, y_j) \\ D(i-1,j) + g \\ D(i,j-1) + g \end{cases} \tag{7.1}$$

The s function represent the score for a match or a mismatch at the matrix

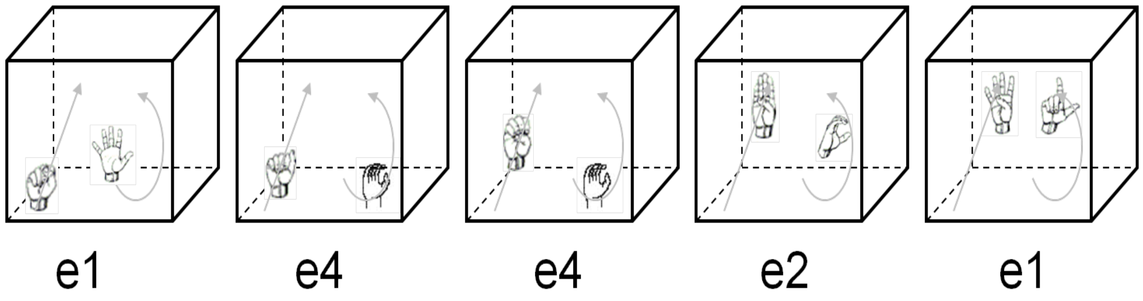


Figure 7.6: **A Sequence Representing a Sign as Stored in the Extended Top Node.** A complete representation of a whole sign from beginning to end in sequence form as represented in the extended top node by the sequence: e1, e4, e4, e2, e1. Each element of the sequence represents a spatio-temporal configuration of the hands.

element $D(i, j)$ according to the corresponding scoring scheme. The g constant indicates the gap cost according to the scoring scheme. The score at the bottom right of the matrix, see Figure 7.8, indicates the degree of similarity between both sequences for the global alignment. The alignment can be traced back using a traceback matrix, T, starting from the bottom right element and traveling backwards according to:

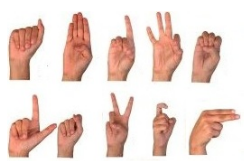
$$T(i, j) = argmax \begin{cases} D(i-1, j-1) \\ D(i-1, j) \\ D(i, j-1) \end{cases} \tag{7.2}$$

7.2.6 Experimental Design

We measured recognition accuracies and the computational resources taken by each HTM algorithm on an 32 bit Intel Core Duo CPU E6550 @ 2.33GHz and 2.00 GB RAM with Windows 7 installed. The measurements were done with the time function of the time module from Python. Ten runs were carried out for each simulation and the mean and variance were calculated from the results.

Several topologies were tried out varying the number of layers in the network as shown in the Results section, see Figure 7.12. The topology shown in Figure 7.4 was selected after it proved to be the one that optimized the performance of the network during inference over several alternative network designs.

The suitability of the optimal topology in Figure 7.4 was confirmed by a cross correlation analysis, see Figure 7.11. HTM theory predicts that HTMs networks work better if highly correlated channels are grouped in the lower levels of the hierarchy [Hawkins, 2006], leaving the upper layers to find out more complex, distant or not so obvious correlations. The network topology that we found to be optimal according to its performance during inference was in fact one that grouped highly



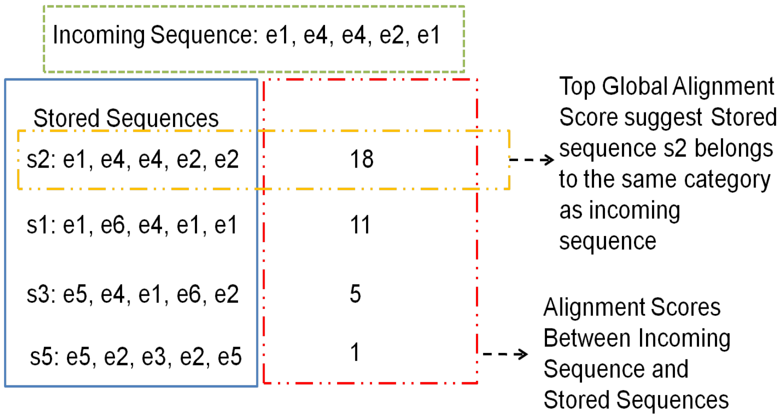

Incoming Sequence: e1, e4, e4, e2, e1
Stored Sequences
s2: e1, e4, e4, e2, e2
18
s1: e1, e6, e4, e1, e1
11
s3: e5, e4, e1, e6, e2
5
s5: e5, e2, e3, e2, e5
1
Top Global Alignment Score suggest Stored sequence s2 belongs to the same category as incoming sequence
Alignment Scores Between Incoming Sequence and Stored Sequences

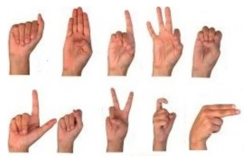
Figure 7.7: **Alignment Rankings.** The extended top node performs sequence alignment between each incoming sequence and all the previously stored sequences. A ranking according to similarity score obtained from the sequence alignment algorithm is generated. The category of the stored sequence with the highest similarity to the incoming sequence is inferred to be the same as the category of the incoming sequence.

correlated input channels, such as the fingers of each hand, in lower level nodes, Figures 7.11 and 7.4.

We measured the performance of the different methods being tested by measuring, after training, what percentage of unseen signs' instances were assigned to their proper categories using the inference scores provided by the corresponding algorithm. K-fold cross validation was used during simulations to determine the recognition accuracy during inference for each algorithm being compared. The training data set was randomly partitioned into 10 groups. Each learning algorithm was trained 10 times, using all of the training set data points except one and then tested on the data left out in the iteration. The number of errors was recorded and the mean error over the 10 tests was reported as the accuracy level achieved by the algorithm.

For the cross validation analysis in our simulations, we used a uniform value of $k = 10$ since it seems to be a good rule of thumb, although the true best value probably differs for each algorithm and each data set. We chose k-fold cross validation for being less "wasteful" of data than test set cross validation and less "expensive" than leave-one-out cross validation.

Our lack of in house electronic data gloves hardware made us rely on a previously generated and publicly available data set whose number of categories and instances per category was predetermined. In order to simulate the performance of the algorithms with additional training instances per category, we generated 23 artificial instances per category by averaging the values of random tuples and triplets of instances from the time series data set.



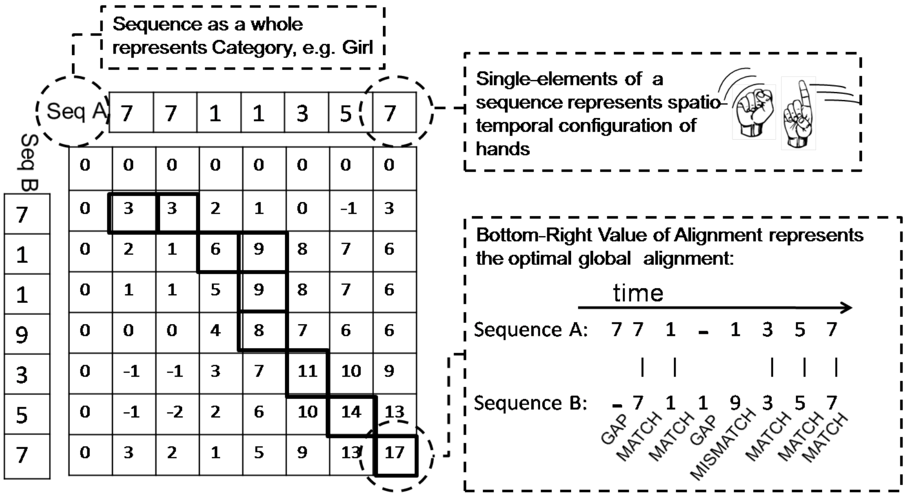

Sequence as a whole represents Category, e.g. Girl
Seq A
7 7 1 1 3 5 7
Seq B
7 1 1 9 3 5 7
Single-elements of a sequence represents spatio-temporal configuration of hands
Bottom-Right Value of Alignment represents the optimal global alignment:
time
Sequence A: 7 7 1 - 1 3 5 7
Sequence B: - 7 1 1 9 3 5 7
GAP MATCH MATCH GAP MISMATCH MATCH MATCH MATCH
17

Figure 7.8: **Dynamic programming.** The extended top node uses sequence alignment using dynamic programming to measure similarity among two sequences. The external row and column represent 2 sequences formed by spatial coincidences units. The score at the bottom right of the matrix indicates the top global alignment between both sequences. A scoring scheme was used to score matches, gaps and mismatches in the alignments.

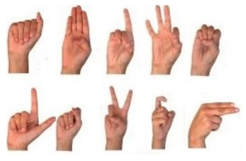
7.3 Results

The original data contained intrinsic noise, making it necessary to filter the data, see Figure 7.9. For each channel, an average value for each category was determined. Then, when any instance differed significantly from the category average, the difference was subtracted from the absolute values for that particular instance. This procedure visibly reduced noise deviations in the data set.

We created an HTM topology formed by 3 layers as shown in Figure 7.4 that proved to be the one that optimized the performance of the network over several alternative network designs as explained below. The bottom layer nodes received their input data from the 11 channels of input data coming from each glove. Several topologies were tried out varying the number of layers, nodes, fan-in edges, fan-out edges, spatial-pooler specificity coefficients and clustering coefficients for the nodes.

Since the original data contained continuous variables, the next filtering step was to transform the continuous input values into discrete input values. That is, the continuous input space had to be converted into a discrete input space. For each channel, the range of possible values was segmented into a finite number of regions. Each region represented a particular discrete value of the input space. If the value of a continuous variable from an input channel fell into a particular region, the channel adopted the discrete value associated with that region.

An entropy analysis for each channel was performed in order to determine the proper segmentation value for each channel, see Figure 7.10. The objective was to determine in how many discrete segments the continuous variable should be divided



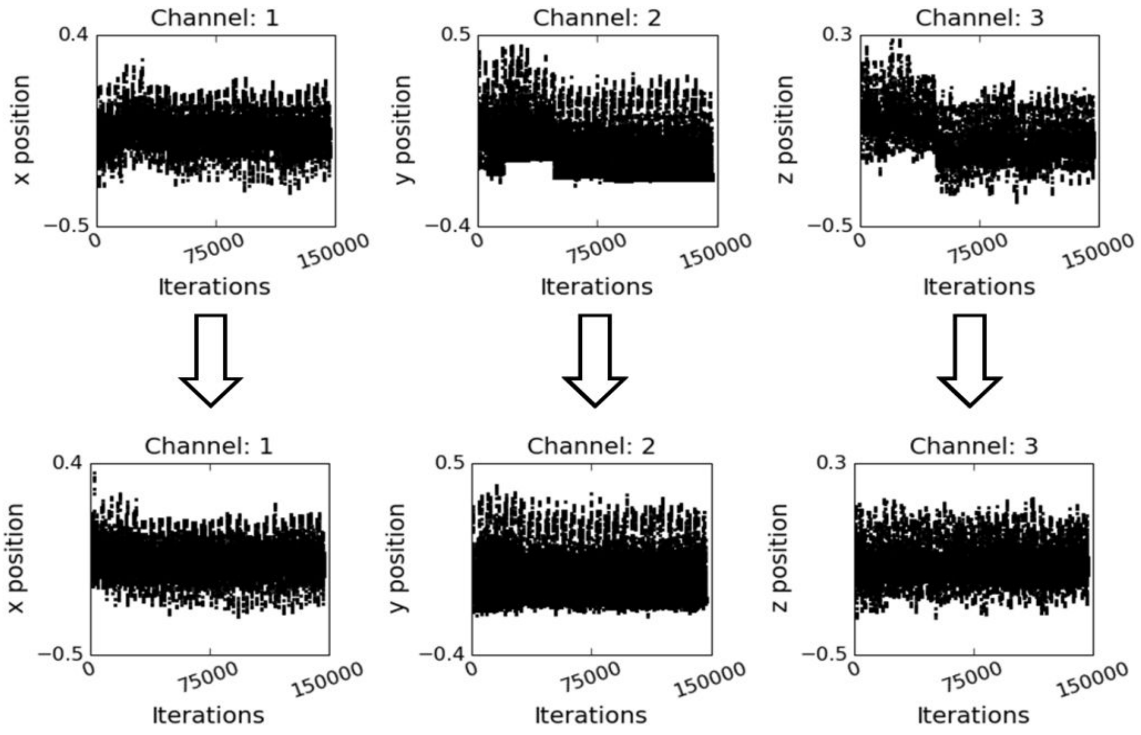


Figure 7.9: **Pre-processing of the data for normalization purposes.** Signals from the original data set underwent a normalization process to remove offsets and drifts in the data.

in order to minimize entropy. The entropy of a sequence of values in a channel $X = x_1, x_2, x_3, ..., x_N$ is defined as:

$$H(X) = \sum_{n=1}^{N}(p_i \cdot log(p_i)) \tag{7.3}$$

$$p_i = \frac{|x_i|}{||X||} \tag{7.4}$$

$$||X|| = \sum_{1}^{N}(|x_i|) \tag{7.5}$$

The probability of value i is represented as p_i, $|x_i|$ is the number of times value x_i occurs and $|X|$ is the length of the sequence. The optimal parameters suggested by the entropy analysis and confirmed by manual supervision settled down on segmentation values between 2-6 regions for each channel.

A cross correlation analysis was performed in order to find out correlations among input channels, see Figure 7.11. This information was used to design the optimal network topology, see Figure 7.4. HTMs work better if highly correlated channels are grouped locally in the lower levels of the hierarchy [Hawkins, 2006], leaving the upper layers to find out more complex, distant or not so obvious correlations.

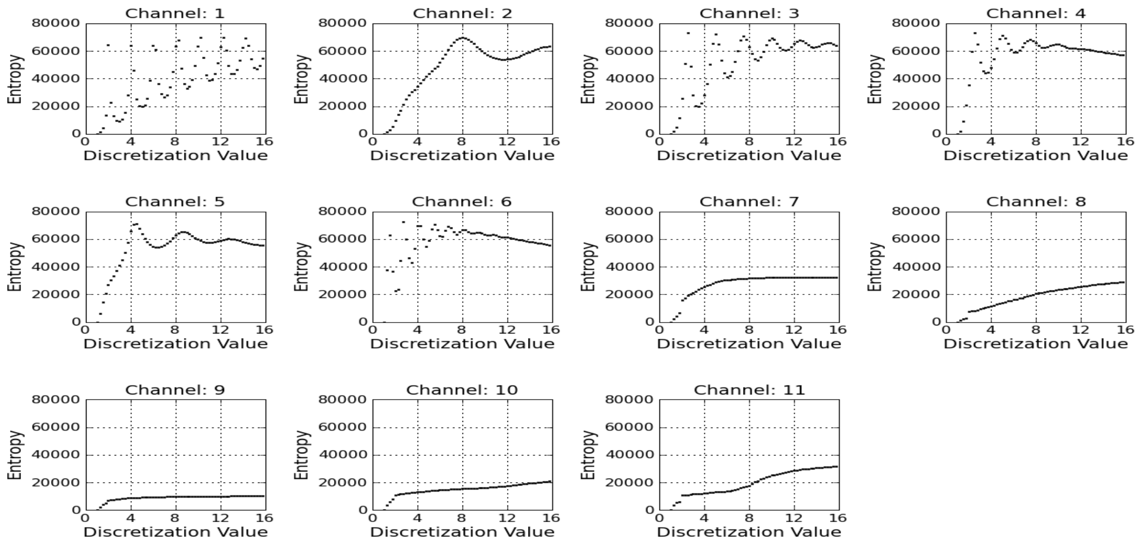


Figure 7.10: **Entropy analysis of left hand.** Determining the right partitioning values, ν, to transform the continuous input variable into ν discrete states. The optimum ν according to simulation trials coincided with low entropies when comparing different ν values' impact on performance. The entropy of a sequence of values in a channel $X = x_1, x_2, x_3, ..., x_N$ is defined in the text in Equation 7.3.

Accordingly, our network topology grouped in its bottom layers those variables that were highly correlated according to Figure 7.11.

HTM theory predicts the ability of HTM networks to warp time by increasing the number of layers in the topology [George and Jarosy, 2007]. That is, a higher number of layers in the topology should extend the temporal invariance of the algorithm at the higher nodes of the network and hence, the HTM algorithm should perform learning and inference on problems where data structures slowly unfold over time. To test this theoretical prediction, we carried out a number of simulations testing the recognition accuracy of different Markovian HTM networks with increasing number of layers in their topology to determine how increasing the number of layers affects performance. Since HTMs make an inference for each time instant, or frame, an aggregation of all the inferred categories over the whole length of a sign was performed. The most common category over the set of all the inferences was taken as the inferred category for the whole sign. As can be seen in Figure 7.12, increasing the number of layers improves performance on the problem at hand but only up to the 3 layers level. Increasing the number of layers further does not improve recognition performance. This first experiment highlighted the necessity for HTM algorithms to be expanded in order to accommodate recognition problems where instances unfold slowly over time.

Next, we compared the recognition accuracies reached by the different methods being compared in this work and our extended HTM algorithm. Figure 7.13 provides a summary of results for different simulation types and methods.

The results of the simulations using a binary HTM network as described in [Hawkins, 2006] are in rows 1-3. Binary HTM networks obtained the poorest recog-

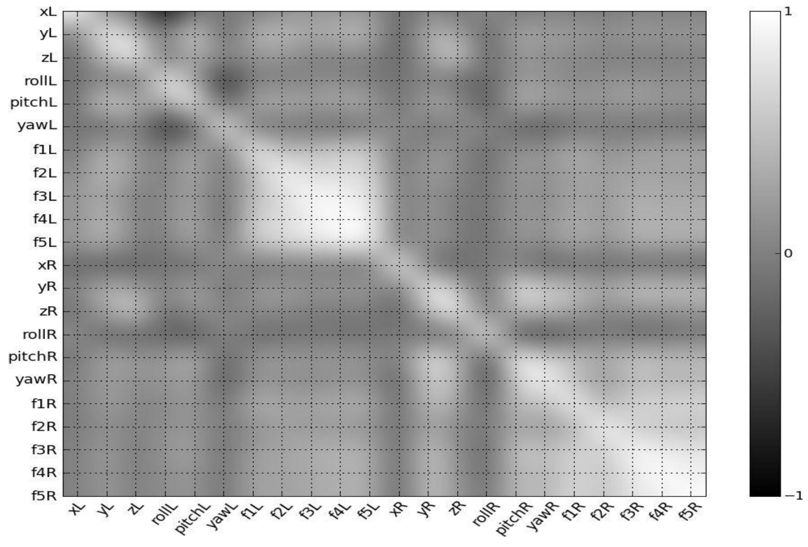


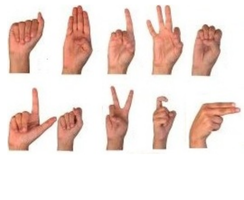
Figure 7.11: **Cross Correlation Study.** Cross-correlation matrix of the 22 channels from the data set. This matrix was used to find the optimal topology of the network since highly correlated input channels should be grouped in the network's lower layers. All the fingers of each hand are highly correlated. This justifies its clustering in the lower layers of the HTM topology employed.

nition accuracy results of all the methods being compared for the task of SLR. This can be explained in the sense that the top node in this binary HTMs was just trying to guess the proper category of an instance from a given frame during the performance of a sign. However, a given time frame of the 11 channels of information for each hand can be shared by several signs since a frame is just a flash-view arrangement of the hands in space. Only the appropriate sequence of spatial arrangements over time uniquely constitutes a sign. As explained above, inference was carried out by pooling the inferences for all the frames constituting a sign. As Figure 7.13 shows, this approach was not very successful to deal with this type of problem.

The recognition accuracy of a Markovian HTM network using Markov Chains for its temporal clustering is shown in row 4 of Figure 7.13. This Markovian HTM using a probabilistic Bayesian-like model with Markov Chains improves performance significantly over a binary HTM. Still, the approach does not warp time enough for robust recognition.

Row 5 shows the results of the Metafeatures T-Class approach used by the creators of the sign gestures data set used in this work. Row 6 shows the results of using HMMs on the data set. Both the Metafeatures T-Class approach and a traditional HMM algorithm obtained robust recognition results on the sign language data set. This was expected since both methods possess an established track of robust recognition performance in the literature [Kadous, 2002; AL-Rousan et al., 2009; Mohandes et al., 2007] for this type of problems.

The rest of the simulations, rows 7-16, were carried out with our extended HTM system as described previously and employed different degrees of specificity and



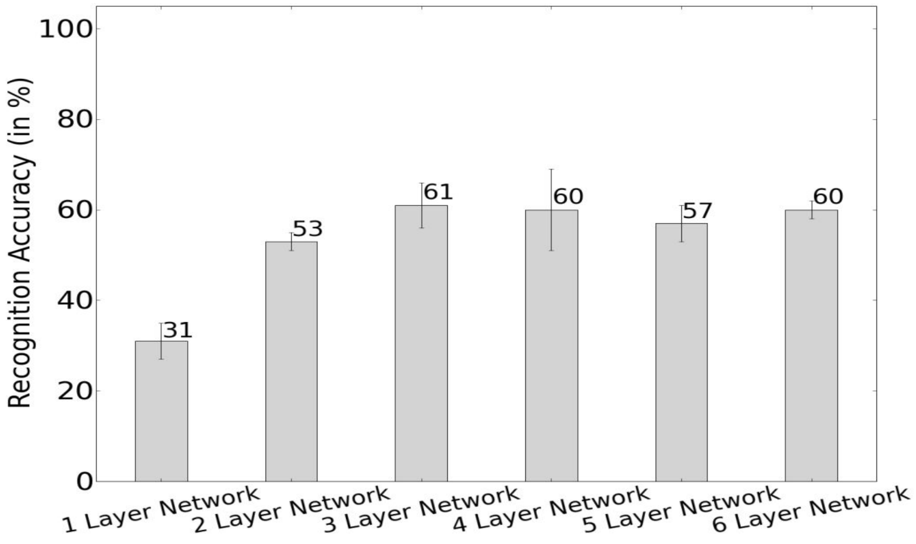
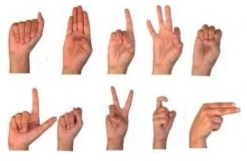


Figure 7.12: **Optimal topology search.** Recognition accuracy during inference for Markovian HTM networks with different number of layers. The bars show the percentage of correct classifications achieved by 1, 2, 3, 4, 5 and 6 layer networks.

generalization capabilities depending on the degree of granularity used to partition the input space. If a simulations was carried out by partitioning the input space in a high number or regions (5 or more), the “Spe” tag was used for *Specificity*. That is, a high degree of granularity is used in the transformation of the continuous input space into a discrete space so high discrimination between similar values can be achieved. If a simulation was carried out by partitioning the input space in just a few regions (4 or less) the “Gen” tag was used for *Generalization*, see results in Figure 7.13. In this way, values that are not so close might be assigned to the same region since the continuous input space is partitioned into just a few regions. Simulations with input data derived from the original data such as first and second derivatives, integrals and averages of the position information for each of the x, y and z channels, are indicated with the “Der”, “Der2”, “Int”, and “Ave” tags in Figure 7.13. Simulations in which only the channels from the right hand were used are indicated by the “Rha” tag. Combinations of different HTM networks that exploit different data representations improve the performance of the algorithm. Results of combining several data representations are shown in rows 14, 15, and 16 in Figure 7.13.

The degree of granularity used for the discretization of the continuous input space in the data set was of critical importance for the performance of our extended HTM algorithm. A trade-off emerged between the specificity and generalization capabilities of the network. Excessive granularization, that is, partitioning the continuous input space into too many regions, increases the specificity of learning, but leads to over-fitting, lesser generalization abilities and also an explosion in terms of storage and processing requirements. The more instances available for training, the higher the degree of specificity that can be reached, but our data set was constrained



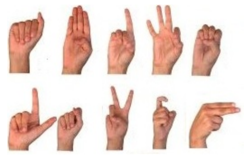
Row	Simulation type	Accuracy
1	Binary HTM (Gen)	23%
2	Binary HTM (Spe)	27%
3	Binary HTM (Optimal)	38%
4	Markovian HTM	61%
5	Metafeatures TClass	93%
6	Hidden Markov Models	88%
7	Extended HTM with DP (Gen)	73%
8	Extended HTM with DP (Spe)	74%
9	Extended HTM with DP (Rha)	70%
10	Extended HTM with DP (Der)	57%
11	Extended HTM with DP (De2)	38%
12	Extended HTM with DP (Int)	59%
13	Extended HTM with DP (Ave)	56%
14	Combined Extended HTM (Gen+Spe)	82%
15	Combined Extended HTMs (Gen+Spe+Rha)	85%
16	Combined Extended HTMs (Gen+Spe+Rha+Der+De2+Int+Ave)	91%

Figure 7.13: **Simulations performance.** Recognition accuracies during inference for the different methods being compared in this work. The last 3 rows show the results of combining several data representations in our extended HTM system.

to just 27 samples for each sign. Hence, a highly partitioned input space hampers the ability of the network to generalize and produce proper categorization on unseen input signs. On the other hand, a very unspecific partition of the input space, favors generalization capabilities but also decreases the specificity of the network, that is, the number of false positives increases. An entropy analysis was performed to find out the optimum degree of granularity needed to transform the input space into a discrete space while maintaining an optimum trade-off between specificity and generalization capabilities. Some granularizations, or data representations, were optimal just for the recognition of some signs while not for others. There was no apparent optimum data representation for all signs. Combinations of different HTM networks through a pooling system that aggregates the results of different network simulations improved overall performance of the system with accuracy jumping up to 91%.

The input data's absolute x, y and z coordinates of the hands is sometimes not enough to completely describe a sign. Other variables such as first and second derivatives of the x, y, and z coordinates further describe a particular sequence of hand movements. Therefore, we used derived data from the absolute values provided by the data set and performed simulations using the derived variables such as velocity and acceleration information corresponding to the x, y and z variables.

Combinations of different HTM networks that exploit different data representations improve the performance of the algorithm. Therefore, a method was needed in order to carry out combinations of results of several simulations. We settled



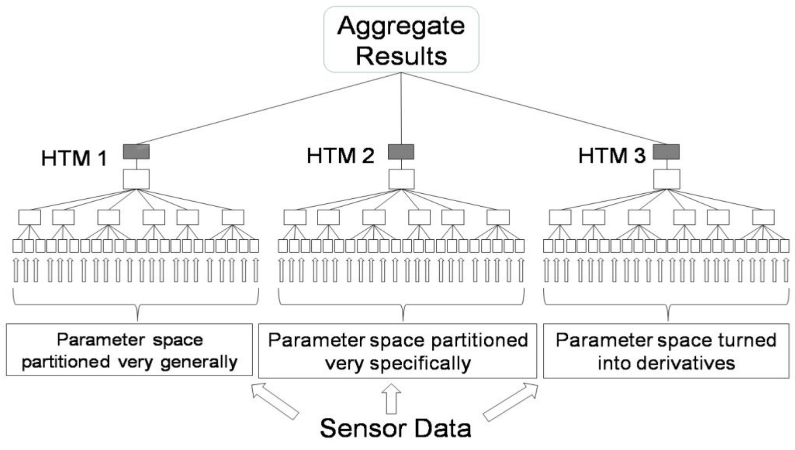

Aggregate Results
HTM 1
HTM 2
HTM 3
Parameter space partitioned very generally
Parameter space partitioned very specifically
Parameter space turned into derivatives
Sensor Data

Figure 7.14: **Network aggregation.** The results of HTM networks with different configuration parameters and fed with different data representations can be pooled to improve overall performance.

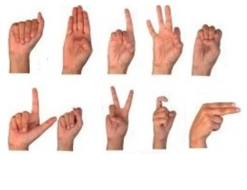
down with a simple aggregated sum of results from different simulations as a way of pooling the results of several simulations into a combined result, see Figure 7.14. That is, each trained HTM network was tested by making it to perform inference over a set of unseen instances. For each inference over an unseen instance, a rank of sequences stored in the sequential pooler of the top node which were similar enough to the input instance was generated. Every sequence in the rank had a particular score and category associated with it. The score was the result of performing a global alignment between the input sequence and this specific sequence stored in the sequential pooler. Pooling the results of several simulations simply consisted of adding all the scores associated to a certain category in the similarity rank generated for each input instance, see Figure 7.15.

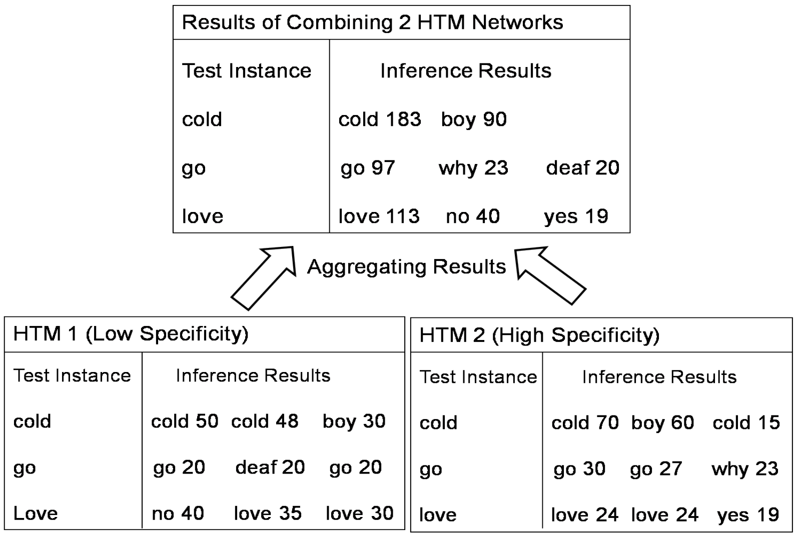
We also studied the impact of training sets vocabulary size on recognition accuracy, i.e. how increasing the number of categories to be recognized negatively affects performance for the different algorithms being compared. However, the degree of negative impact is different for each method as can be seen in Figure 10.11.

To determine the effect on recognition accuracy of increasing the number of training instances for each method being compared, we plotted the results of simulations in which we increased the number of training instances in discrete steps for the 5 methods being compared. As can be observed in Figure 10.12, increasing the number of training instances improves performance in general but this trend is more marked for the extended HTM methodology.

The inference time of our extended HTM algorithm was on the order of milliseconds showing its potential for real time applications since CPU resources consumption during inference is a limiting factor for real time SLR.

A defining characteristic of HTM systems is that they require a relatively high number of training instances for good recognition performance [George and Jarosy, 2007]. However, we did not generate the data set used for our simulations and hence,





Results of Combining 2 HTM Networks			
Test Instance	Inference Results		
cold	cold 183	boy 90	
go	go 97	why 23	deaf 20
love	love 113	no 40	yes 19

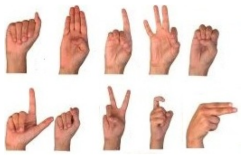
HTM 1 (Low Specificity)			
Test Instance	Inference Results		
cold	cold 50	cold 48	boy 30
go	go 20	deaf 20	go 20
Love	no 40	love 35	love 30

HTM 2 (High Specificity)			
Test Instance	Inference Results		
cold	cold 70	boy 60	cold 15
go	go 30	go 27	why 23
love	love 24	love 24	yes 19

Figure 7.15: **Aggregation of results.** The results tables of two different HTM network simulations can be combined into a single table by adding up the scores associated to each category in the similarity rank.

we could not generate additional instances. We were therefore, constrained to the existing 27 instances per category. To study how increasing the number of training instances improves performance for our extended HTM algorithm, we needed to obtain additional instances of training data beyond the 27 instances provided by the data set we employed. To solve this problem, we used the original data in the training set to generate additional artificial data and plotted the recognition accuracies results in Figure 7.18. The Figure suggest that our system has a wide room for improvement as increasing the number of training instances markedly improved performance more than for the other methods.

Most of the literature on SLR uses data generated ad-hoc for a particular experiment. There exist no standardized sign language data set for performance comparison purposes. Hence, simple optimization of the category list can boost performance. To show this, we created a confusion matrix to determine how each method mislabeled classes during inference. We trimmed the 10 most often mislabeled categories of the confusion matrix for each method, and run simulations again with just the remaining 85 categories, Figure 7.19. Simply removing the 10 most conflicting categories in the data set significantly improves the performance of all the algorithms, but among the top 3 performing methods (Tclass, HMM and Expanded HTM) our extended HTM obtained the biggest improvement. This highlights the need in the SLR community for standardized data sets in order to compare fairly the recognition performance of SLR algorithms since performance can be easily enhanced by manual curation of the vocabulary set.



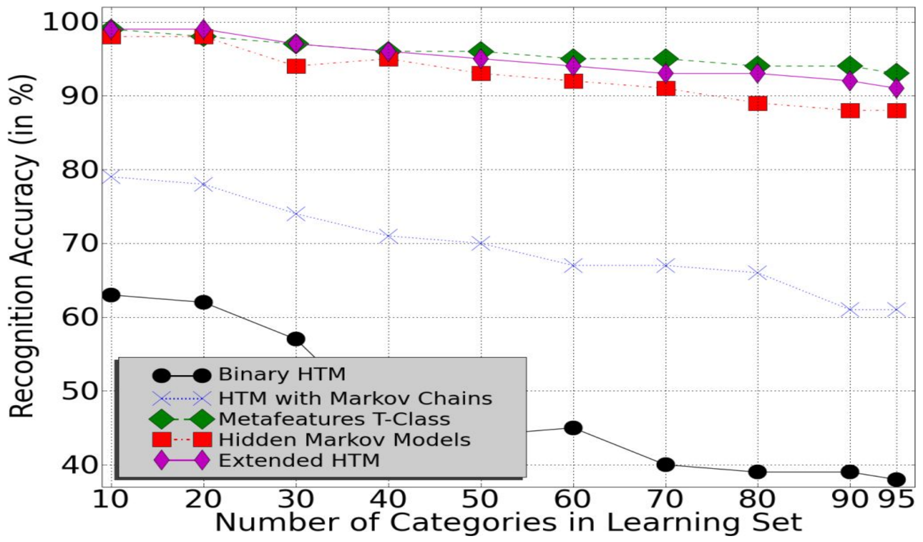
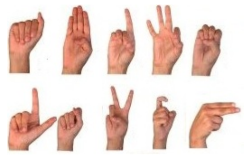


Figure 7.16: **Decreasing performance with increasing vocabulary size.** The recognition accuracy for each method being compared decreases as the vocabulary size of the sign gestures data set to be learnt increases.

7.4 Discussion

Our extended HTM formalism top node's fundamental property was its ability to store and compare sequences of spatio-temporal input vectors representing signs. In SLR, it is the orderly sequence of input vectors arriving over time what constitutes a sign, as oppose for instance to image recognition, where at any time instant, the input vector represents a complete characterization of an image category. As shown in Figure 7.13, the extended HTM improves the performance of traditional HTMs for the problem of SLR and reaches similar levels of performance to well-established methods often used in the literature such as HMMs. This good performance comes about because of the specific nature of our tackled problem. In our data set, each frame considered in isolation does not represent any particular sign just a flash-view spatial arrangement of the hands in space. Only the appropriate sequence of spatial arrangements over time uniquely constitutes a sign. The top node in traditional HTMs just tries to guess the proper category of an instance for every given frame during the performance of a sign. But a given time frame or arrangement of the 11 channels of information for each hand can be shared by several signs and hence the poor results of this method. To overcome the traditional HTMs shortcomings, our Extended HTM formalism creates a top node whose main task is to store sequences of spatio-temporal input vectors as incoming input instances unfold over time.

A critical aspect for the performance of the algorithm is the degree of granularity used for the discretization of the continuous input space. The discretization process represents a trade-off between specificity and generalization capabilities of the network. Excessive granularization, that is, partitioning the continuous input space into too many regions, increases the specificity of the learning, but leads to overfitting and less generalization abilities of the network. Obviously, the more instances



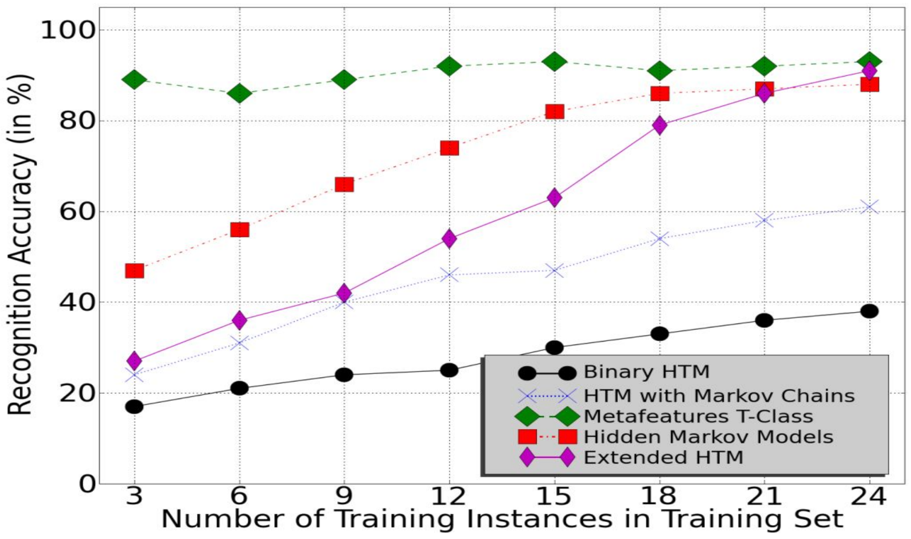


Figure 7.17: **Improving performance with increasing number of training instances.** The recognition accuracy for each method being compared increases as the number of training instances gets larger.

available for training, the higher the degree of specificity that can be reached, but our data set was constrained to just 27 samples for each sign. A very unspecific partition of the input space favors generalization capabilities but also decreases the specificity of the network, that is, the number of false positives increases. An entropy analysis was performed to find out the optimum degree of granularity to transform the continuous input space into a discrete space.

We are aware that the use of electronic gloves for automatic SLR imposes additional hardware requirements for natural context applications. Image based SLR often can be computationally demanding due to the necessity of performing preprocessing of video streams which sometimes limit its applicability to real time recognition systems. In the work by [AL-Rousan et al., 2009], their system required up to 7 seconds just for image pre-processing. The emergence of cheap 3D scanners which interpret 3D scene information using continuously-projected infrared structured light, such as Microsoft's Kinect, offers some obvious advantages for real-time-anywhere SLR. As the precision of such systems is expected to become sensitive enough to precisely detect fine wrist or finger movements, the technology holds great potential for the near future. Although, in this work we have focused on input coming from data gloves, HTM theory in general and the extended HTM approach in particular are very flexible in terms of accepting input data streams from a variety of sensors. As long as the data representations in the input data contains a spatio-temporal structure, our extended HTM approach after training with a sufficient number of training instances would be able to achieve good recognition accuracy. The method is also light on computing power during inference, requiring only milliseconds, which makes it easily applicable to real-time requirements contexts.

The extended HTM system is successful when used upon a learning task such

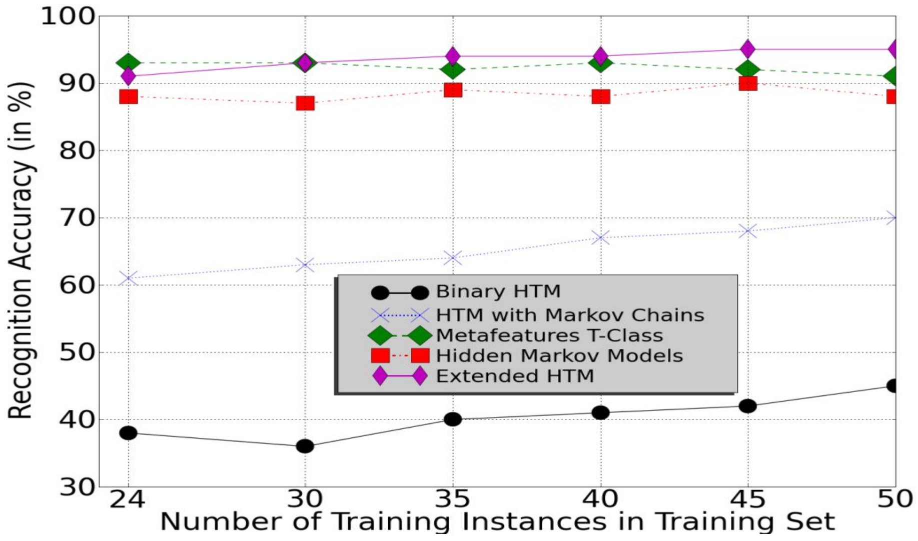


Figure 7.18: **Performance with artificial data.** 26 additional instances of artificial data for each category were generated bringing the total number of instances per category to 53. Each method being compared was trained with this expanded training set and recognition accuracy was measured for comparison purposes. Our extended HTM method shows a larger performance improvement than the Tclass and HMMs methods.

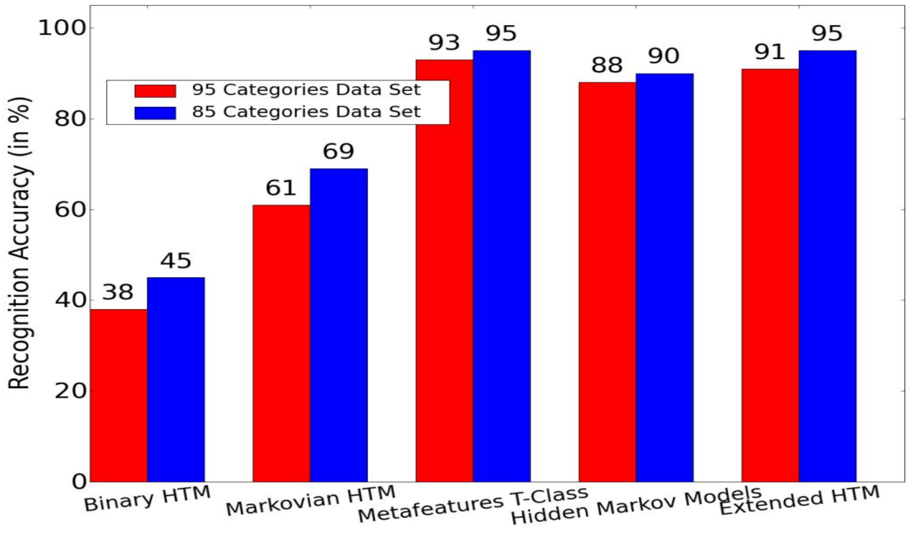


Figure 7.19: **Performance with trimmed data from confusion matrix.** Each method was run without the 10 classes that generated the highest misclassification rate in a confusion matrix. Of the top 3 performing methods (Tclass, HMM and Expanded HTM), performance improved for all on the trimmed data set but more significantly for our extended HTM methodology.

as SLR. It is important to emphasize that SLR performance is highly dependent on the specific language to recognize, the amount of pre-processing performed on input data, the usage of language models, the type and amount of input data, the number of training instances and the size of the vocabulary. Our approach does not make use of language/context models, nor manually defined features to be extracted from the input vectors and it uses lower dimensional input vectors and fewer training instances than most methods used in the literature. Hence, the extended HTM constitutes a valid alternative to traditional HMM methods used in SLR.

Chapter 8

HTMs in the Recognition of Gaze Gestures Offline

8.1 Introduction

In this chapter we perform a study on the usage of traditional HTMs in the off-line recognition of gaze gestures [Rozado et al., 2011b]. This serves the purposes of a preliminary exploration for this input modality as an innovative form of HCI.

The usage of predefined gestures in human-computer interaction, HCI, often employs the hands, head or mouse. The arrival of smartphones and tabletop computers with often touch sensitive surfaces as their only input modality has prompted a recent interest in the subject of gestures for HCI purposes.

Video-based gaze tracking systems can determine where a user is looking at on a screen. Gaze tracking is a very convenient technology for pointing but problematic when trying to distinguish whether the user looks at an object to examine it or to interact with it. This is known as the Midas touch problem, and it highlights the need for additional gaze interaction methods beyond dwell time selections [Mollenbach et al., 2009]. Gaze gestures hold great potential in HCI due to the fast nature of eye saccadic movements, and its robustness to inaccuracy problems, calibration shifts and the Midas problem. The main concern with gaze gestures is the accidental detection of a gaze gesture during normal gaze activity.

The use of gaze gestures in HCI is a relatively new concept and the amount of research done on it is rather limited [Drewes and Schmidt, 2007]. Gaze gestures can be employed by people with severe disabilities, who use gaze as a mono-modal input in their HCI. Gaze gestures can also provide an additional input channel in multi-modal interaction paradigms providing a new venue of interaction with small screen devices such as smartphones or in scenarios where traditional interaction methods are out of reach such as media center devices or surgery rooms. In this work, we have created an in-house data set of 50 gaze gestures and used the neuroinspired Bayesian pattern recognition paradigm known as Hierarchical Temporal Memory, HTM, to learn them. HTMs are appropriate for this problem due to their robustness to noise and their ability to analyze patterns with a multi-dimensional structure. The temporal structure of gaze gestures that unfolds over time requires an appropriate temporal codification for HTMs to properly perform inference, hence, we analyze the impact of different temporal codifications on performance.

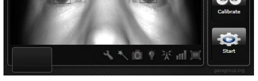
8.2 Eye Tracking and Gaze Gestures

Gaze tracking video-oculography determines a person Point of Regard or PoR (i.e. where a person is looking at) by gathering information from eye position and movements [San Agustin et al., 2010]. Infrared illumination is used to improve iris to pupil contrast and to create a reflection on the cornea, or glint. Due to the spherical shape of the eyeball, this glint remains stationary as the eye moves in its orbit and it is used as a reference point from which to estimate gaze direction. This is done by calculating the vector distance from the corneal reflection and the center of the pupil. Video-oculography is limited by some optical and anatomical constraints and as of yet its maximum accuracy is limited to about 0.5°.

We define a gaze gesture as an ordered sequence of gaze positions over time. Different conceptualizations of gaze gestures exist. Gaze gestures can be relative, i.e. they can be performed anywhere on the screen, or absolute, requiring the user to direct gaze to a sequence of absolute positions on the screen. Due to the limited accuracy of eye tracking technology, fine discrimination between close points on the screen is often not possible. This limitation and the discomfort that continuous micro-movements generate on users, advocates the merits of absolute gaze gestures. Here, we consider a modality of gaze gestures consisting on gliding the gaze along a predefined path on the screen using microsaccadic gaze movements and employing the cursor position as feedback to ensure that the path is followed correctly.

8.3 Experimental Setup

Gaze data acquisition was carried out using the ITU Gaze Tracker [San Agustin et al., 2010] software in a remote setup. We used a Sandberg webcam with no infrared filter, a 16mm lens and two infrared lamps. Image resolution was set to 640×480 pixels, and the frame rate was 30 fps. The distance from the eye to the camera was approximately 60 cm. A filter algorithm was employed on the raw



gaze data to smooth out microsaccades and involuntary jerks while maintaining an acceptable real-time latency. The gaze accuracy achieved with the setup was about 1.5°.

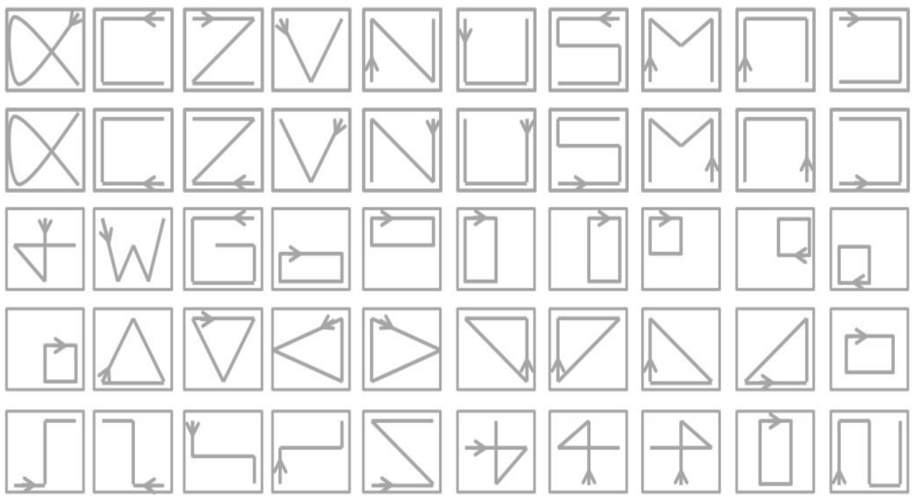


Figure 8.1: **Gestures set.** Set of gestures employed in the user study to evaluate performance. Arrowheads indicate the starting point of a gesture.

Gaze gestures data for training the HTM networks was generated with no black or special purpose background, nor markers to attract or better position the gaze in specific coordinates of the screen. The data set was designed trying to minimize the spatial overlap by maximizing orthogonality. Yet, some gestures were intentionally designed with complete overlap, but different start/end points, to illustrate the importance of temporal coding. 30 instances for each of the 50 categories in the data set, Figure 8.1, were generated by a user experienced with eye-tracking. Test data to measure the performance of HTM inference was gathered by 5 test subjects. All of them were male, regular computer users, not familiar with eye tracking and with ages ranging from 25 to 59 years. Participants were instructed to perform the data set in Figure 8.1 as fast and accurately as possible. One participant with no prior experience on eye-tracking repeated the task over 5 blocks to study learning effects. After completing the experiments, participants filled out a short questionnaire rating speed, accuracy, fatigue and ease of use of their experience with gaze gesture as a form of HCI.

We used Numenta's Nupic package (v1.7.1) [George and Hawkins, 2009] to construct and train the HTM networks. The raw gaze data, consisting on a time series of (x, y) coordinates, was transformed into a *m x n* matrix representing the screen on which the gaze gesture was performed. The matrix was initially filled with 0s but those areas of the screen/matrix scanned by the gaze during the performance of a gaze gesture were assigned a number codifying its temporal structure, see Figure 8.2. We used three types of temporal codification:

- No temporal codification.

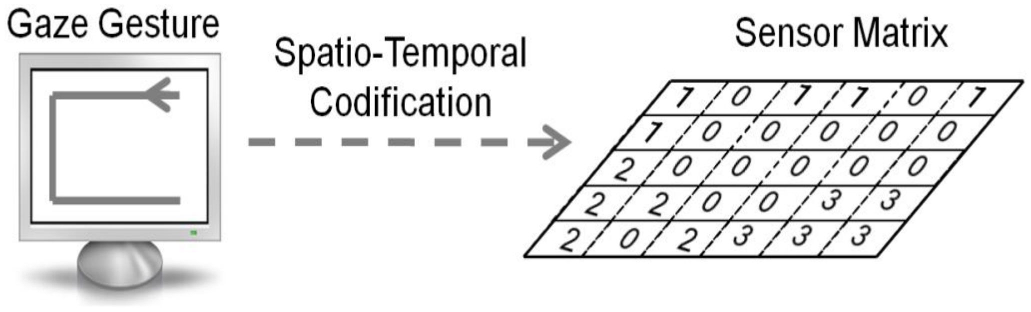


Figure 8.2: **Gaze gestures data codification**. The data structure fed to the HTM network consisted of a matrix codifying the spatio-temporal characteristics of a gaze gesture on the screen. In the type of temporal coding illustrated in this Figure, 0 codifies that gaze did not hoover over that area of the screen during the performance of this particular gaze gesture, "c", or that the gaze tracker did not fetch gaze data over that area. A 1 codifies the spatial positions of gaze during the beginning of the gesture, a 2 codifies the spatial positions hoovered by gaze during the middle instants of the gesture performance and a 3 indicates the positions over which gaze hoovered during the final phase of the gesture. The lack of a perfect representation of the idealized gesture in the matrix illustrates the limitations in terms of resolution and accuracy of gaze tracking technology and the presence of noise in the data.

- Temporal codification in seconds by storing in the corresponding matrix element the second in which gaze was determined to passed over a particular area of the screen.
- Three Temporal Stages codification by dividing the total time employed during performance of a gesture in 3 slices (beginning, middle, end) and assigning correspondingly to the matrix the numbers 1, 2 or 3 depending when the gaze was determined to hovered over the corresponding area.

8.4 Results

We studied the amount of time needed to complete a gaze gesture by plotting on a histogram, Figure 8.3, the distribution of time lengths per gesture required by users while performing the set of 50 gaze gestures. Inference accuracy for gaze gestures recognition varied for different network topologies with a two layer network showing the best performance in Figure 8.4.

Several data representations of the gaze gestures where tried out, see Figure 8.5, on the optimal 2-Layer HTM network determined above. The data representations consisted namely in: no temporal codification, a finely grained temporal codification in seconds and a temporal codification of 3 stages (beginning, middle and end) that turned out to be the optimal one with up to 96% recognition accuracy.

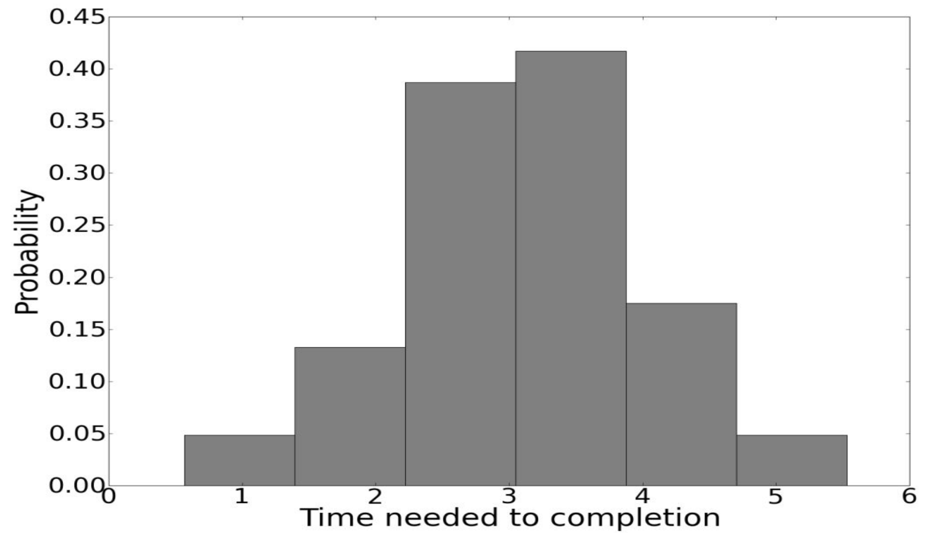


Figure 8.3: **Time per gesture.** Histogram of times per gesture employed by users while performing the gaze gesture set.

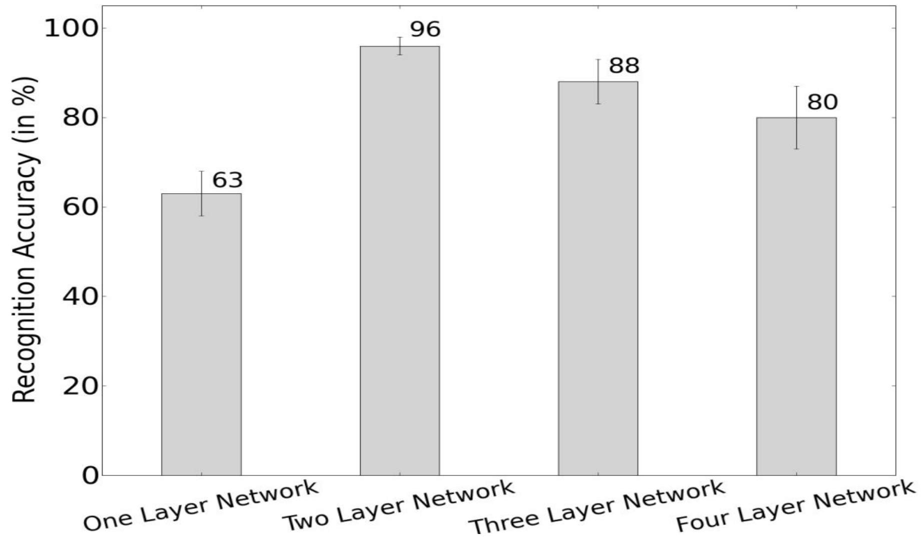


Figure 8.4: **Accuracy performance for different network topologies during inference.** The bars show the percentage of correct classifications achieved by 1, 2, 3 and 4 layer networks.

Experiments were carried out to determine HTM decreasing performance with larger gaze gestures vocabularies, Figure 8.6. The 3 temporal stages data representation proved to be the most resistant to increasing vocabulary size.

HTM performance improved markedly with increasing number of training instances, Figure 8.7. The 3 temporal stages data representation proved to perform better than the others for the whole range of training instances availability.

A single user with no prior experience in gaze tracking repeated the 50 gaze

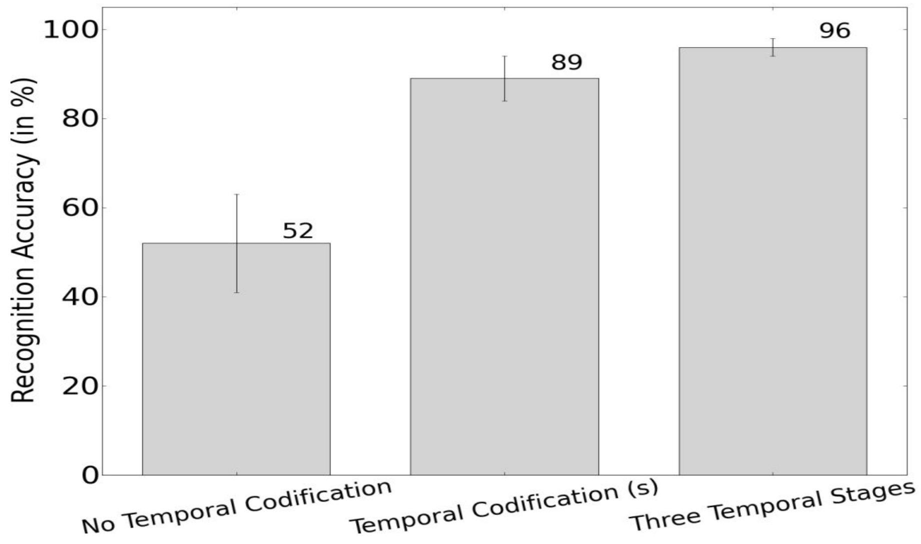


Figure 8.5: **Accuracy performance of different data representations on the optimal 2 layer HTM network.** Namely, no temporal codification, temporal codification in seconds and 3 stages temporal codification.

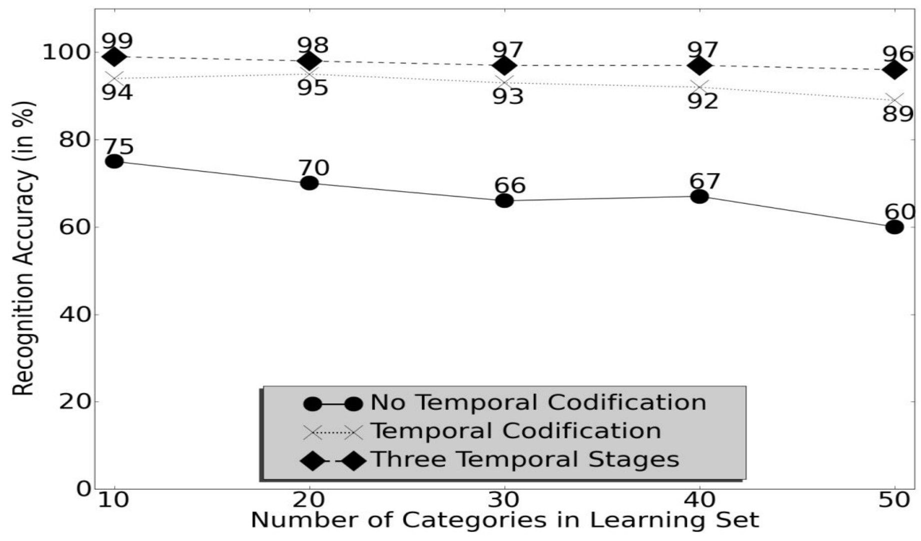


Figure 8.6: **Decreasing performance.** The figure shows the decreasing performance of the optimal 2-layer HTM network using different data representations as the vocabulary size of the gaze gestures set to be learnt increases.

gestures data set over five blocks in order to determine learning effects, see Figure 8.8.

To study gaze gesture recognition in real time, we measured HTM inference scores obtained on gaze data during normal computer use and inference scores obtained when specifically performing inferences over gaze gestures data, Figure 8.9. It is clearly visible in the graph that no inference score appropriate threshold exists

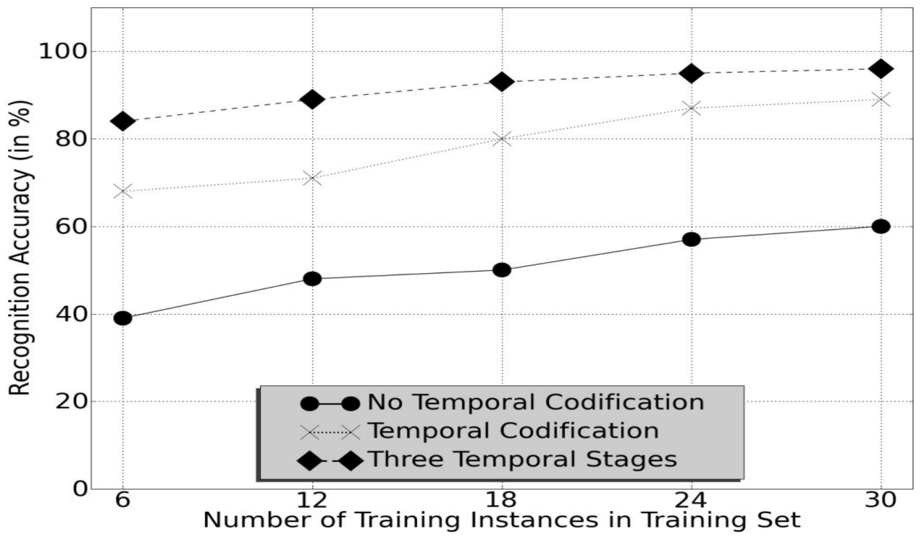


Figure 8.7: **Improving performance.** HTM networks using different data representations improve performance as the number of training instances increases.

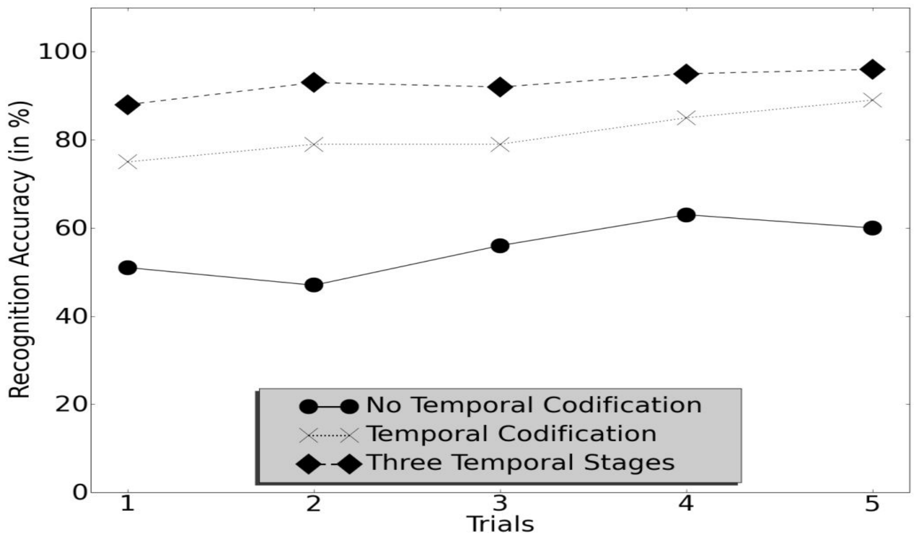


Figure 8.8: **Learning effects.** Recognition accuracy of the 2-layer HTM network using different data representations over 5 blocks for a single user.

to clearly discriminate consciously performed gaze gestures from other types of gaze activity. The overlap between both types of distribution indicates that real time gaze gesture recognition represents a challenge for traditional HTMs.

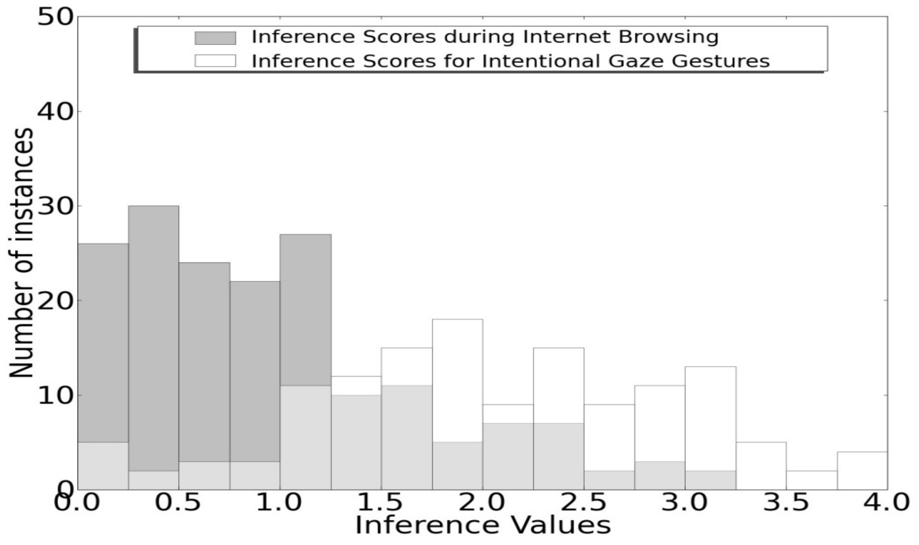


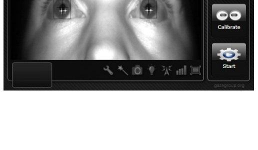
Figure 8.9: **Scores overlap.** HTM inference scores during normal gaze activity while using a computer (dark grey bars distribution) and the HTM scores obtained on consciously performed gaze gestures (white bars distribution). The area of overlap between both types of inference scores is shown in light grey bars.

8.5 Discussion

For optimal performance, HTMs require training instances to be composed of a complete spatial structure at any time instant. This creates a challenge for data structures with a temporal component unfolding over time. To address this issue, data representation and coding become key when designing and training an HTM network.

Gaze gestures patterns as existing in the original data set are not fit to be learned by a traditional HTM network since at any moment in time the complete spatial representation of the pattern is not complete. The original data set consisted of just a temporal series of (x,y) coordinate pairs. The original time-series were transformed into a 2 dimensional matrix containing the on-screen path performed by gaze during the performance of a sign. This data structure captured the complete spatial structure of a gaze gesture. However, a gaze gesture possesses a temporal structure as well. This aspect is critical since the temporal order of a gaze gesture differentiated several gestures with complete overlap in their 2D spatial representation such as the gaze gesture in the 1^{st} row, 2^{nd} column and the gaze gesture in the 2^{nd} column of the 2^{nd} row in Figure 8.1. In order to codify the temporal information of a gesture, several approaches were explored as shown in Figure 8.5.

Time needed to complete gaze gestures is an important design and constrain parameter for gaze-computer interaction. Our experiments show that gaze gestures are fast to complete, Figure 8.3, easy to learn, Figure 8.8, and they do not occupy screen-real state since no visual markers to aid in gaze trajectory were used. Furthermore, in our user study, participants showed through questionnaires satisfaction



with this innovative input modality.

Gaze gestures however present problems of their own, the main one being accidental gesture completion during normal gaze activity. As Figure 8.9 shows, it is a challenge for the HTM algorithm to perfectly partition consciously performed gaze gestures from unintended gaze gestures completed during normal gaze activity while using a computer. Furthermore, gaze gestures also generate a cognitive load on the user which is forced to memorize and reproduce sequences of eye movements to recreate a gesture without an immediate feedback.

Our results suggest that our system can be expanded to a larger vocabulary set with still acceptable recognition performance, see Figure 8.6. Figure 8.7 illustrates the importance of having a large number of training instances for the HTMs to achieve good recognition performance.

A trade-off emerges from our study between complex gaze gestures and simpler ones. Simpler gaze gestures are easy on the user, yet the possibilities of accidental gesture completion are high. Complex gaze gestures decrease the possibility of accidental recognition during normal gaze activity and augment the interaction vocabulary space, imposing however, a cognitive load on the end user.

Our work shows that humans adapt quickly and comfortably to this innovative modality of HCI. The good recognition results achieved by the HTM algorithm and the positive feedback from users, illustrate that using gaze gestures recognized through HTMs constitutes an innovative, easy-to-learn and viable approach to HCI for several environments and device combinations. However, as Figure 8.9 indicates, the marked overlap of scores obtained by normal gaze activity during computer usage and consciously performed gaze gestures, indicates a challenge for real time gaze gesture recognition. An appropriate threshold for gesture recognition in this type of scoring space would never find a satisfactory partitioning threshold for acceptable sensitivity and specificity. In the next 2 chapters, we explore alternative approaches to properly partition the scoring space to discriminate consciously performed gaze gestures from other types of gaze activity. In particular, we show how our extended HTM algorithm is able to reach acceptable sensitivity and specificity for real time gaze gesture recognition in contrast to traditional HTM.

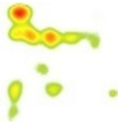
Chapter 9

Gaze Gestures Recognition in Real Time

9.1 Introduction

While the previous chapter focused on off-line recognition of gaze gestures, this chapter focuses on the more challenging problem of gaze gestures recognition in real time using a low cost gaze tracking system [Rozado et al., 2010a]. That is, we aim to distinguish consciously performed gaze gestures from otherwise standard gaze activity during HCI. Furthermore, this chapter also studies different modalities of performing a gaze gesture: gliding and saccadic.

The usage of predefined gestures in human-machine interaction is not new. The idea of a gesture consists on employing body motion, generally the hands and/or the head, to convey information [Drewes, 2010]. The emergence of tabletop computers and smartphones with touch sensitive surfaces as their only input modality has spurred a growing interest in the subject of gestures in human-computer interaction or HCI [Drewes and Schmidt, 2007; Kapuscinski and Wysocki, 2009]. Gestures can be simple and/or culturally independent, such as pointing with the index finger or the head, or quite elaborate, such as handwriting or sign-language for deaf/mute people [Rozado et al., 2010b]. Common gestures usually employed in human-machine interaction are often done with a pen. The mouse can also be employed to perform 2D sequences of movements, such as the gestures employed by the Firefox and



Opera web browsers. The information transmitted with a gesture can be mapped to a specific letter, command or macro. Using gestures in human-machine interaction involves learning a set of gestures and their semantics.

Gaze movements are employed in human-human interaction in the form of rolling the eyes to express disagreement or indicating direction with a fast movement of the eyes along with a slight head movement [Drewes, 2010]. In the present work, we explore the feasibility of using gaze gestures as an input modality to control a computer by carrying out a sequence of gaze positions. In the following sections, we highlight the potential of the gaze gestures modalities and recognition algorithms under study for human-machine interaction in the context of gaze tracking.

Gaze tracking systems using video-based hardware and algorithms can determine the point of regard (PoR) on a screen for a certain user [Duchowski, 2007; Hennessey et al., 2008]. Usually, the PoR is employed as a pointing device, that is, as a substitute of the mouse [Jacob, 1991]. This communication channel has been shown to enable users with severe motor disabilities to interact with a computer by using gaze alone for pointing and even selecting objects on the screen using dwell time activation, albeit slower and more error prone than with traditional input devices [San Agustin, 2010]. Although gaze tracking is a very convenient technology for pointing [Jacob, 1991; San Agustin et al., 2010], problems arise when trying to distinguish whether the user looks at an object to examine it or to interact with it [Mollenbach et al., 2009]. This is known as the Midas touch problem [Jacob, 1991] and it highlights the need for additional forms of interaction through gaze. Ideally, selection should be performed with an external switch, decoupling pointing and selections, in the same way as it is done with a mouse. However, people with severe disabilities or users in certain environments are often unable to use an external selection device, and must therefore rely on gaze-only activation techniques. Among these, blinking and dwell-time activations are the most common. In the former, a click command is issued when the system detects a blink with a predefined duration, while in the latter the click command is issued when a fixation longer than a predefined threshold is detected. Gaze gestures emerge as potential candidates to bridge the gap between pointing and selection in gaze interaction systems by circumventing the Midas touch problem.

A gaze gesture can be defined as a sequence of strokes. A stroke is an intentional movement between two fixation points. Different patterns of strokes define different gestures [Mollenbach, 2010]. Different gestures can then be mapped to issue different commands for human-machine interaction purposes. Several types and modalities of gaze gestures can be defined as we describe in the Methodology section.

Gaze gestures, when used in combination with other input devices, can provide an additional input channel that augments and enhances the interaction with a computer. This multi-modal interaction paradigm would not only benefit people with disabilities, but it could also provide a new venue of interaction with small screen size devices such as smartphones or tablets. Gaze gestures can be employed by people with severe disabilities, who use gaze as a mono-modal input in their human-computer interactions [Vickers et al., 2010; Reddy and Basir, 2010]. It could also be beneficial in scenarios where traditional interaction methods such as key-

board or mouse are out of reach or inconvenient to use. Gaze gestures differentiate themselves from traditional control commands used in gaze-computer interaction such as fixations and dwell times. Due to the fast nature of the saccadic movements involved in gaze gestures, this selection technique can potentially be faster and less stressful than dwell time. Moreover, gaze gestures can also be very robust to inaccuracy problems and calibration shifts. However, there can be an overlap between natural search patterns and the gaze patterns of a gesture, which could lead to false positives, i.e. the accidental detection of an involuntary gaze gesture. For gaze interaction purposes, it is desirable to minimize unintended gaze gestures recognition [Mollenbach, 2010]. Increasing the complexity of the gaze gesture, i.e. the number of strokes, minimizes the overlap between natural gaze patterns and consciously performed gaze gestures and increases the interaction vocabulary, but it also introduces a greater cognitive complexity and physiological load on the end user.

The amount of research done on gaze gestures is rather limited since gaze gestures are a relatively new concept. Authors in [Qvarfordt and Zhai, 2005] used gaze gestures in a dialog system to facilitate communication. Their work studied gaze patterns in human-human interaction and used the results to mediate a human-machine dialog. It is important to notice that users did not learn a set of gaze gestures to operate the system nor were they aware they were performing gaze gestures. Some research has been done on using gaze gestures for gaze-based input of characters [Wobbrock et al., 2007; Bee and André, 2008; Isokoski, 2000]. The work from [Mollenbach, 2010] constitutes one of the most elaborate and comprehensive analysis of gaze gestures to date exploring the interaction possibilities of single stroke gaze gestures. The work from [Isokoski, 2000] represents one of the first attempts to employ gaze gestures to issue commands, in particular to enter characters. Here, the author proposes an eye-typing system that requires the user to look at off-screen targets in a certain order to input characters. The approaches in [Wobbrock et al., 2007] and [Bee and André, 2008] allow an experienced user to enter characters at up to 7.99 words per minute. Increasing the set of gaze gestures so that each gesture represents one symbol could potentially increase the text-entry performance. However, a study by Wobbrock et al. in [Wobbrock et al., 2008] demonstrated that a smaller set of symbols is preferable. In their work, they proposed EyeWrite, a gaze-based text-entry system where each letter is mapped to a gesture performed looking at the four corners of the screen. The average typing speed was only around 2 words per minute, and the complexity of the alphabet was reported to be too high. Furthermore, recognizing an extensive symbol alphabet is a challenging computational problem in the realm of pattern recognition.

In this work, we present and evaluate two alternative modalities of gaze gestures execution and two algorithmic methods of gaze gestures detection:

- Gaze gestures consisting on gliding the gaze along a pre-defined path and detected with traditional Hierarchical Temporal Memory (HTM) networks.
- Gaze gestures consisting on performing a pre-defined sequence of saccadic movements and detected by the Needleman-Wunsch algorithm for sequence

alignment.

The reason to use different algorithms for the recognition of each gaze gesture modality was the fact that each modality of performing the gestures generated markedly different data structures that required specialized algorithms for detection. The gliding gaze gestures generated 2 dimensional spatial patterns that develop over time while the saccadic gestures generated a sequence of spatial positions on the screen.

Our results suggest that natural eye movements during computer usage can be robustly separated from intentional gaze gestures when using the appropriate gaze gesture modality (saccadic gestures) and pattern recognition algorithm (sequence alignment with the Needleman-Wunsch algorithm). We also learnt that users adapt quickly and comfortably to this somehow counter-intuitive modality of human-machine interaction. Therefore, saccadic gaze gestures recognized through sequence alignment constitutes an innovative, robust, easy-to-learn and viable approach to human-machine interaction for several environments and device combinations.

9.2 Methodology

We have created a gesture recognition engine conceptually placed between the gaze tracker and the application a user would wish to control. Figure 9.1 illustrates the architecture of our system. In the following subsections, we elaborate on the methodology used in our experiments.

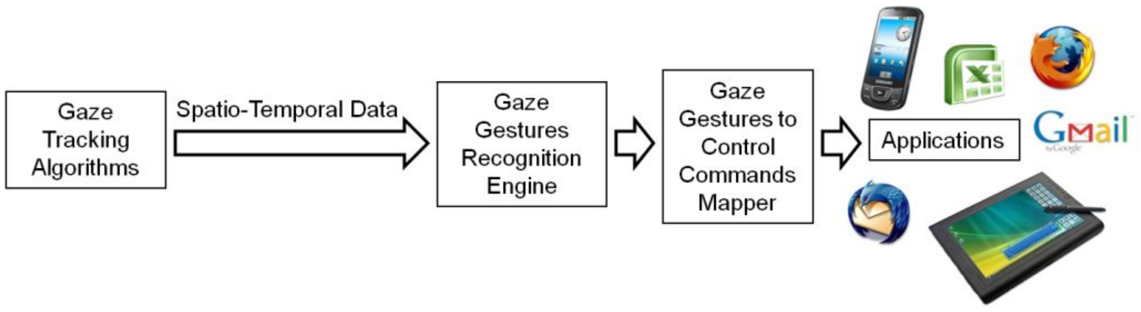

Gaze Tracking Algorithms
Spatio-Temporal Data
Gaze Gestures Recognition Engine
Gaze Gestures to Control Commands Mapper
Applications
Gmail

Figure 9.1: **System architecture.** For application purposes, our gaze gesture recognition engine is conceptually placed between the Gaze Tracker engine and the applications that we wish to control. The gesture recognition engine receives the gaze tracking data through a customized client and carries out recognition to discriminate between gaze gestures performed on purpose by the user and otherwise normal gaze activity. Finally, the gesture recognition system maps a recognized gesture to a specific key combination, script or macro that is then sent to a particular application.

9.2.1 Eye tracking

The eyes represent the main organ that humans employ to perceive and interact with the world. We use our eyes to obtain information about our surroundings by directing our gaze to objects of interest. When something attracts our attention, we position our gaze on it, thus performing a *fixation*. A fixation usually has a duration of at least 100 to 150 ms. The fast eye movements that occur between fixations are known as *saccades*, and they are used to reposition the eye so that the object of interest is projected onto the fovea. The direction of gaze thus reflects the focus of our *attention*. When we interact with the world, we look at the objects to examine their characteristics. For instance, before grabbing a cup of coffee we will glance at it and then we will move the arm to grab it. Therefore, eye gaze also provides a sense of our *intention*.

A video-based gaze tracking system seeks to find where a person is looking, i.e. the Point of Regard (PoR), by means of information obtained from the eye by one or more cameras that record the user's eye region. Most systems employ infrared illumination to improve the quality of the image and to estimate gaze. Since infrared light is invisible for the human eye, it is not distracting nor annoying to the user.

Depending on the hardware configuration of the different components, gaze tracking systems can be classified as either *remote* or *head-mounted*. In remote systems, the camera and the light sources are placed at a distance from the user, normally around the computer screen, whereas in head-mounted systems the components are placed on the user's head, usually mounted on a helmet or a pair of safety glasses [Li et al., 2006; San Agustin et al., 2010]. In our experiments we use a head mounted configuration using the low cost ITU gaze tracker [San Agustin et al., 2010].

Interactive gaze-controlled applications use the user's eye gaze as an input to control an interface, employing a gaze tracking system that detects and tracks the point of regard (PoR) of the user over time. This type of gaze-based applications allow the user to look around and select objects on the screen, and the interface reacts in real time upon the user's eye movements. By substituting or complementing the mouse, the user's PoR can be employed to control a graphical user interface. Ideally, selections should be performed with an external switch, therefore decoupling pointing and selections, in the same way as it is done in a mouse. However, people with severe disabilities are often unable to use an external selection device, and must therefore rely on gaze-only activation techniques. Among these, blinking and dwell-time activations are the most common. In the former, a click command is issued when the system detects a blink with a duration in a predefined range (usually longer than 1 second), while in the latter the click command is issued when a fixation longer than a predefined threshold (usually in the range of 300 to 3000 milliseconds) is detected.

9.2.2 Gaze Gestures

A gaze gesture is defined in [Vickers et al., 2010] as "a definable pattern of eye movements performed within a limited time period, which may or may not be con-

strained to a particular range or area, which can be identified in real time, and used to signify a particular command or intent".

Different conceptualizations of gaze gestures exist. Gaze gestures can be relative, i.e. they can be performed anywhere on the screen, or absolute, requiring the user to direct gaze to a sequence of absolute positions on the screen. Due to the limited accuracy of eye tracking technology, fine discrimination between close points on the screen is often not possible. This technological limitation and the fact that it is uncomfortable for users to accurately generate sequences of micro-movements, advocates the merits of performing gaze gestures by using absolute positions on the screen. Gaze gestures can also be classified as single stroke, when composed of just one saccadic movement, or complex, when they involve more elaborate paths [Mollenbach et al., 2010]. The main advantage of simple gaze gestures is that they are easy to memorize and perform by users. Yet they markedly overlap with normal gaze activity when interacting with a computer, thus limiting their applicability as an input channel since normal inspection and navigation patterns might accidentally be confused with gaze gestures. Complex gaze gestures have the advantage of greatly increasing the vocabulary size of gaze interaction. However, complex gaze gestures generate a cognitive and physiological load on the user. Cognitively it is difficult for users to remember a large set of complex gestures, and physiologically it is tiring and challenging to complete them [Mollenbach, 2010]. Finding the right trade-off between simple and complex gaze gestures is therefore paramount to successfully use gaze gestures as an input channel. Furthermore, gaze gestures can be classified as saccadic gaze gestures, when the movements between fixation points are saccadic (ballistic) or gliding gaze gestures, where the gaze is glided along the whole trajectory of the gesture. In this work, we used absolute saccadic gaze gestures of intermediate complexity consisting on performing a sequence of saccades (strokes) between different areas of the screen.

Two different modalities of performing a gaze gesture have been considered in this paper: gliding the gaze along a predefined path, and performing saccades (strokes) between a sequence of different areas of the screen. Gliding the gaze along a path without performing a smooth pursuit of a moving target is physiologically impossible [Robinson, 1965]. Users can instead perform short saccades following the path of the gesture, employing the cursor position as feedback to ensure that the path is correct.

Due to the different nature of the two types of gaze gesture methods under analysis, two different algorithms for recognition of gaze gestures were used. Each algorithm was matched in its learning and inference capabilities to the particular type of gaze gesture modality (gliding or saccadic) it had to recognize. We used HTM networks to detect the gliding gaze gestures and sequence alignment to detect the saccadic gaze gestures.

9.2.3 Hierarchical Temporal Memory

The algorithm employed in this work to detect the type of gestures performed by gliding gaze along a predefined path was Hierarchical Temporal Memory. HTMs

are specifically design to learn models of spatio-temporal data. Since a gaze gesture consists of a 2 dimensional shape constructed over time, HTMs were considered suitable for recognition purposes.

For the purpose of training the HTM network, the raw data was transformed into an $m \times n$ two-dimensional matrix representing the screen on which the gaze gesture was performed. The matrix was initially filled with 0s and represented a linear mapping with the screen. To codify the spatial structure of a gesture, those areas of the screen/matrix scanned by gaze during the performance of a gesture were assigned a number. This number codified the time dimension of the gaze gesture. To codify the temporal structure of a sign, we divided evenly the total time employed in performing the gesture in 3 slices: beginning, middle, and end, and assign correspondingly the numbers 1, 2, 3 to the coordinates of the matrix over which the gaze hovered at a particular time instant as shown in Figure 9.2. Hence, this data structured contained both the spatial representation of the gaze gesture and its temporal structure. During real time recognition, the HTM would received a continuous updated representation of this matrix and perform inference every 500 milliseconds over the activity occurred during the last 5000 milliseconds.

We created a three layers HTM network composed of 5 nodes and 30 sensor nodes as illustrated in Figure 9.2. The bottom layer nodes received their input data from the 30 channels of input data coming from the 2D matrix. Several topologies were tried out varying the number of layers, nodes, fan-in edges, fan-out edges, spatial-pooler specificity coefficients and clustering coefficients for the nodes. The topology shown in Figure 9.2 was selected by manual curation after it proved to be the one that optimized the performance of the network, during inference, over several alternative network designs.

9.2.4 Sequence Alignment and Dynamic Programming

For the recognition of gaze gestures performed by a sequence of saccadic movements over different areas of the screen, we used a measure of similarity between sequences. Sequence similarity was calculated using the Needleman-Wunsch algorithm of dynamic programming for sequence alignment.

Sequence alignment is a method used to identify the similarity between two or more sequences. Sequence alignment is commonly used in analysis of sequences present in natural language, financial data and biological sequences such as DNA, RNA or protein sequences from which to infer functional, structural or evolutionary relationships between the sequences.

In sequence alignment, the degree of similarity between two sequences can be inferred from the number of similar elements occupying the same position in a sequence (matches), the number of dissimilar elements (mismatches) and the number of gaps needed to make the sequences align. The overall score can be interpreted as a measure of how similar both sequences are. Aligned sequences can be represented as rows within a matrix. Gaps can be inserted between elements to allow similar elements to be aligned in successive columns.

Sequence alignment can be divided into two categories: global alignments and

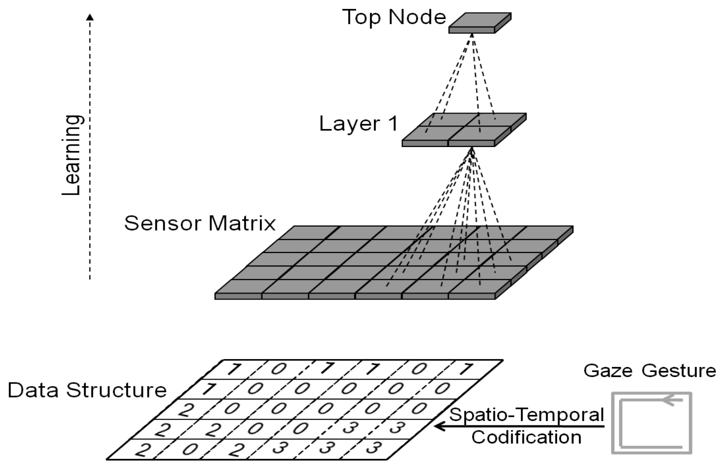


Figure 9.2: **HTM topology for recognizing gliding gestures.** Bottom level nodes are fed with input from a matrix codifying the spatio-temporal structure of a gaze gesture. A 0 element codifies that the gaze was not determined to hoover over that area of the screen during the performance of a gaze gesture, “c”. A 1, 2 or 3 codifies the spatial positions of gaze during the beginning, middle and end of a gesture. The lack of a perfect representation of the idealized gesture in the matrix illustrates eye tracking’s accuracy limitations and the presence of noise.

local alignments. Global alignment forces the alignment to span the entire length of the sequences in a search for global optimization, while local alignment searches for regions of similarity within longer sequences that overall may not be very similar.

Dynamic programming is an optimization algorithm that can produce high-quality sequence alignment, guaranteeing to find the optimal alignment through an exhaustive search of the alignment space. Dynamic programming is an NP hard algorithm. Heuristic algorithms exist that, while not guaranteeing finding the best possible alignment, are more efficient in terms of the computational resources that they require.

Sequence alignments can be represented both graphically and in text format. Generally, sequences are written in rows arranged so that aligned elements appear in successive columns. In text formats, aligned columns containing identical or similar elements are indicated with a system of symbols indicating matches, mismatches and gaps.

A widely used global alignment technique is the Needleman-Wunsch algorithm [Needleman and Wunsch, 1970], based on dynamic programming. Its implementation consists on the usage of a substitution matrix to assign scores to elements matches, mismatches and gaps (Equation 9.1). A scoring scheme is used by assigning a positive score to a match, a negative penalty score for a mismatch and a smaller penalty for a gap. A standard extension to linear gaps cost is to modify the algorithm and assign different scores for opening and extending gaps. Figure 9.3

shows the results of pairwise global alignment between two artificial gaze gestures, according to the Needleman-Wunsch algorithm.

Equation 9.1 is the mathematical formalism of the Needleman-Wunsch algorithm: i, j are the indexes of rows and columns in the matrix, g is the gap penalty, and $s(i, j)$ is the match/mismatch score for residues i and j. Although dynamic programming is extensible to more than two sequences, its computational costs make it unsuitable for multiple sequence alignment, with heuristic algorithms being the appropriate choice. For short sequences and pairwise linear alignment, it remains a valid choice since it guarantees to find the best possible alignment. Since we were performing short, pairwise linear alignments, the Needleman-Wunsch algorithm seemed like an appropriate choice. Figure 9.3 shows a sequence alignment calculated with the Needleman-Wunsch algorithm for two sequences of gaze gestures.

$$|D(i,j)| = max \begin{cases} D(i-1,j-1) + s(x_i, y_j) \\ D(i-1,j) + g \\ D(i,j-1) + g \end{cases} \tag{9.1}$$

The score at the bottom right of the matrix in Figure 9.3 indicates the degree of similarity between both sequences for the global alignment. The alignment can be traced back (highlighted in bold in Figure 9.3) using a traceback matrix, T, starting from the bottom right element and traveling backwards according to:

$$T(i,j) = argmax \begin{cases} D(i-1,j-1) \\ D(i-1,j) \\ D(i,j-1) \end{cases} \tag{9.2}$$

In order to carry out the recognition of gaze gestures based on sequences of saccadic gaze gestures, our gesture recognition engine carried out the following procedure. The computer monitor was divided into 9 different spatial areas, named from A to I, as represented in Figure 9.4. The recognition engine constantly updated which area of the screen the user was looking at and kept a pipe of the previous 6 areas. This pipe of spatial areas over which the user's gaze hoovered was constantly being updated every time a change of area was detected. During normal use of the computer, this stream or pipe containing the present and previous 6 areas over which gaze hoovered was aligned against the set of predefined gaze gestures in order to find degrees of similarity above a threshold that would indicate that a conscious gaze gesture had just been performed.

9.3 User Study

A user study consisting of two parts was carried out to test the performance of the two different gesture modalities and the two methods used to detect them. First, a pilot experiment with 7 participants was used to gather preliminary results about the gaze gesture modality that provided the best performance in terms of accuracy, speed and low degree of false positives. A second experiment with 20

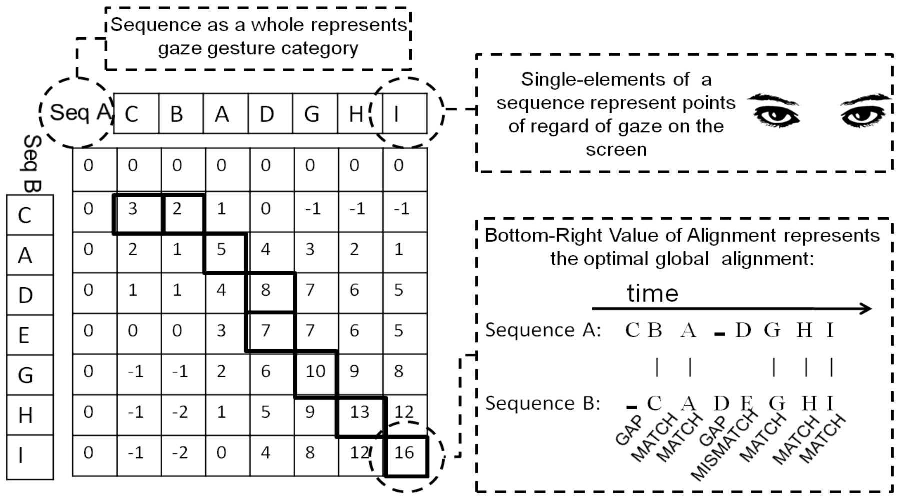

Sequence as a whole represents gaze gesture category
Seq A
C B A D G H I
Seq B
C A D E G H I
0 0 0 0 0 0 0 0
0 3 2 1 0 -1 -1 -1
0 2 1 5 4 3 2 1
0 1 1 4 8 7 6 5
0 0 0 3 7 7 6 5
0 -1 -1 2 6 10 9 8
0 -1 -2 1 5 9 13 12
0 -1 -2 0 4 8 12 16
Single-elements of a sequence represent points of regard of gaze on the screen
Bottom-Right Value of Alignment represents the optimal global alignment:
time
Sequence A: C B A - D G H I
Sequence B: - C A D E G H I
GAP
MATCH
MATCH
GAP
MISMATCH
MATCH
MATCH
MATCH

Figure 9.3: **Dynamic programming matrix used for the alignment of saccadic gaze gestures.** Two sequences conforming a gaze gestures: Seq A and Seq B are aligned using the following scoring scheme: match=3, mismatch=−1, gap=−1. Each element of the sequences represents an area of the screen. The top global alignment can be found in the bottom right box of the matrix. The actual alignment is show in the bottom right of the Figure.

participants was then carried out in order to measure the performance differences and user preferences of saccadic gaze gestures performed with or without dwell time to indicate the beginning and end of a gesture.

9.3.1 Pilot Experiment

Experimental Setup

Several participants took part in the pilot experiment. All of them were male and were familiar with gaze tracking technology. The experiments were carried out using the open source ITU Gaze Tracker [San Agustin et al., 2010] in a head-mounted setup to perform gaze tracking as the one shown in Figure 9.5. The eye image data was captured using an off-the-shelf webcam (Sandberg Nightcam 2), which was mounted on a cap. Contrary to most webcams in the market, this webcam model does not include an infrared filter, thus permitting the use of the infrared spectrum. The camera contains a set of 6 infrared LEDs that improve pupil-to-iris contrast. Its resolution was set to 640×480 pixels, and the frame rate was set to a value of 30 frames per second. In our head-mounted setup, the distance from the eye to the camera was approximately 5 cm. The gaze accuracy of the setup was about 1°.

The ITU Gaze Tracker streamed all calculated gaze coordinates through a TCP/IP server. This raw data was accessed by an in-house developed client. The gaze gestures data used for training the HTM networks was stored in text files.

During the generation of the data set and during the evaluation of the ability of the recognition algorithms to correctly classify unseen instances, we did not use a

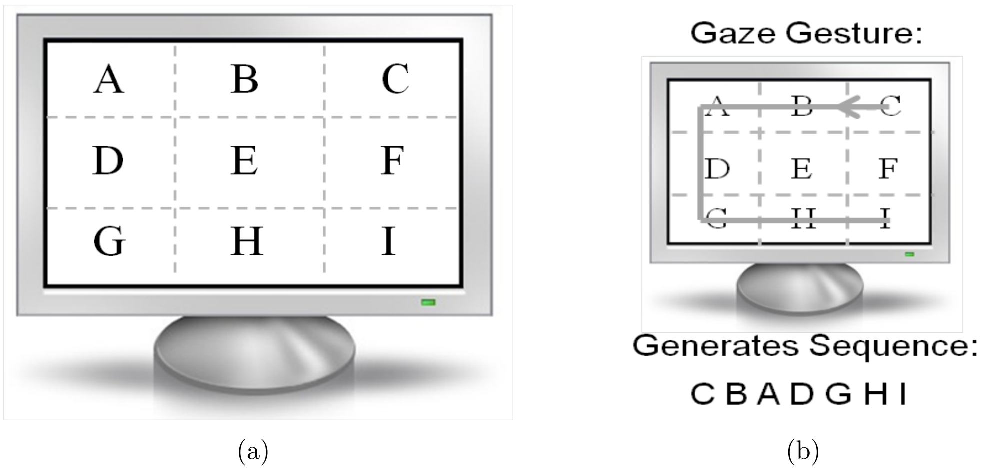


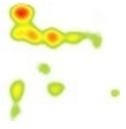
Figure 9.4: **Computer screen and gaze gestures sequences.** Panel *a* shows how the screen real state was partitioned in 9 areas and given letter names from "A" to "I" for the saccadic gestures modality and the sequence alignment method for detection. Panel *b* shows how a predefined sequence of areas over which a user's gaze hoovered while performing a gaze gesture was monitored and transformed into a sequence.

black or special purpose background, nor markers to attract or better position the gaze in specific coordinates of the screen. The performance of the gaze gesture was always measured over a normal desktop, with open windows, panels, programs, etc. This was done to recreate the possible scenario in which potential applications of gaze gestures would take place. It also underlines one of the intrinsic advantages of gaze gestures: they do not take screen real estate.

The objective of the pilot experiment was to obtain preliminary results on the performance of the two different gaze gestures modalities, i.e. *gliding gestures*, where HTMs were used for detection, and *saccadic gestures*, where sequence alignment with the dynamic programming Needleman-Wunsch algorithm was employed for detection. For each gaze gesture modality, we measured the performance of two different techniques: in one, users had to perform a dwell in the beginning and in the end of the gaze gesture; with the other technique users had to perform the gesture without the need to perform dwell. The dwell time was set to 800 ms. The user was notified when a dwell was detected by audio feedback. In total, four different experimental conditions were studied in the pilot experiment: *gliding without dwell*, *gliding with dwell*, *saccadic without dwell*, and *saccadic with dwell*.

Participants completed 3 different tasks, an *accuracy task*, a *browsing task*, and a *velocity task*. In the accuracy task, users had to perform the sequence of 10 gaze gestures shown in Figure 9.6 for each experimental condition.

In the accuracy task, participants were instructed to complete the gestures as fast and as accurately as possible. The user was notified with audio feedback every time the system detected a gesture, regardless of whether the detected gesture was



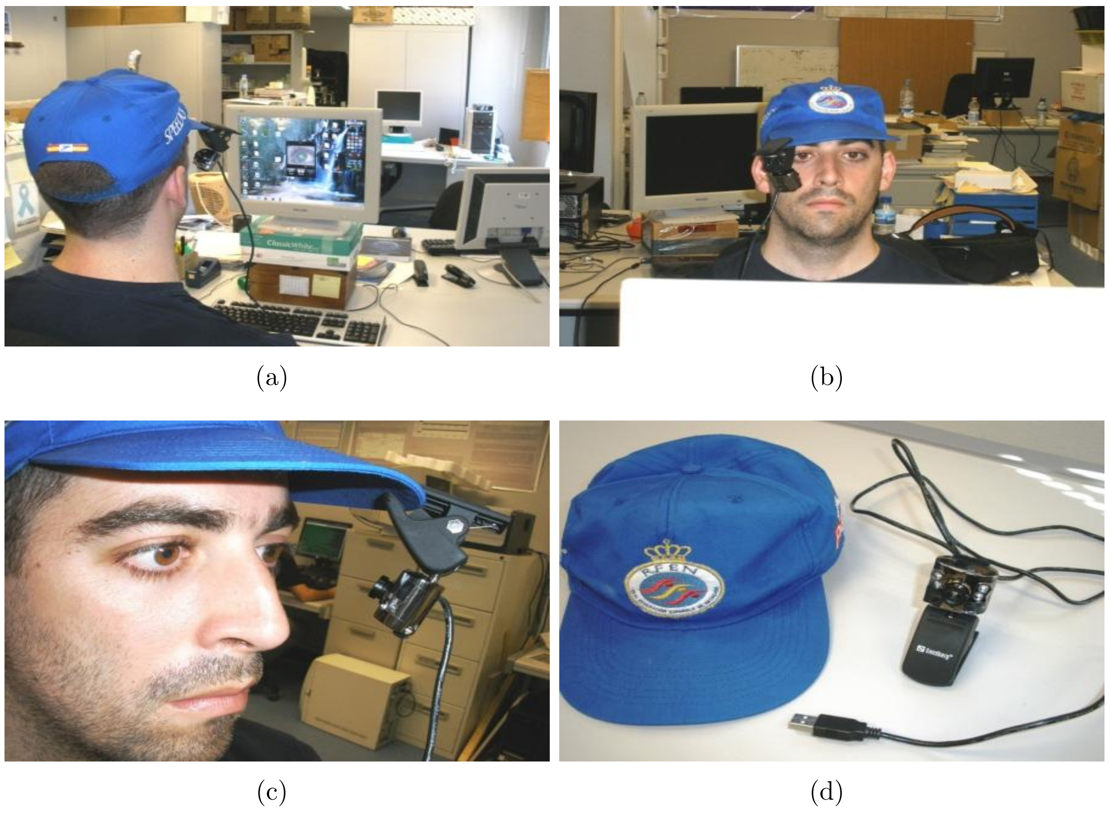


Figure 9.5: **Experimental setup.** Panels *a*, *b*, *c* and *d* show the experimental setup and hardware used to generate gaze gestures and to evaluate the performance of different gaze gestures modalities and recognition algorithms.

correct or not. When a gesture was detected, the user had to proceed to perform the next gesture. For each trial, we measured the *accuracy* if terms of the percentage of correct gestures detected from the total.

The browsing task required participants to browse the Internet during 5 minutes in each of the 4 experimental conditions. During this time, the number of *involuntary* gaze gestures detected, i.e. false positives, was measured.

To obtain a measure of the time per gesture, participants carried out a velocity task in which they were instructed to perform a gesture indicating with a switch the beginning and end of the gesture. The time of the gesture was considered to be the interval between both switch activations. We refer to that quantity as Time per gesture or TPG. Three gestures were chosen for this task, one with three strokes (*c*), one with 4 strokes (*j*) and one with 5 strokes (*hook*), see Figure 9.6.

Results of the Pilot Experiment

The pilot experiment consisted on 3 different tasks: accuracy, browsing, and velocity. Figure 9.7 shows the results obtained for the accuracy task. The gliding modality with no dwell and using HTMs for detection obtained an average accuracy of 62%.

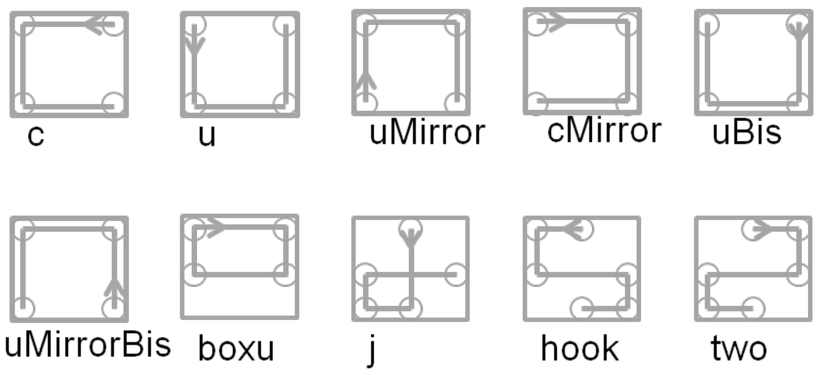


Figure 9.6: **Gestures set.** Set of 10 gaze gestures employed in the user study to evaluate the performance of different gaze gestures modalities.

Using dwell improved the accuracy, 74%. The saccadic modality using the sequence alignment algorithm had a notably better performance, with an average accuracy of 92% with no dwell, and 95% accuracy with dwell. As can be seen in Figure 9.7, the gliding modality presented a higher variability in the data than the saccadic modality.

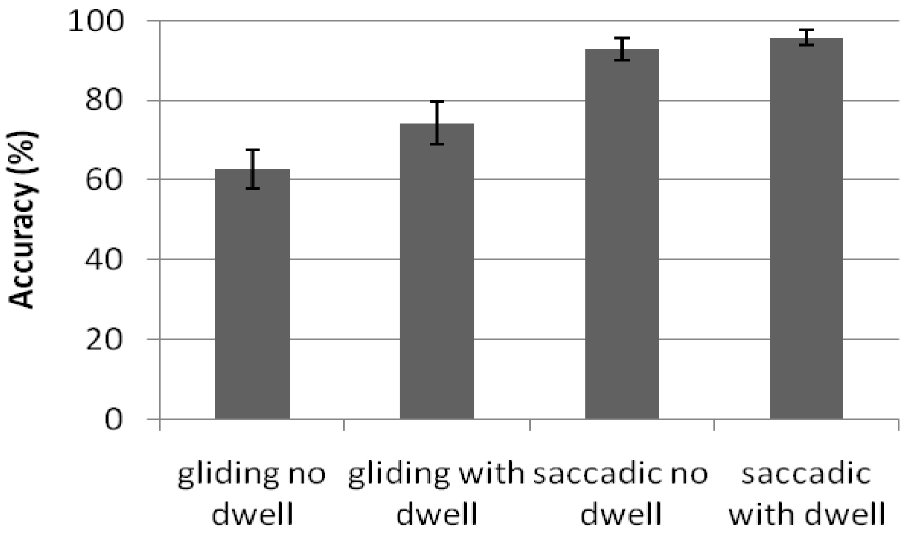


Figure 9.7: **Accuracy test in pilot experiment.** Average accuracy for each of the four conditions in the pilot experiment. Error bars show the standard error of the mean.

Figure 9.8 shows the results for the browsing task. The number of involuntary gaze gestures per minute was 2.5 for gliding without dwell and less than 1.5 for the gliding with dwell condition. The saccadic without dwell had 0.8 involuntary false positives and the saccadic with dwell modality 0.2.

In the velocity task, participants performed gaze gesture c, j and hook and the

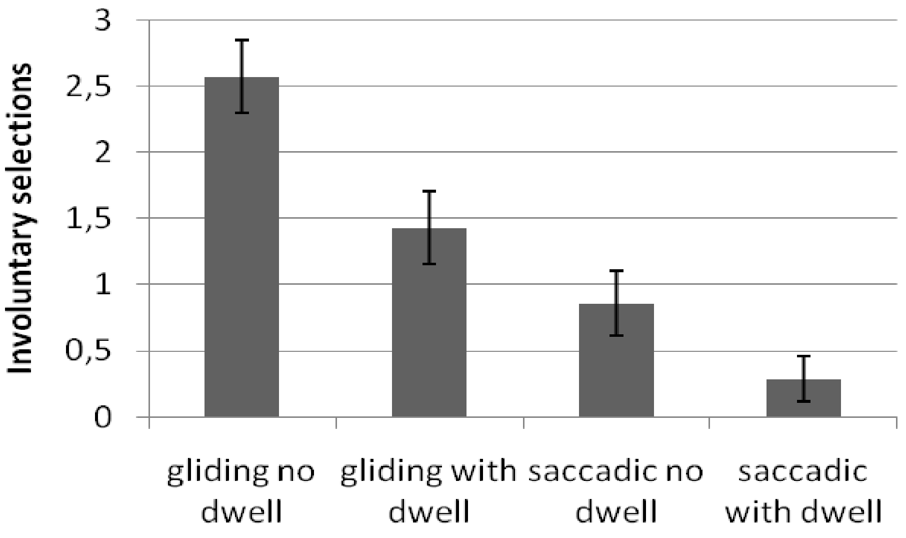


Figure 9.8: **Browsing test in pilot experiment.** Average number of involuntary gaze gestures during 5 minutes of Internet browsing for each of the four conditions in the pilot experiment. Error bars show the standard error of the mean.

time from beginning of the gesture to completion was the result of the subject pressing the Enter key at the beginning and end of the gesture. Figure 9.9 shows the results obtained. The average TPG for the gliding modality without dwell for c, j and hook were 4.5, 5.3 and 5.6 seconds respectively. The average TPGs for the gliding modality with dwell were 5.7, 6.2 and 6.5 seconds. The average TPGs for the saccadic modality without dwell were 1.8, 2.1 and 2.5 seconds. Finally, the average TPGs for the saccadic modality with dwell were 2.6, 3.0 and 3.4 seconds.

Discussion of Pilot Experiment

The gliding modality showed a significantly worse performance compared to the saccadic modality in the pilot experiment both in terms of recognition accuracy and TPG. For that reason, only the modality based on saccadic gaze gestures was considered for further study in a main experiment with a larger set of participants.

9.3.2 Main User Study

Experimental Setup

The main experiment was carried out by 20 participants. All of them were male, with ages ranging from 20 to 59 years old. All of them used computers regularly, but only five of them were familiar with eye tracking. All participants had a European cultural background and all of them but two had an academic education.

The hardware employed was the same as that described in Section 9.3.1. Participants completed the same 3 tasks as in the pilot experiment: an *accuracy task*, a *velocity* task and a *browsing task*. The procedure of both tasks is the same as

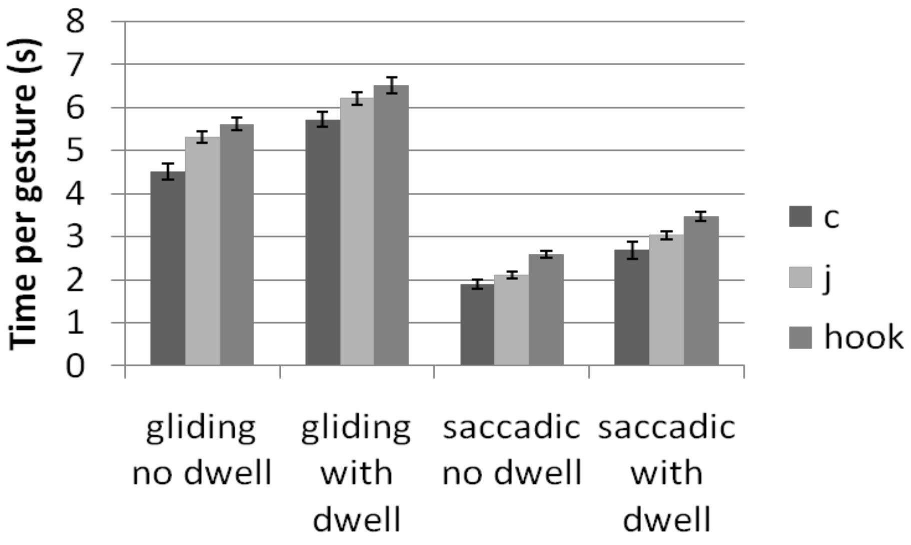


Figure 9.9: **Velocity test in pilot experiment.** Average TPG and accuracy for each of the four conditions in the pilot experiment. Error bars show the standard error of the mean.

described in Section 9.3.1. The experiment was conducted using a 2×5 within subjects design. Factors were gesture technique (*saccadic without dwell* and *saccadic with dwell*) and block. The dependent variables measured were *TPG* and *accuracy* in the velocity and accuracy tasks, and number of *involuntary* gaze gestures in the browsing task. Prior to the experiment, participants were given a brief verbal introduction to the experiment and some time to get acquainted with the system. After completing both tasks, they filled out a questionnaire rating speed, accuracy, fatigue and ease of use for saccadic gaze gestures technique with or without dwell.

Results

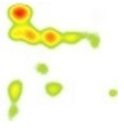
Figure 9.10 shows the accuracy rates of gaze gestures recognition with dynamic programming for the conditions with and without dwell time: 95 and 92%.

Figure 9.11 shows the TPGs for gaze gestures *c*, *j* and *hook*, performed with and without dwell time.

Figure 9.12 shows the number of involuntary gaze gestures per minute detected by the system, i.e. false positives, while the users were browsing the Internet. The condition in which a dwell time was employed had a significantly lower average value.

A group of 10 users repeated the accuracy and velocity tasks over 5 trials to study learning effects, see Figure 9.13 and Figure 9.14.

Upon completing the experiments, participants filled up a questionnaire to give their subjective report on each of the experimental conditions. Figure 9.15 shows the average scores reported by participants for ease of use, speed, tiredness and accuracy of each modality. The gaze gesture technique without dwell obtained a better score in all categories, except in accuracy where both techniques obtained a similar value. Furthermore, when asked which technique would they choose for accessibility



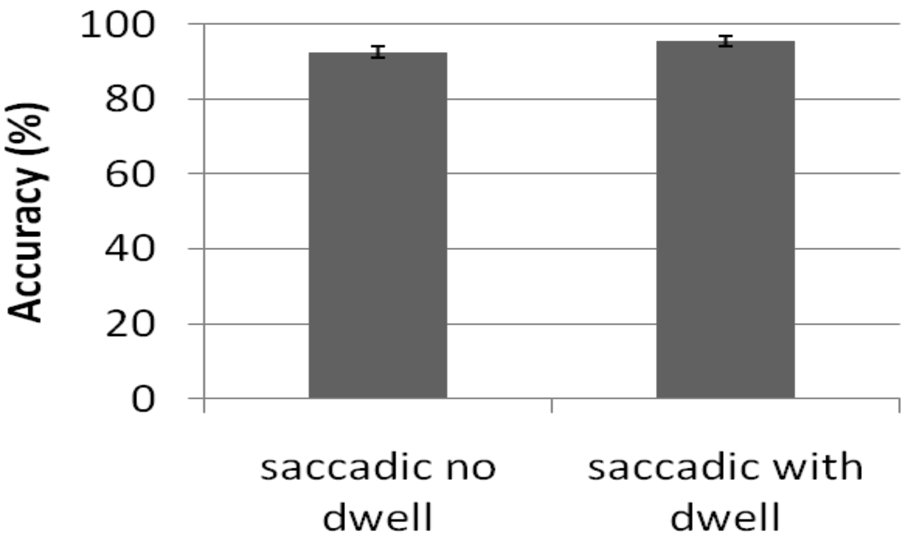


Figure 9.10: **Accuracy test in main experiment.** Average accuracy for each condition in the main experiment. Error bars show the standard error of the mean.

purposes, all 6 participants reported a preference for the saccadic technique without dwell, and claimed that the gaze gesture recognition engine did not interfere with their normal use of the computer.

9.3.3 Recognition Scores Overlap of Intended and Non-Intended Gestures

An additional experiment was carried out to gain more insight into the reasons for the differential performance of each algorithm method and gaze gesture modality. We investigated the recognition scores overlap between intended gaze gestures and normal gaze activity for both recognition algorithms.

During real time recognition, in addition to intended gaze gestures, the recognition engine is exposed to the normal gaze activity performed by the user while using a computer. These patterns are constantly being fed to the recognition algorithms, which, on the search for an intended gaze gesture, perform inference on them. That is, the task of the real-time recognition algorithm is to segment natural eye movements from gaze gestures performed on purpose.

An experiment was designed in which 200 instances from voluntarily performed gaze gestures and 200 instances of normal gaze activity were generated with the experimental conditions of gliding gaze gestures with no dwell and saccadic gaze gestures with no dwell. The recognition scores generated by each recognition algorithm, HTM and Needleman-Wunsch, for each condition were plotted in a histogram.

In the case of the HTM algorithm, the range of scores of normal gaze activity overlaps over a significant subrange with the range of scores obtained by intentional gaze gestures. This is represented in Figure 9.16.

Figure 9.17 shows the range of scores for the sequence alignment algorithm for

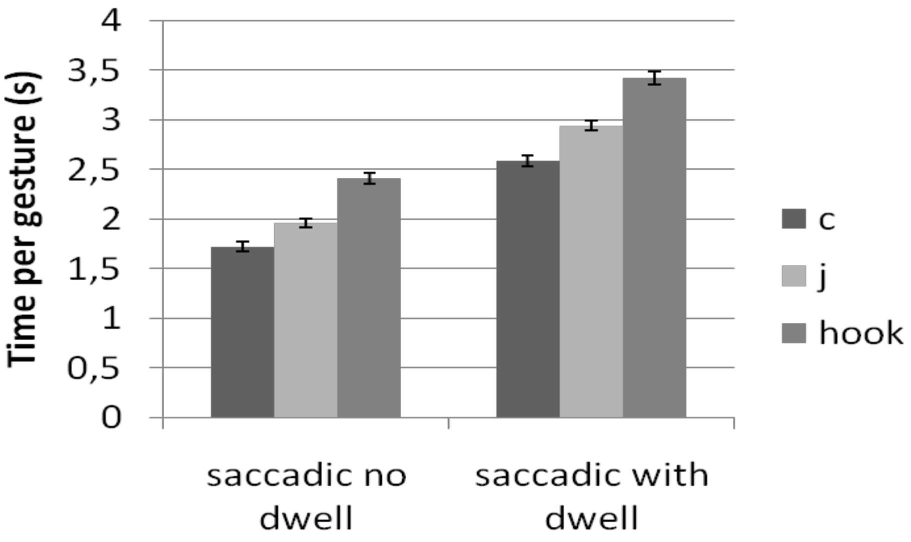


Figure 9.11: **Accuracy test in main experiment.** Average velocities for each condition in the main experiment. Error bars show the standard error of the mean.

saccadic gaze gestures. As can be observed, in this case there is a clear separation between alignment scores obtained for intended gaze gestures and the rest of gaze activity.

9.4 Discussion

The data provided by the gaze tracker and used during the experiments had a relatively high amount of noise and around 1° accuracy. The lack of perfect accuracy is due to the physiological and optical limitations of video-oculography gaze tracking [San Agustin, 2010]. Furthermore, the accuracy deteriorated markedly over time due to the lack of head movement invariance in head-mounted eye tracking setups. The noisy nature of the data constituted a challenge for the recognition engines. In our experiments, the two different modalities of performing a gaze gesture (gliding and saccadic) were recognized with two different algorithmic methods.

We used HTM networks on gliding gaze gestures. HTMs are a conexionist pattern recognition algorithmic optimal for handling data with complex spatio-temporal structures [Hawkins, 2006]. The gliding gaze gesture instances generated noisy and high-dimensional data as can be seen in Figures 9.2 and 9.18, which justified the usage of HTM networks.

The Needleman-Wunsch algorithm was chosen for the recognition of saccadic gaze gestures sequences for its ability to measure sequence similarity. The saccadic gaze gestures generated small sequences of data, 7 elements long. The elements of the sequence represented screen areas over which a subject had scanned his or her gaze over time, see Figure 9.4. High similarity among these gaze sequences indicates high-likelihood of them belonging to the same gesture category.

As shown in Chapter 8, recognition of isolated saccadic or gliding gaze gestures

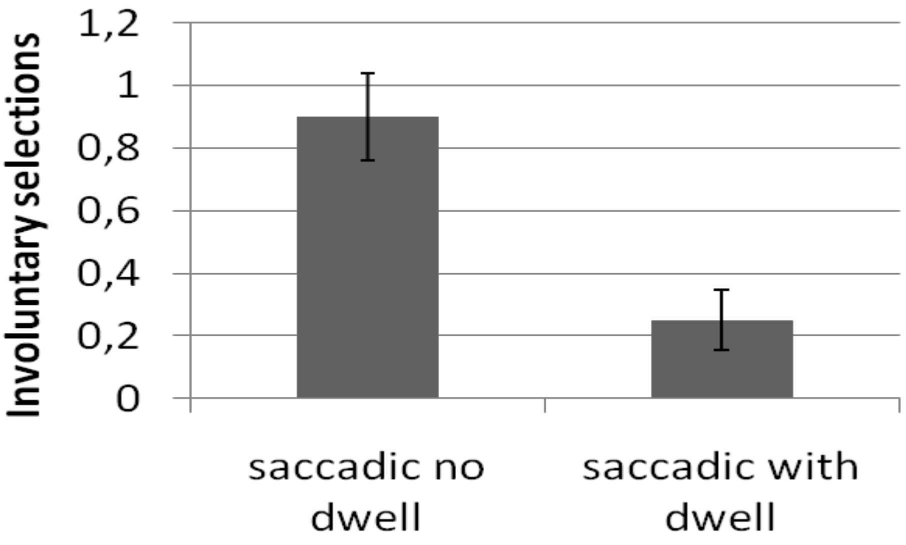


Figure 9.12: **Browsing test in main experiment.** Average number of involuntary gaze gestures per minute detected for each condition in the main experiment. Error bars show the standard error of the mean.

is a relatively trivial task for both our conexionist approach with HTM networks and our sequence alignment approach with dynamic programming. The challenge is gaze gesture recognition in real time. That is, to distinguish intentional gaze gestures intertwined with typical gaze activity during normal operation of a computer through gaze or traditional input methods. This requires finding the right trade-off between sensitivity and specificity of the recognition algorithms. On the one side, we aim to detect the maximum amount of intentionally performed gestures, that is, to maximize sensitivity. On the other hand, it is essential to minimize false positives, that is, to maximize specificity. This challenge was only partially overcome by the HTM recognition engine, but it was consistently overcome by the sequence alignment methodology as shown by the accuracy, velocity and browsing tasks results of both modalities and recognition methods in Figures 9.7 and 9.10.

The suboptimal performance for HTM recognition of gliding gaze gestures was due to several factors. First, the noisy nature of the raw data generated by the eye tracker and the low resolution of the data structure representation matrix, 6×5, created a significant degree of overlap between several instances from different categories of gaze gestures. This could be empirically confirmed as observed in Figure 9.18. An obvious solution that could partially solve the problem would consist on increasing the granularity of the data structure by increasing the dimensions of the matrix structure encoding the sign, see Figure 9.2. However, this would require gathering more training data from users since the degree of overlap among different instances within a category with increased granularity would decrease, and therefore clustering same category instances would be harder for the HTM algorithm. Due to our limited resources to gather additional training data, we constricted ourselves to 30 training instances per category of gliding gaze gesture and the 6×5 matrix data

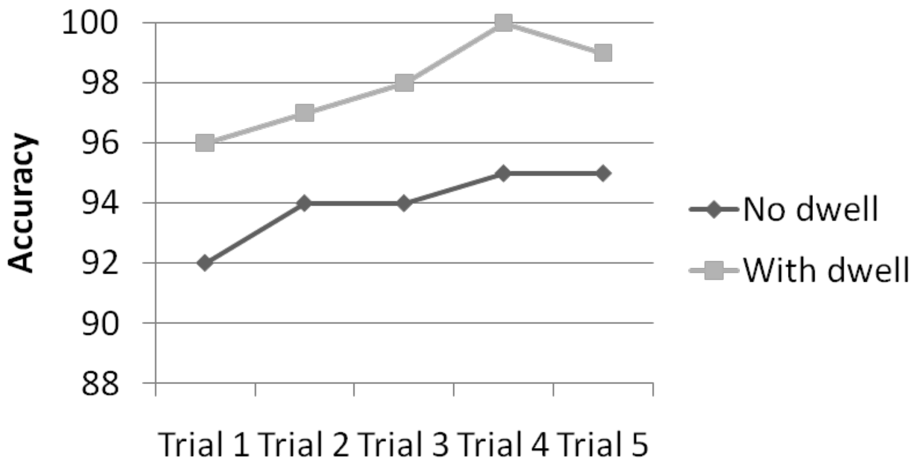


Figure 9.13: **Accuracy learning curve.** Average TPG for each condition over 5 blocks. A learning effect can be noted for both conditions.

structure.

Another reason for the suboptimal performance of HTM recognition of gliding gaze gestures is the pervasive trade-off that emerges between sensitivity and specificity. Decreasing the detection threshold of the similarity score produced by the recognition algorithm increases the chances of detecting a gesture when it is performed but also increases the amount of false positives generated by the algorithm. On the other hand, increasing the detection threshold of this similarity score produced by the recognition algorithm increases the strictness of the similarity, lowering the number of false positives produced, but also missing some true positives that are noisy or not accurate enough. A naive solution to circumvent this problem is to impose on the user the need to indicate through an external switching action, the beginning and end of a gaze gesture. This adds a load on the user and increases the time needed to perform a gesture but also simplifies the task of recognition for the algorithm, which obtains the segmentation values of where a gaze gesture starts and ends. This solution is not always appropriate since the amount of switching channels available to persons with disabilities is markedly limited and in some cases (locked-in patients) non existent. A switch can be simulated by a fixation detection algorithm and dwell time. In this way, the user indicates the beginning and the end of a gaze gesture through dwell activation at the beginning and end of a conscious gaze gesture. As Figures 9.7 and 9.9 show, this strategy improves performance significantly but also increases the amount of time the user needs to perform a gesture.

Finally, the main reason for the suboptimal performance of the gliding gaze gestures recognized with the HTM algorithm was the range of overlap in recognition scores or inference values generated by the algorithm for both the normal gaze activity and the scores obtained by intentional gaze gestures. This is represented in Figure 9.16. For the offline recognition of isolated gaze gestures, the value of the similarity score [George and Jarosy, 2007] produced by the HTM algorithms for the

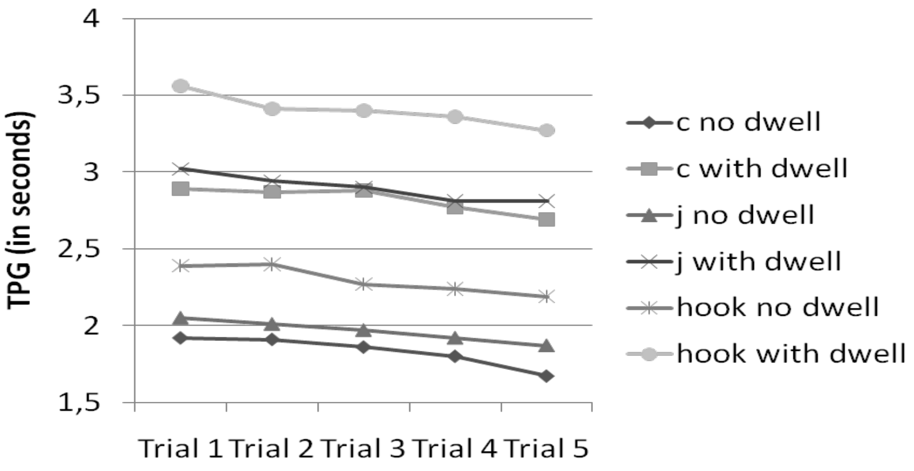


Figure 9.14: **TPG learning curve.** Average TPG for each condition over 5 blocks. A learning effect can be noted for both conditions.

top guess is not critical, as long as it is the largest of all the algorithm guesses for all categories. During inference on different instances of consciously performed gaze gestures, a wide dispersion of similarity scores, several orders of magnitude (Figure 9.16), emerges. That range partially overlaps with inference values from normal gaze activity. Due to this overlap, certain gaze activity unrelated to gestures can be mistaken with an intentional gaze gesture. This region of overlap explains to a great extent the suboptimal performance of HTM algorithms to successfully perform real-time recognition of gaze gestures.

The sequence alignment algorithm used to recognize the saccadic gaze gestures performed very robustly under high levels of noise. The Needleman-Wunsch algorithm is a computationally expensive algorithm, yet we were working with very small sequences, 7 elements long, and small data sets, 10 categories, so this did not constitute a problem. The recognition of saccadic gaze gestures with sequence alignment was very accurate and generated very few false positives during normal browsing of information by a user on the computer, see Figures 9.7 and 9.8. Furthermore, these types of gestures were also much faster to perform by the users than the gliding gestures, as observed in Figures 9.9 and 9.7. This good performance was due in part to the inherent ability of dynamic programming algorithms to warp time and become invariant to sequences that can be performed over different time scales. The reason for this good performance was the clear distinction of alignment scores obtained by conscious gaze gestures and the rest of gaze activity, as shown in Figure 9.17. The clear partition between both distributions results in a significantly lower number of false positives compared to the HTM algorithms employed to recognize gliding gestures. The performance of the sequence alignment algorithm on saccadic gaze gestures improved even more in terms of accuracy and a reduced false positives rate when an artificial beginning and end of a gesture through dwell time was provided by the user, Figures 9.7, 9.8, 9.9, 9.10 and 9.12. Yet, this strategy also increased the time needed to perform a gesture, as shown in Figures 9.9, 9.7 and 9.10.

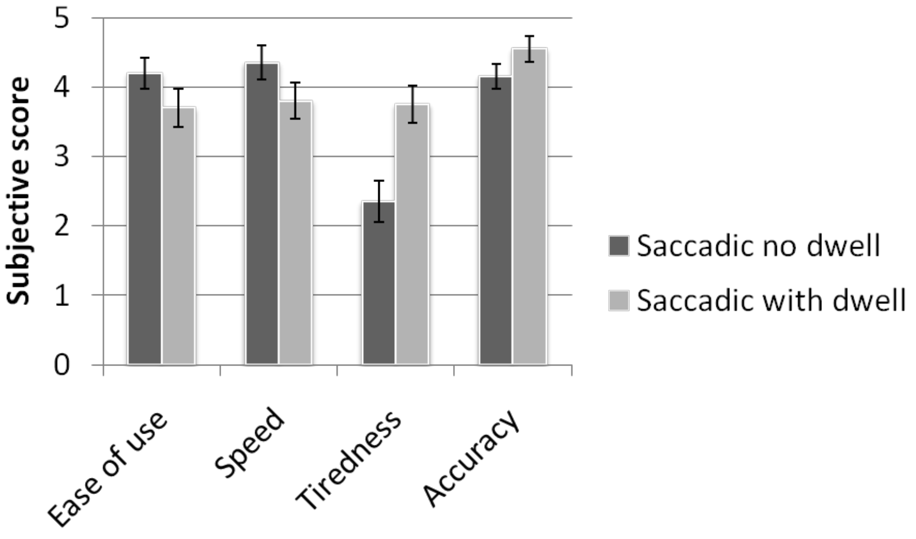


Figure 9.15: **Subjective results in main experiment.** Average score given by the participants rating the ease of use, speed, tiredness and accuracy of each experimental condition. Error bars show the standard error of the mean.

An additional advantage of the saccadic modality has to do with the low variability in terms of accuracy performance and completion velocity that emerged from the pilot experiment results. As can be seen in Figures 9.7 and 9.9, the error bars are markedly larger for the gliding modality in comparison to the saccadic modality. Analysis of the experimental data pointed at a higher intrinsic intra-personal variability of the results obtained, i.e. the experimental results in terms of accuracy and velocity differ significantly within the same user during different trials. The lower dispersion in terms of performance of the saccadic modality highlights again the superiority of this modality over the gliding gaze gestures.

In terms of the actual users of the system, see Figure 9.15, all reported that the saccadic gaze gestures were more intuitive, faster, less straining for eye muscles and less frustrating since, when using the sequence alignment algorithm for detection, it provided a high degree of accuracy. We did encounter some initial reluctance from some users to the idea of using gaze gestures to control a computer since normally the eye is not used to trigger actions. This was due to the counterintuitive nature of employing gaze gestures as control commands. Human beings in general are not used to employ gaze as an output actor; rather, gaze is normally employed for pointing and targeting and then as an input sensor. However, with a bit of training an adaptation, this issue can be easily overcome in a couple of sessions to the extent of becoming a relatively straight forward way of emitting commands. The performance improvement in terms of speed after just 5 short (less than 10 minutes each) experimental sessions is clearly visible in Figure 9.13 and Figure 9.14.

Even though the saccadic gaze gestures with no dwell generated some false positives (Figure 9.12), users reported to prefer not to use dwell because of the faster nature and less stressful condition of not using dwell-activated fixations to signal

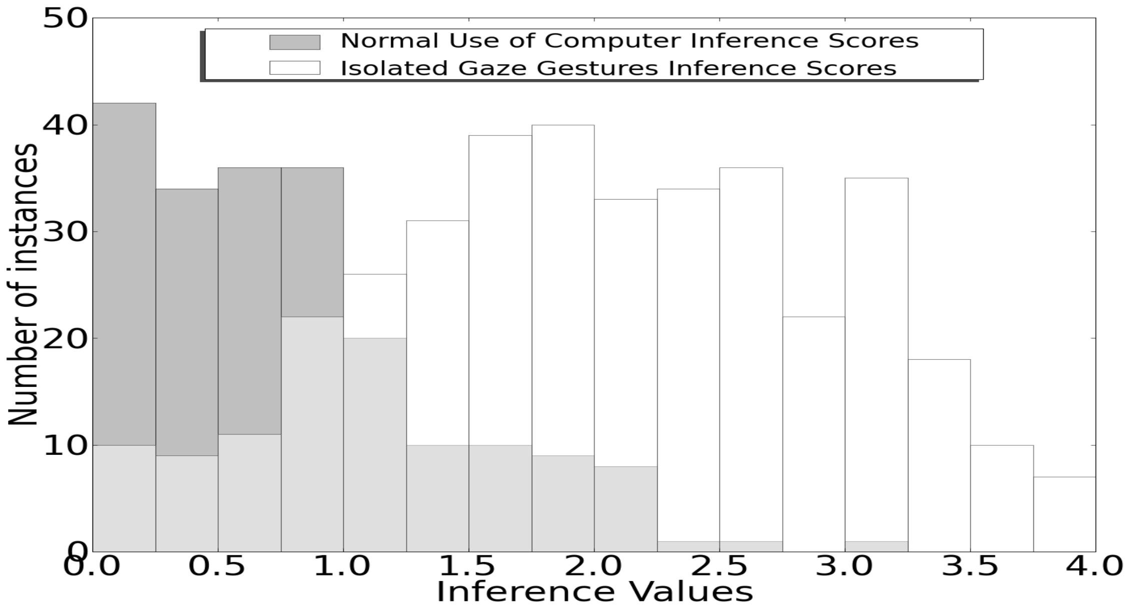


Figure 9.16: **HTM recognition scores distribution for gliding gaze gestures and rest of gaze activity.** The dark grey bars correspond to the histogram distribution of inference values generated by HTM networks when fed with normal gaze activity generated during standard user-computer interaction. The white bars correspond to the distribution of inference values generated by the HTM networks when fed with a wide array of consciously performed gaze gestures. Light grey areas correspond to a significant region of overlap between both distributions.

the start and end of a gesture (Figures 9.10 and 9.15). In a real scenario, users should be able to set their preferred dwell time to signal the beginning and end of a gesture. In this case, users might trade off a decrease in speed for a lower rate of false positives. The proper approach should in any case find a balance between the costs of making an error with the likelihood of making one [Mollenbach, 2010].

At the current state of the technology, the use of gaze gestures is constrained to a limited group of users or environments. For normal users that can employ their hands in standard environments, traditional devices to generate commands such as keyboard and mouse offer superior performance in terms of robustness and intuitiveness. Nonetheless, we believe that as mobile electronic devices become more pervasive and powerful, their inherent reduced display size holds great potential for using gaze gestures to generate control commands. Also in certain environments where use of the hands may be limited, such as a surgery room, eye tracking technology and gaze gesture recognition as a sub-application of the field might be useful. But probably the field where gaze gestures can have its most immediate impact is as an input channel for users with severe disabilities, such as those with high vertebrae spinal cord injuries and particularly locked-in patients who have lost the ability to speak and move and whose gaze constitutes the only output communication channel available. Video-based eye tracking technology has so far allowed these users to po-

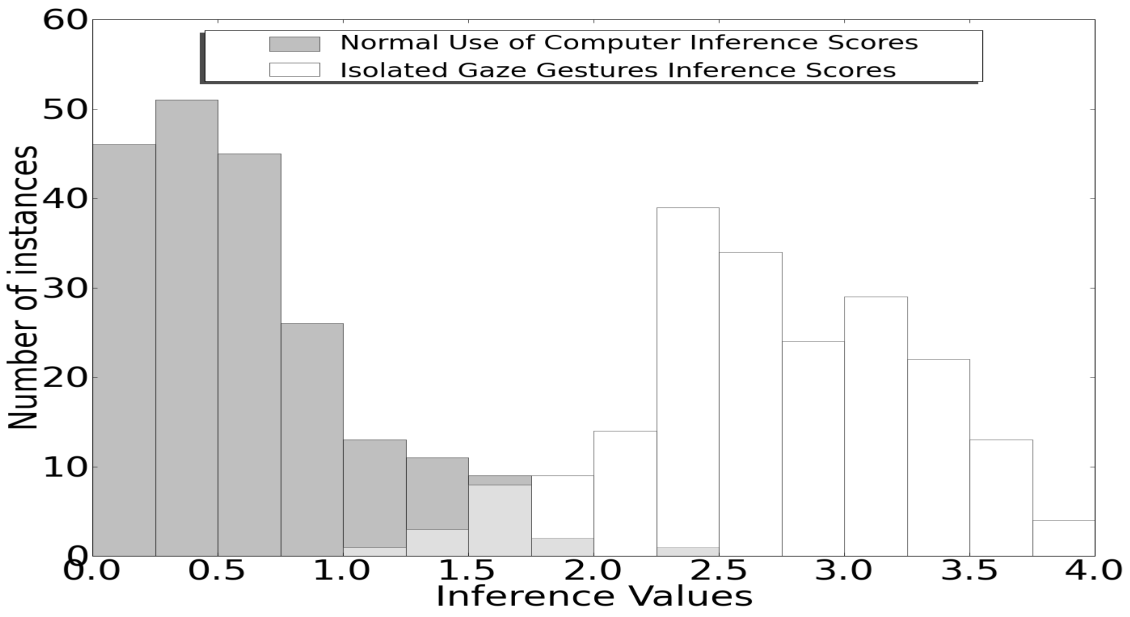
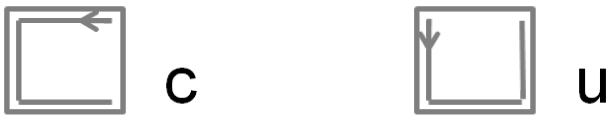


Figure 9.17: **Needleman-Wunsch sequence alignment scores distribution for saccadic gaze gestures and rest of gaze activity.** The dark grey bars correspond to the distribution of inference values generated by the sequence alignment methodology when fed with normal gaze activity generated during standard user-computer interaction. The white bars correspond to the distribution of inference values generated by the sequence alignment methodology when fed with a wide array of consciously performed gaze gestures. Almost imperceptible light grey areas correspond to a minimal region of overlap between both distributions.

sition a cursor on the screen with 0.5° accuracy [Hennessey et al., 2008]. However, the emission of switching actions are challenging for this group of users. Hence, using gaze gestures to generate commands, such as "Enter", "Next Page", "Escape" or even whole macros/scripts, constitutes a valuable information channel asset. Using gaze gestures as a communication channel between users and computers offers several advantages in terms of speeding up command completion, freeing up on-screen real-estate and avoiding the stressful Midas Touch problem inherent to dwell selection in gaze interaction.

9.5 Conclusion

In this work we have proposed and compared two different modalities of performing gaze gestures, gliding gestures and saccadic gestures, in the context of human-machine interaction. Each modality of gaze gesture was recognized by an appropriate machine learning algorithm: HTMs and the Needleman-Wunsch algorithm, respectively. Based on the results of experiments and users reports, we conclude that gaze gestures consisting of sequences of saccades recognized by a sequence alignment algorithm are far superior to performing gaze gestures by gliding the eyes along a



1	0	1	1	0	1
1	0	0	0	0	0
2	0	0	0	0	0
2	2	0	0	0	0
2	2	2	2	3	3

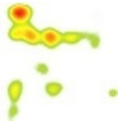
1	1	1	0	0	3
1	0	1	1	0	0
1	0	0	0	0	3
2	2	0	0	0	3
2	2	2	3	3	3

Figure 9.18: **HTM data structure overlap for different categories.** Different categories of gliding gaze gestures generate very similar data structures due in part to their geometric similarity, the noise present in the data, and the physiological limitations on accuracy of eye tracking technology.

predefined path and using an HTM conexionist algorithm for recognition. The four dimensions used to compare both methods, performance of the recognition algorithm, speed of gaze gesture completion, false positives rate and user satisfaction, show that the saccadic sequences recognized by dynamic programming outperform significantly the gliding gestures recognized with HTMs by being faster to perform, less error prone and lighter on the subject while preventing accidental algorithmic recognition of unintentional gaze gestures. Hence, we have shown that the combination of saccadic gaze gestures recognized by sequence alignment holds great potential to reliably convey information from the eyes to a computer. We therefore conclude that saccadic gaze gestures constitute an emerging approach to traditional gaze-computer interaction that is worth of increasing attention.

The motivation for our work has been to build a system able to robustly recognize gaze gestures in real time. The purpose of such system would be to constitute a communication channel between humans and computers, targeting specifically either people with disabilities or environments where traditional input channels are not suitable or could be augmented. For accessibility purposes, our system strives for providing severely motor impaired users with as much autonomy as possible in the areas of communication, environmental control and mobility.

Gaze gestures are not intrinsically intuitive since humans rarely use their eyes for conscious effector purposes beyond gaze pointing. Yet, our experiments prove that it only requires a few minutes of practice to memorize a small set of gestures and perform them repetitively to emit control commands without suffering from eye strain. An advantage of using gaze gestures for human-machine interaction is that gaze gestures are not affected by the low accuracy problems intrinsic to eye tracking, since they do not require a high accuracy resolution and can even be detected in the absence of a calibration procedure. In addition, the 6 muscles involved in eye



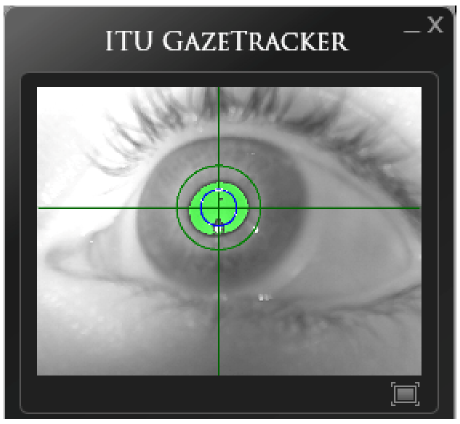
movements are known to be particularly resistant to strain and degenerative conditions [Mollenbach, 2010; Drewes, 2010]. Furthermore, gaze gestures do not occupy real estate on the screen. Hence, freeing up space for other purposes.

For all the above mentioned reasons, we conclude that gaze gestures constitute an innovative way of human-machine interaction. One that can be used by severely paralyzed individuals unable to use a switch but still able to control their gaze as a mono-modal input to control a computer. This group would include people with ALS, brain-stem stroke, or high vertebrae spinal cord injuries. Environments where traditional input devices are out of reach (media centers) or the hands can not be used (surgeons controlling a computer during surgery) could also benefit from gaze gestures for interaction purposes. In addition, situations in which calibration is not possible, such as interactive displays in buildings for random users walking by, represent another potential setup where gaze gestures could be of use. Moreover, gaze gestures could be used in the context of multi-modal input constituting an additional information channel between users and mobile devices with limited screen real estate such as smartphones, projected displays or tablet computers. It is important to notice that many of these devices are already being manufactured with a built-in camera, which paves the way for gaze interaction features to be easily integrated in these apparatus. Summing up, we postulate that gaze gestures as implemented in this work represent a useful addition to the toolbox of utilities being developed for gaze interaction.

In the next chapter, we evaluate the performance of our extended HTM algorithm in comparison to traditional HTMs in the real time recognition of gaze gestures using again a low cost eye tracking but this time on a remote setup.

Chapter 10

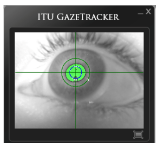
Extended HTMs in the Real Time Recognition of Gaze Gestures



10.1 Introduction

In this chapter we explore the performance of our extended HTM in comparison to traditional HTMs in the real time recognition of gaze gestures [Rozado et al., 2011a]. We use the same extended top node described in Chapter 7. In contrast to Chapters 8 and 9, we employ in this experiment a remote eye tracking setup instead of the previously used head mounted setups. A remote setup increases the noise of the gaze tracking data and decreases the accuracy of the calibration but it also increases the comfort of the user which does not have to wear any head gear and has some room for slight head movements. Hence, in the present Chapter as in Chapter 9, we continue exploring the feasibility of gaze gestures as an input modality to control a computer by carrying out sequences of gaze positions, tracking them using remote video oculography and comparing their real time recognition with traditional and extended HTM.

As discussed in Chapter 9, the potential of gaze gestures as a form of HCI relies



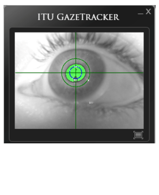
heavily on the ability of machine learning algorithms to properly discriminate intentional gaze gestures from otherwise normal gaze activity during HCI. Gaze gestures can be described in terms of their spatio-temporal structure. The recognition of time series consisting of a spatio-temporal structure unfolding over time is a challenging pattern recognition problem. Spatio-temporal encoding of the features to be learned in a hierarchical system can be a successful approach to solve this type of problems. In this work, we present an extension of an existing pattern recognition algorithm, Hierarchical Temporal Memory, HTM, to improve its performance in real time gaze gestures recognition. Hence, we suggest an extended HTM algorithm to improve traditional HTM performance on this type of problems.

Gaze gestures are composed of a spatial shape that renders itself very suitable to the data structure that HTMs are designed to encapsulate in their hierarchy. However, the temporal evolution of the gaze gesture pattern creates difficulties for traditional HTMs to properly separate the input space into categories as shown in Chapter 9. Our extended Top Node and its inherent ability to warp time, expands the ability of HTMs to handle instances where sequences unfold over time at different speeds and with considerable levels of noise.

Even though remote eye tracking systems are prohibitively expensive for mainstream users (with prices ranging anywhere between 5000$ and 50,000$), in this work we have employed an open source eye tracking algorithm [San Agustin et al., 2010; San Agustin, 2010] and off-the shelf components to build the gaze tracking hardware. Hence, our gaze gesture recognition setup is extremely low cost (less than 50$) but more noisy that commercial systems. Its big advantage is that, due to its low cost, it is widely accessible to most computer users. This shows that gaze gesture recognition does not need expensive high-end hardware and that already built-in cameras on consumer devices such as smartphones or tablets could easily be used for gaze gesture recognition. The ability of our extended HTM system to flexibly learn a wide range of spatio-temporal patterns and its tolerance to noise make it specially appropriate to handle noisy gaze gestures recognition. Our results demonstrate that the extended HTM algorithm can robustly separate natural eye movements during computer usage from intentional gaze gestures. This indicates that using gaze gestures and recognizing them with our extended HTM algorithm constitutes an innovative, robust, easy-to-learn and viable approach to HCI for several environments and device combinations.

10.2 Methods

In our experiments, we compare the performance of two modalities of gaze gestures: those performed using dwell times at the beginning and at the end of the gesture to signal the gaze gesture boundaries and those performed without dwell times. Additionally, we compared the performance of two recognition algorithms: traditional HTM and extended HTM.



10.2.1 Eye tracking

Eye tracking technology is briefly discussed in Chapters 5, 8 and 9. The main difference in terms of hardware setup between the experiments in this Chapter and the experiments in Chapter 9 is that here gaze tracking was carried out in a remote setup as shown in Figure 10.3, while in Chapter 10 a head mounted setup was used.

10.2.2 Gaze Gestures

Gaze gestures were defined an described in length in Chapters 5, 8 and 9.

We carried out preprocessing of gaze gesture data as shown in Figure 8.2 for feeding data to traditional HTMs. The preprocessing consisted on transforming the time series of (x, y) gaze coordinates into a spatio-temporal code in a 6×5 2-Dimensional matrix. This matrix represents the screen over which the gaze gesture was performed and the elements in the matrix indicate both the temporal and spatial structure of the gaze gesture on the screen. Several temporal codifications of the gaze gestures where tried out, as described in Chapter 8, namely: no temporal codification, temporal codification in seconds and three temporal stages codification Due to the noise presented in the gaze tracker data, the mapping from the intended gaze gesture to the streamed gaze data is not perfect as can be observed in Figure 8.2. This constitutes and additional challenge for the recognition algorithm.

For the extended HTM, input data consisted on a matrix representation containing the gaze trajectory over the previous 300 milliseconds.

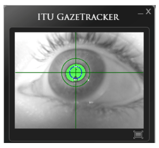
10.2.3 HTM Formalism

HTM theory has been described in Chapters 4, 6 and 7. We used Numenta's Nupic package (v1.7.1) [George and Hawkins, 2009] which is an implementation of a traditional probabilistic HTM to run our experiments.

10.2.4 Extended HTM

The extended HTM model has been described in Chapter 7. Gaze gestures are composed of sequences of spatial arrangements over time that together constitute a sign. At any given time, *t*, the complete representation of the gesture is not available, just a particular spatial PoR of the eyes. It is the temporal sequence of PoRs what constitutes a gesture. The different nature of this kind of problem and the sub-optimal performance of traditional HTM networks to deal with it, justified the usage of the extended HTM to adjust recognition to the temporal requirements of multi-variable time series and in particular gaze gesture recognition.

Figure 10.1 shows an illustration of the extended top node we referred to as *"sequential pooler"*. On it, we can observe how as time passes, our extended top node threads sequences by incorporating in a linear array the spatio-temporal arrangements of the eyes over time. In this fashion, whole sequences capture the entire spatio-temporal structure constituting a gaze gesture from beginning to end.



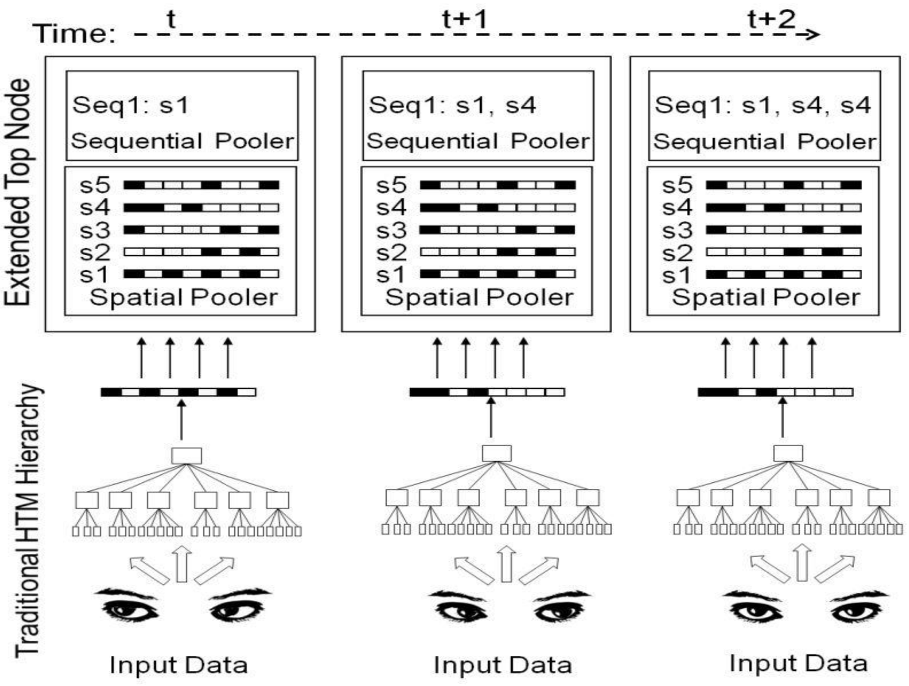


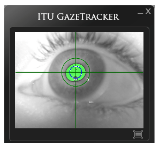
Figure 10.1: **Extended HTM formalism to capture the temporal structure of gaze gestures data.** Traditional top nodes receive binary vectors representing the temporal groups active in the nodes underneath in the hierarchy and map this incoming vectors to categories. Our extended top node instead stores sequences of incoming vectors in an abstraction we referred to as *sequential pooler* and maps whole sequences to single categories. These sequences represent the "utterance" of a sign over time.

Topologically, our extended top node sits at the top of the HTM network, receiving its inputs from a traditional top node underneath serving the purposes of its unique children node, see Figure 10.2. For the real time recognition of gaze gestures, we compared the performance of traditional HTM and the extended HTM topologies shown in Figure 10.2. As can be seen in Figure 10.2, traditional HTM require a data structure containing the complete spatio-temporal characterization of a gaze gesture over time. The extended HTM however, receives the data structure of a gaze gesture as it evolves over time.

10.2.5 Experimental Setup

For the reasons explained in Chapters 8 and 9, neither black or special purpose background, nor markers to attract or better position the gaze in specific coordinates of the screen were used during experimental work.

The remote eye tracking system setup is shown in Figure 10.3. Experiments were carried out using the open source ITU Gaze Tracker [San Agustin et al., 2010] in a remote setup to perform gaze tracking, Figure 10.3. The eye image data was



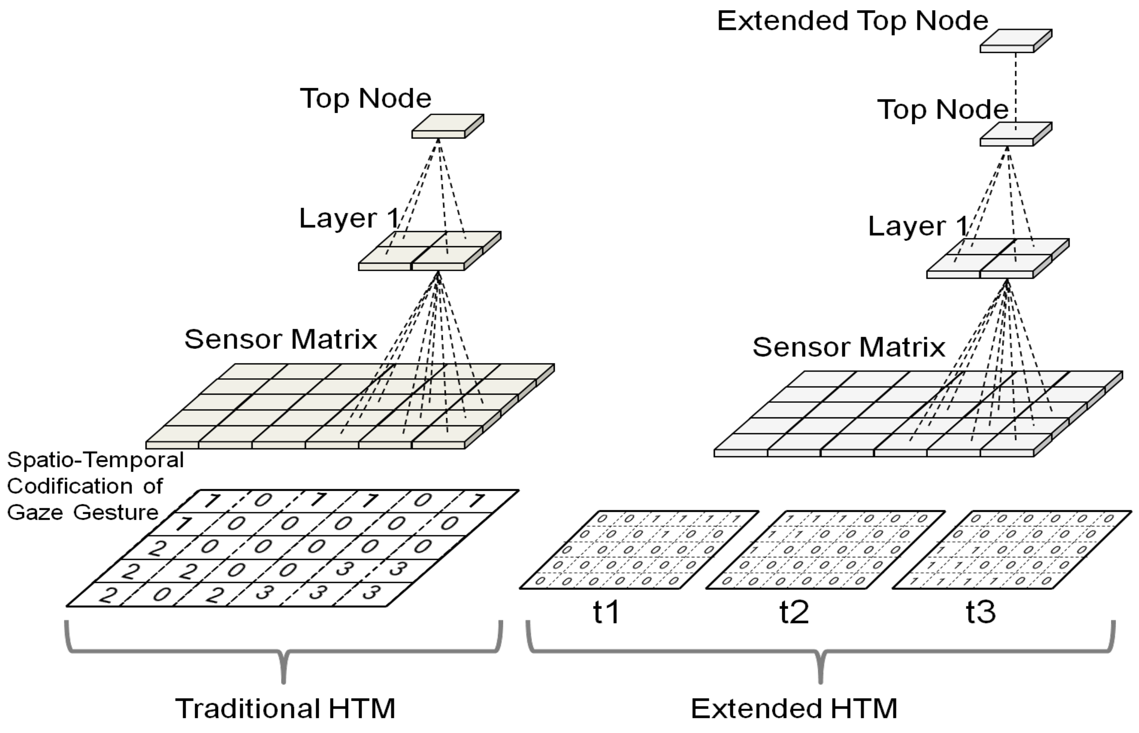
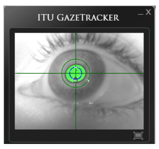
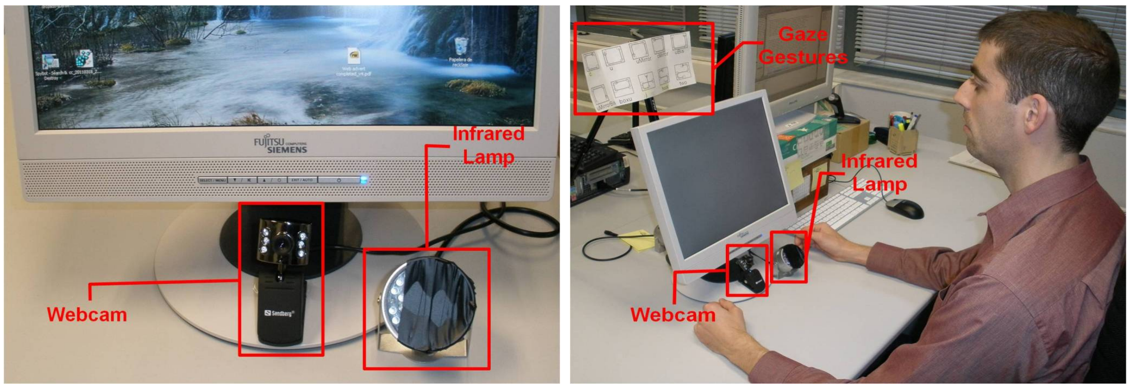


Figure 10.2: **Traditional and extended HTM topologies employed during experiments**. In the traditional HTM, the bottom level nodes are fed with a sensor matrix containing a complete spatio-temporal codification of a gaze gesture. The extended HTM contains, in addition to a traditional HTM network, our customized extended top node. The data structures fed to the extended HTM are the paths scan by gaze over time (t1, t2, t3. . .) during the performance of a gaze gesture.

captured using an off-the-shelf webcam (Sandberg Nightcam 2), mounted under the computer monitor. One infrared lamp was used to improve pupil-to-iris contrast and to create a glint on the cornea that the gaze tracking algorithm uses as reference to measure pupil center displacements during calibration and tracking. Camera resolution was set to 640×480 pixels, and the frame rate oscillated between 15 and 30 frames per second. In this remote setup, the distance from the eye to the camera was approximately 50-60cm. The gaze accuracy of the setup was about 1.5^o, with marked oscillations among different users. Some users achieved up to 0.7^o accuracy while others could not achieve better accuracy than 2^o. These differences are attributable to individual eye shape, degree of concentrations during calibration, tracking elements arrangement, and light conditions.

The user study was carried out by 15 participants. 13 of them were male and 2 female, with ages ranging from 20 to 59 years old. All of them used computers regularly, 5 of them were already familiar with eye tracking and 10 of them had never used a gaze tracking system before. All of them were regular computer users. The subset of 10 participants with no previous gaze tracking experience was involved in an experiment about learning effects. All participants had a European or Latin





(a) (b)

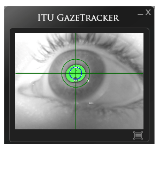
Figure 10.3: **Experimental setup.** Panels *a*, and *b* show the low cost experimental setup and hardware used to carry out remote eye tracking and the subsequent generation of gaze gestures data. The webcam used has a cost of less than 30$ and simply lacks an infrared filter. The infrared lamp costs less than 5$ and part of it has been covered to decrease the intensity of the illumination.

American cultural background and academic education.

In our experiments, we measured the performance of two different modalities of performing the gaze gesture: in one, users had to use dwell time at the beginning and at the end of the gesture to signal the segmentation of the gesture within other types of gaze activity; This requirement obviously makes recognition easier for the algorithm but increases the load on the user and the time required to carry out the gesture. In the alternative modality, users had to perform the gaze gesture without using dwell time to indicate beginning and end of a gesture, making it more comfortable and faster to carry out the gesture by the user, but making the recognition more challenging. The dwell time threshold was set to 400 ms. In the dwell time modality, the user was notified by audio feedback when the dwell time threshold was surpassed. In summary, two different experimental conditions were studied in the experiment: *saccadic gaze gestures without dwell time*, and *saccadic gaze gestures with dwell time.*

We measured the performance of traditional and extended HTMs by measuring after training each network, what percentage of unseen signs' instances from the experimental subjects were assigned to their proper categories using the inference scores provided by the corresponding algorithm during inference.

Participants were asked to complete 3 different tasks, an *accuracy task*, a *velocity task* and a *browsing task*. In the accuracy task, users had to perform the sequence of 10 gaze gestures shown in Figure 8.1 for each experimental condition. Participants were instructed to complete the gestures as fast and as accurately as possible. Users were notified with audio feedback every time the system detected a gesture, regardless of whether the detected gesture was correct or not. When a gesture was detected, the user had to proceed to perform the next gesture. For the accuracy



task, we measured the *accuracy* (percentage of correct gestures detected).

In the velocity task, participants were instructed to perform the same gesture 10 times as fast and as accurately as possible. Three representative gestures were selected for this task: “c”, “j” and “hook” These gestures were chosen for representing gestures with 3, 4 and 5 strokes, see Figure 8.1. In each velocity trial, the approximate *time per gesture*, or *TPG*, measured in seconds was calculated by dividing by 10 the total time taken to perform 10 repetitions of the same gesture.

A subset of 10 participants with no previous eye tracking experience, repeated the accuracy and velocity task over 5 blocks to study learning effects.

The browsing task required participants to browse the Internet during 5 minutes. During this time the number of *involuntary* gaze gestures detected, i.e. false positives, was measured for each of the 2 experimental conditions.

10.3 Results

The data set of gaze gestures to train the HTM networks was generated by 1 user who performed 50 instances of each category. The aggregated sum of all instances for each category reflects that the training data contained a certain amount of noise as can be seen in Figure 10.4.

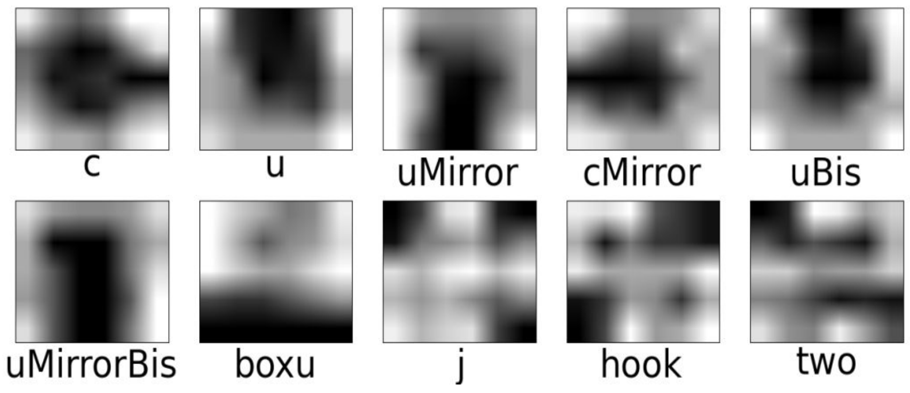


Figure 10.4: **Gaze gestures instances.** Visualization of the degree of overlap of the 50 instances per category of gaze gestures data used for training the HTM networks. A certain degree of noise is clearly visible in the data.

A user study with 15 participants was carried out to determine the performance during inference of our extended HTM system in comparison to traditional HTMs for detection of gaze gestures in terms of accuracy, time per gesture and amount of false positives. The motivation for the modification of traditional HTMs lie on the limited accuracy of traditional HTM to solve the problem of gaze gesture recognition in real time.

HTM theory predicts the ability of HTM networks to warp time by increasing the number of layers in the topology [George and Jarosy, 2007]. That is, a higher number of layers in the topology should extend the temporal invariance of the algorithm at the higher nodes of the network and hence, the HTM algorithm should perform learning and inference on problems where data structures slowly unfold over time. To test this theoretical prediction, we carried out a number of simulations testing the recognition accuracy of traditional HTM networks with increasing number of layers in their topology to determine how increasing the number of layers affects the performance of traditional HTM networks on a problem where instances unfold over time. As can be seen in Figure 10.5, increasing the number of layers improves performance on the problem at hand for traditional HTMs but only up to the 2 Layers level. Increasing the number of layers further does not improve recognition performance. This optimal low number of layers is good in terms of maintaining to a minimum the computational complexity of the algorithm. Still, the recognition accuracy of traditional HTM even at the optimal 2 Layers level is not satisfactory, see Figure 10.5. This first experiment highlighted the necessity for HTM algorithms to be modified in order for them to better accommodate recognition problems where instances unfold over time. Further experiments were all carried out with 2 Layer network topologies as the ones shown in Figure 10.2. Recognition was carried out with traditional HTM and with the extended HTM.

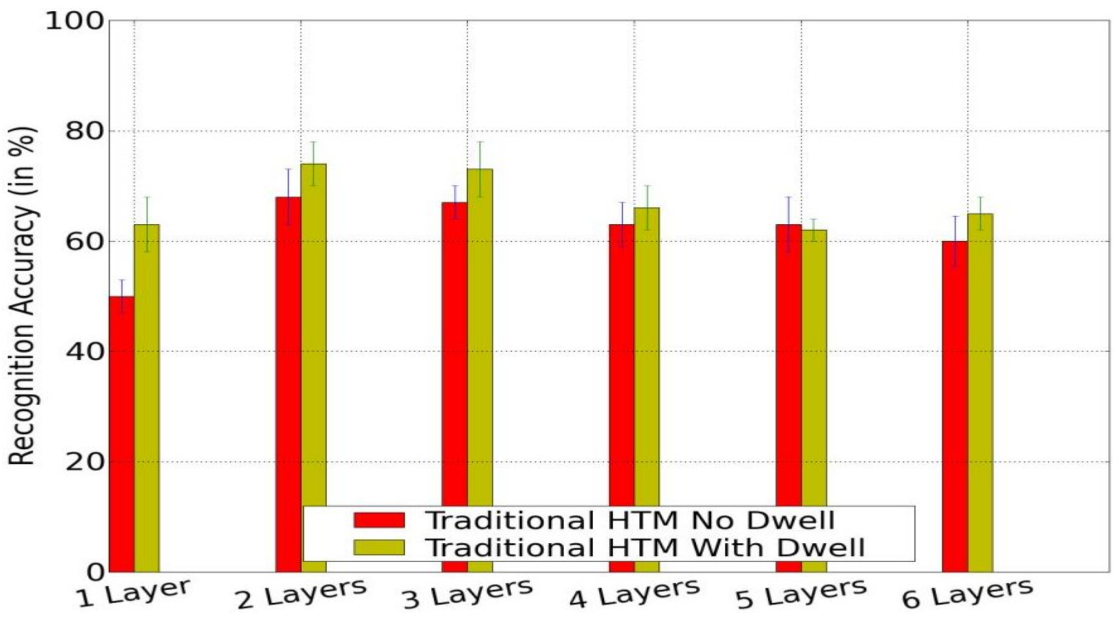
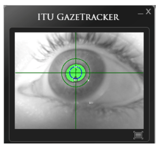


Figure 10.5: **Optimal topology search.** Gaze gestures recognition accuracy for traditional HTM networks with different number of layers for the experimental conditions: with and without dwell. The bars show the percentage of correct classifications achieved by 1, 2, 3, 4, 5 and 6 Layer networks. As can be seen in the graph, best performance for traditional HTMs is achieved by topologies with 2 Layers.

Our main experiment consisted on 3 different tasks: accuracy, browsing, and velocity. Figure 10.6 shows the results obtained for the accuracy task. The extended HTMs with no fixations had an average accuracy of 94% and the extended HTMs with fixations had an average accuracy of 98%. As can be seen in Figure 10.6, the extended HTM algorithm clearly outperforms the traditional HTM in terms of



accuracy.

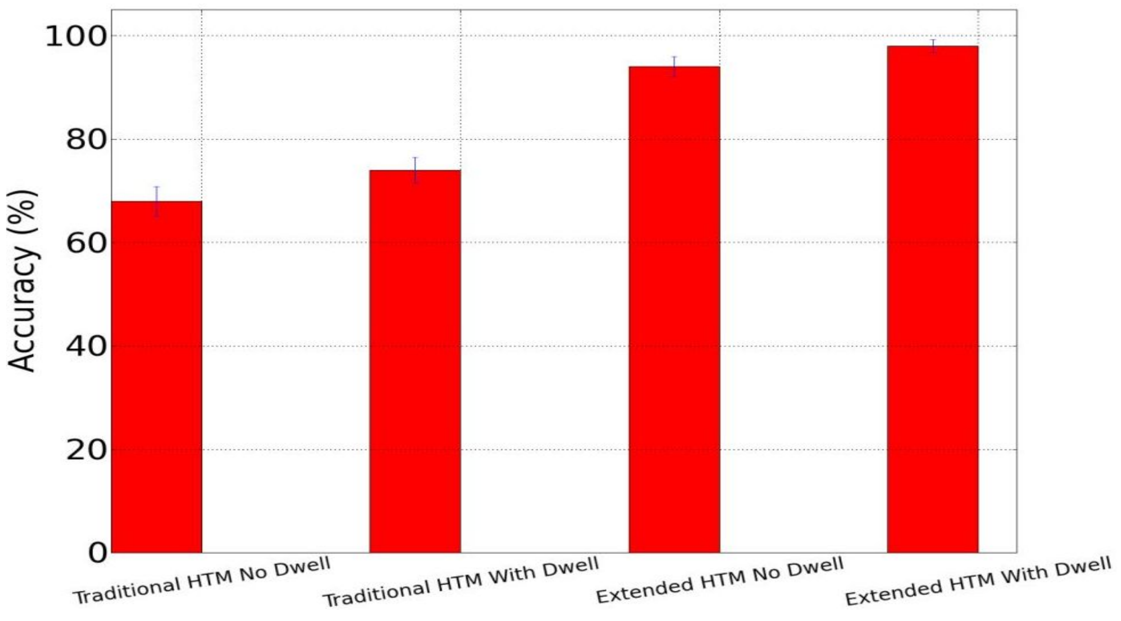


Figure 10.6: **Accuracy experiment.** Average recognition accuracy for each of the four conditions in the experiment. Error bars show the standard error of the mean.

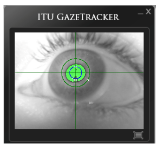
To obtain measurement of TPG, 3 representative gestures with 3, 4, and 5 strokes were selected from the data set from Figure 8.1, namely “c”, “j” and “hook”. The average TPG for each was inferred by measuring the time required to complete 10 repetitions of each and deriving the average TPG. The results are shown in Figure 10.7. Average TPG for gestures “c”, “j” and “hook” with no dwell time were 0.8, 1.3 and 1.9 seconds respectively. Using dwell times to signal beginning and end of gestures generated TPGs of 2, 2.4 and 3.1 seconds respectively.

Figure 10.8 shows the results for the Browsing Experiment, plotting the number of involuntary gaze gestures per minute for the extended and traditional HTMs with and without dwell during 5 minutes of Internet browsing. The best performance (less false positives detected) was obtained by the gesture modality with fixations and recognized with the extended HTM.

A sub-study with 10 participants with no previous eye tracking experience was carried out to study learning effects. Learning effects were studied for both accuracy and TPG. There was no clear learning effect in terms of accuracy over the 5 experimental blocks (Figure 10.9), but there is a noticeable, albeit slight, learning effect in terms of TPG over the 5 experimental blocks (Figure 10.10).

We also studied the impact of vocabulary size on recognition accuracy, i.e. how increasing the number of categories to be recognized negatively affects performance for the different algorithms being compared. As can be seen in Figure 10.11 the effect is more noticeable for traditional HTMs.

HTM requires training sets where each category to be classified is represented by several instances. To determine the effect on recognition accuracy of increasing



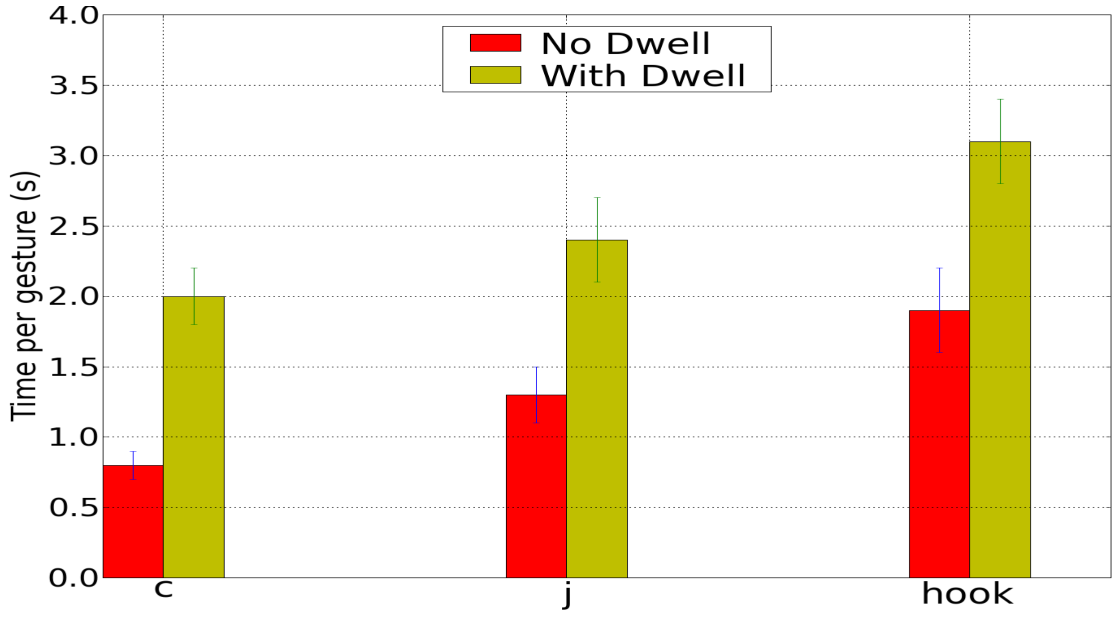


Figure 10.7: **Velocity experiment.** Average TPG for 3, 4 and 5 stroke gestures recognized with the Extended HTM algorithm. Error bars show the standard error of the mean. Note that the modality with dwell includes the time needed for the two long fixations to signal beginning and end of a gesture (400ms each).

the number of training instances for each method being compared, we plotted the results of simulations in which we increased the number of training instances in discrete steps for the methods being compared. As can be observed in Figure 10.12, increasing the number of training instances improves performance in general but the extended HTM always outperforms the traditional HTM methodology.

Even though our extended HTMs achieved better recognition accuracy for gaze gestures than traditional HTMs, the inference time of our extended HTM algorithm was worse than traditional HTMs. This is due to the heavy demands in terms of processing power of the Needleman-Wunsch algorithm [Needleman and Wunsch, 1970] for sequence alignment employed by our extended top node. Still, given the short character of the sequences generated during gaze gesture performance, the recognition time during inference was still in the order of milliseconds. This fast inference capability of our extended HTM algorithm is important for real time applications.

An experiment was designed in which 150 instances from voluntarily performed gaze gestures and 150 instances of normal gaze activity were generated. The recognition scores generated by the traditional and extended HTM algorithm during inference under the experimental condition of saccadic gaze gestures with no dwell time were plotted in a histogram. Figure 10.13 shows how the range of scores obtained by normal gaze activity overlaps over a significant sub-range with the range of scores obtained by intentional gaze gestures when using the traditional HTM as recognition algorithm.

Traditional HTMs are very good at detecting gaze gestures offline [Rozado et al.,

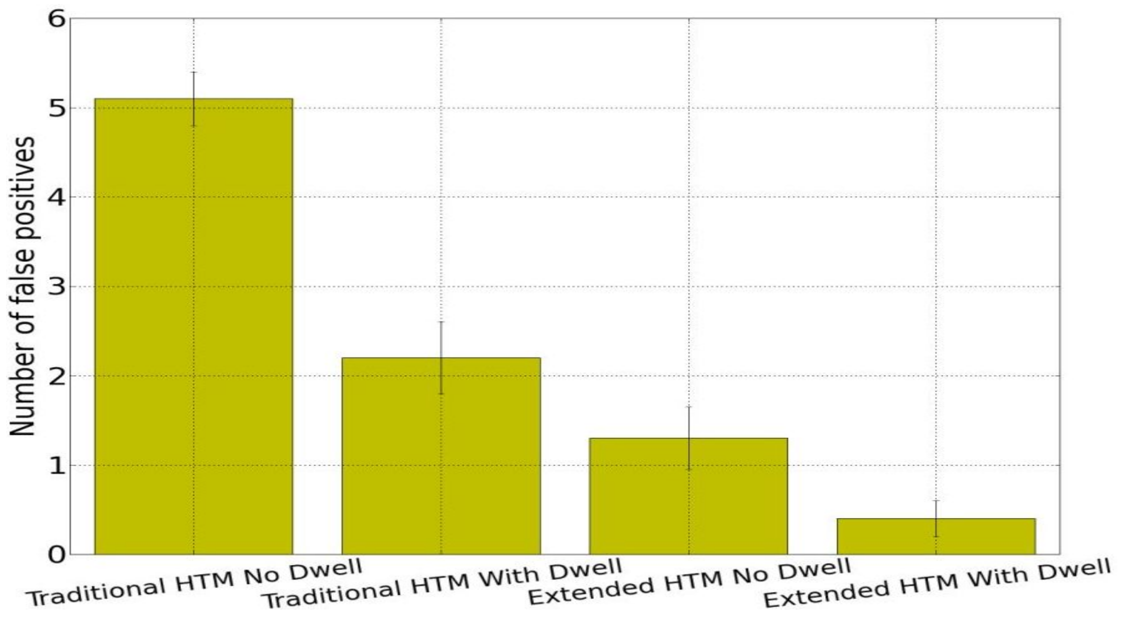
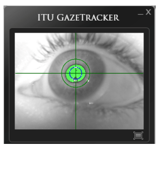


Figure 10.8: **Browsing experiment.** Average number of involuntary gaze gestures detected during 5 minutes of Internet browsing for each of the experimental conditions. Error bars show the standard error of the mean.

2011b]. For different gestures and degrees of noise, traditional HTM almost always gets the proper category of the instance being shown correct. The problem is that the range of scores for different instances varies considerably (from 0.1×10^{-12} to 3) and during normal gaze interaction with a computer, standard gaze activity is wrongly identified as a gaze gesture since the algorithm often assigns to non-gaze gestures activity patterns an inference score also within the (0.1×10^{-12} to 3) range obtained by consciously performed gaze gestures. Therefore, it is not possible to find a proper threshold for the inference scores in terms of what it is a gaze gesture and what it is not. This inability of traditional HTMs to properly discriminate conscious gaze gestures from standard gaze activity can be visualized in Figure 10.13. There is not a proper threshold on the inference score axis to determine what is a gaze gesture and what is standard gaze activity since the scores for both types of gaze movements clearly overlaps. A low threshold would be very sensitive but would generate many false positives. A higher threshold would not generate as many false positives but it would miss a lot of true gaze gestures.

Figure 10.14 shows the range of scores obtained for saccadic gaze gestures and standard gaze activity using the extended HTM algorithm. As can be observed, there is a clear separation between scores obtained for intended gaze gestures and the rest of gaze activity. This means, that with the extended HTMs a clear cutoff threshold for inference values exists that is able to discriminate most intentional gaze gestures from non-gaze gestures activity. Hence, the extended HTM and its ability to warp the temporal unfolding structure of a gesture is able to robustly discriminate conscious gaze gestures from standard gaze activity during gaze computer



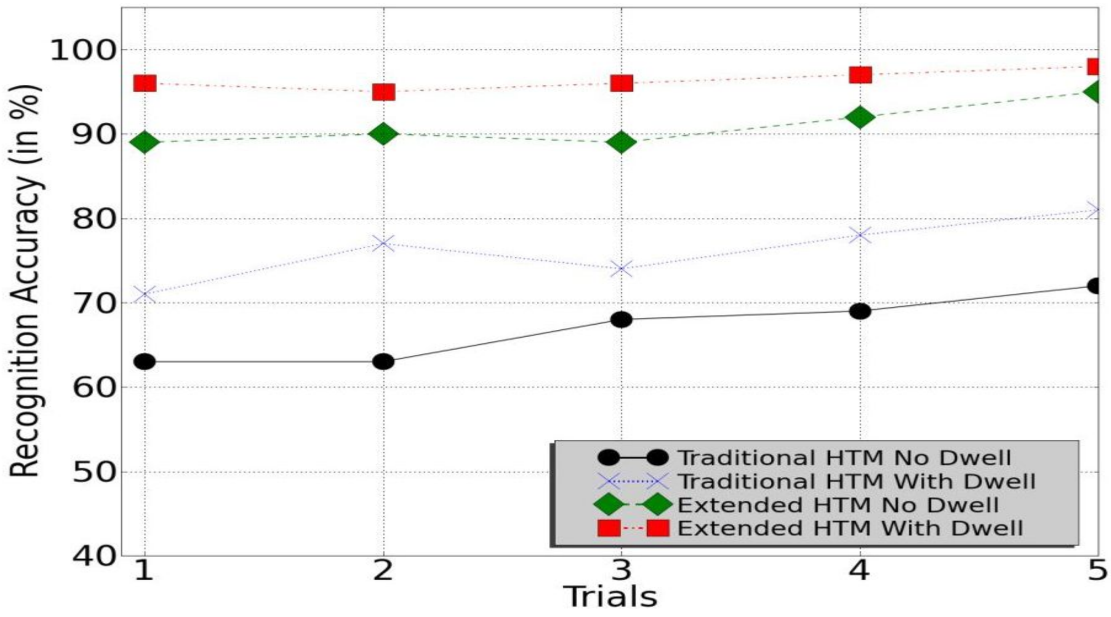


Figure 10.9: **Learning curve on accuracy.** Average recognition accuracy for each experimental condition over 5 blocks for 10 users with no previous gaze tracking experience.

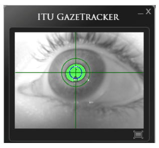
interaction.

10.4 Discussion

The spatio-temporal characteristics of gaze gestures require specific features of the recognition engine. The low cost hardware employed in the experiments generated noisy data that demanded a robust and noise-tolerant recognition engine. In this paper we have presented an alternative approach to how HTMs handle data at the top node in order to improve their performance on machine learning tasks in which instances are composed of a noisy sequence of patterns that unfold slowly over time. As a proof of principle, we have used a data set of gaze gestures, an innovative form of HCI.

Our extension of a traditional HTM system consists of a reformulation of the network's top node that allows it to store sequences of input patterns constituting whole instances of the data set and to map these sequences to categories from the category space. Doing this, the top node is able to warp time and create temporal invariance.

In gaze gesture recognition, it is the orderly sequence of gaze vectors arriving over time what constitutes a gesture, as oppose for instance to image recognition, where at any time point, the input vector represents a complete characterization of an image category. As can be seen in Figure 10.6, the extended HTM algorithm clearly improves the performance of traditional HTMs for gaze gesture recognition. Although in this work we have focused on input data coming from an eye tracker,



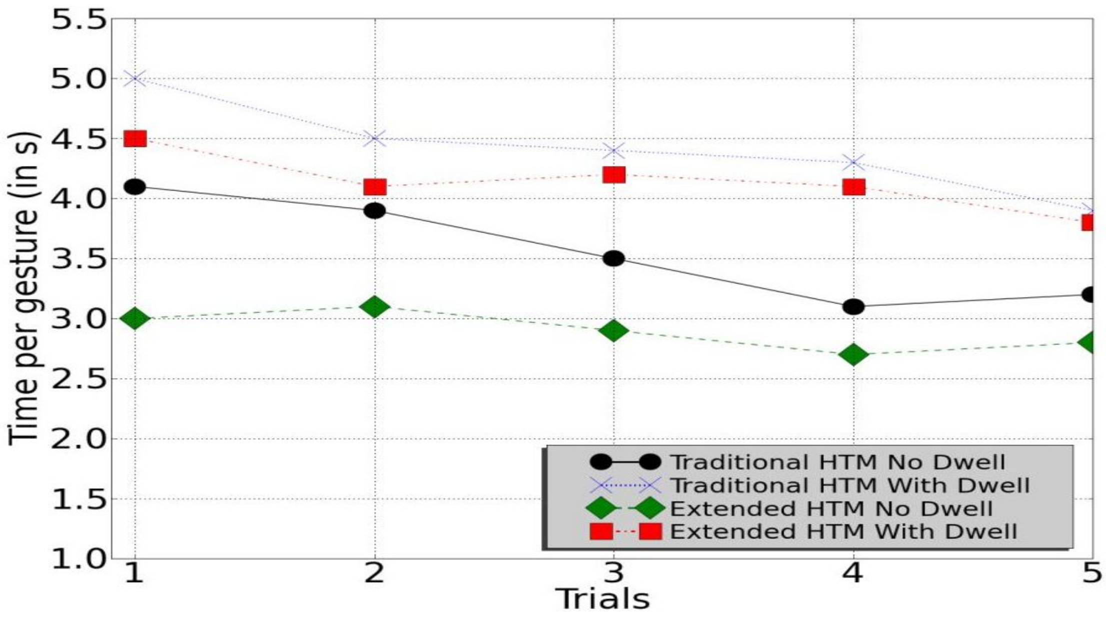


Figure 10.10: **Learning curve on TPG.** Average TPG for each experimental condition over 5 blocks for 10 users with no previous gaze tracking experience.

HTM theory in general and the extended HTM approach in particular are very flexible in terms of accepting input data streams from a variety of sensors. As long as the data representations in the input contains a spatio-temporal structure, our extended HTM approach, after training with a sufficient number of training instances, can achieve good recognition accuracy (e.g up to 98% in our experiments). The method is also light on computing resources during inference, requiring only milliseconds, which makes it easily applicable to real-time requirements contexts.

Recognition of isolated gaze gestures is a relatively trivial task for both traditional HTMs and our extended HTM system [Rozado et al., 2011b]. The challenging problem is gaze gesture recognition in real time. That is, to distinguish intentional gaze gestures intertwined with typical gaze activity during normal gaze interaction with a computer. This requires finding the right trade-off between sensitivity and specificity of the recognition algorithms. On the one side, the aim is to detect the maximum amount of intentionally performed gestures, that is, to maximize sensitivity. On the other hand, it is essential to minimize false positives, that is, to maximize specificity. This challenge was only partially overcome by the traditional HTM in our experiments, but it was robustly overcome by our extended HTM as shown by the accuracy, velocity and browsing tasks results in Figures 10.6, 10.7 and 10.8.

The suboptimal performance of traditional HTM recognition of gaze gestures was due to several factors. First, the noisy nature of the raw data generated by the eye tracker and the low resolution of the data structure representation matrix, 6×5, created a significant degree of overlap between several instances from different categories of gaze gestures, see Figure 10.4. This constitutes a challenge for any

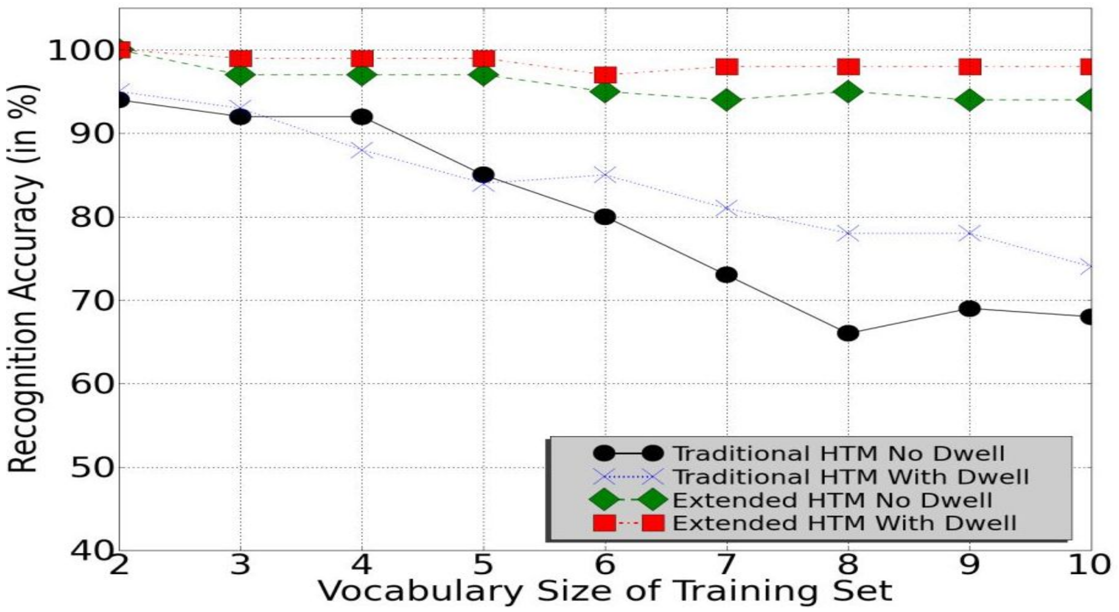
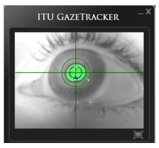


Figure 10.11: **Decreasing performance with increasing vocabulary size.** The recognition accuracy for each method being compared decreases as the vocabulary size of the sign gestures data set to be learnt increases. Yet, this effect is more noticeable for Traditional HTMs.

type of recognition algorithm. A naive solution would be to increase the granularity of the data structure. However, this would require gathering more training data from users since the degree of overlap among different instances within a category with increased granularity would decrease, and therefore, clustering same category instances would be harder for the HTM algorithm. Due to our limited resources to gather additional training data, we constricted ourselves to 50 training instances per category of gaze gestures and the 6×5 matrix data structure.

Gaze gesture recognition is no stranger to the pervasive trade-off between sensitivity and specificity. Decreasing the detection threshold of the similarity score produced by the recognition algorithm increases the chances of detecting a gesture when it is performed but also increases the amount of false positives generated by the algorithm. On the other hand, increasing the detection threshold of this similarity score produced by the recognition algorithm increases the strictness of the similarity, lowering the number of false positives produced, but also missing some true positives that are noisy or not accurate enough and hence only obtain a low similarity score. An obvious solution to circumvent this problem is to impose on the user the need to indicate through an external switching action, the beginning and end of a gaze gesture. This adds a load on the user and increases the time needed to perform a gesture but also simplifies the task of recognition for the algorithm, which obtains the segmentation values of where a gaze gesture starts and ends. This solution is not always appropriate since the amount of switching channels available to persons with disabilities is markedly limited and in some cases, such as locked-in patients, non existent. A switch can be simulated by a fixation detection algorithm



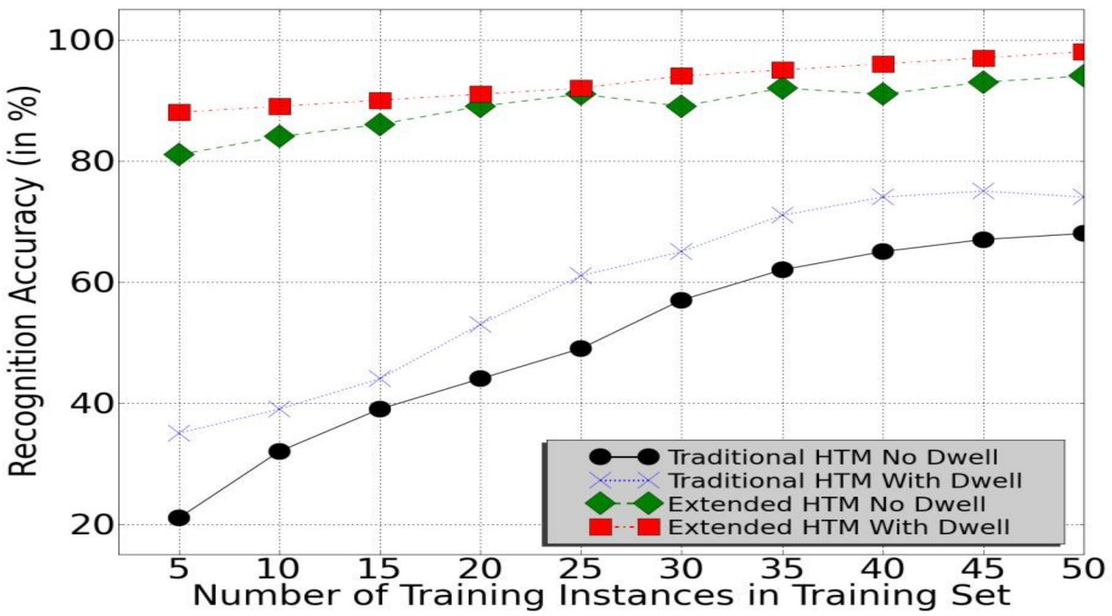
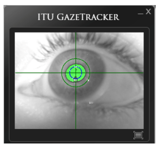


Figure 10.12: **Improving performance with increasing number of training instances.** The recognition accuracy for each method being compared increases as the number of training instances gets larger.

and dwell time. In this way, the user indicates the beginning and the end of a gaze gesture through dwell activation at the beginning and end of a conscious gaze gesture. As Figures 10.6 and 10.7 show, this strategy improves accuracy performance significantly but also increases Time per Gesture, TPG.

The suboptimal performance of traditional HTM algorithms on real time gaze gesture recognition is mainly due to the degree of overlap between the range of inference scores generated by the algorithm for the non-gesture gaze activity and the range of inference scores generated by the algorithm for intentional gaze gestures, as illustrated by Figure 10.13. For the offline recognition of gaze gestures, the value of the inference score [George and Jarosy, 2007] produced by the HTM algorithms for the top guess is not critical, as long as it is the largest of the list of category guesses that the algorithm generates. During inference on different instances of consciously performed gaze gestures, a wide dispersion of similarity scores emerges, see Figure 10.13. That range partially overlaps with inference values from normal gaze activity. Hence, it becomes impossible to determine a threshold able to discriminate gaze activity unrelated to gestures with an intentional gaze gesture for the range of scores generated by the traditional HTM.

The recognition of saccadic gaze gestures with our extended HTM algorithm was very accurate and generated very few false positives during normal browsing of information by a user, see Figures 10.6 and 10.8, clearly outperforming the traditional HTM approach. The Needleman-Wunsch algorithm [Needleman and Wunsch, 1970] employed by the top node for sequence alignment, is a computationally expensive algorithm, yet we were working with very small sequences, 5-7 elements long, and small data sets, 10 categories, so computing power was not a limiting factor. This



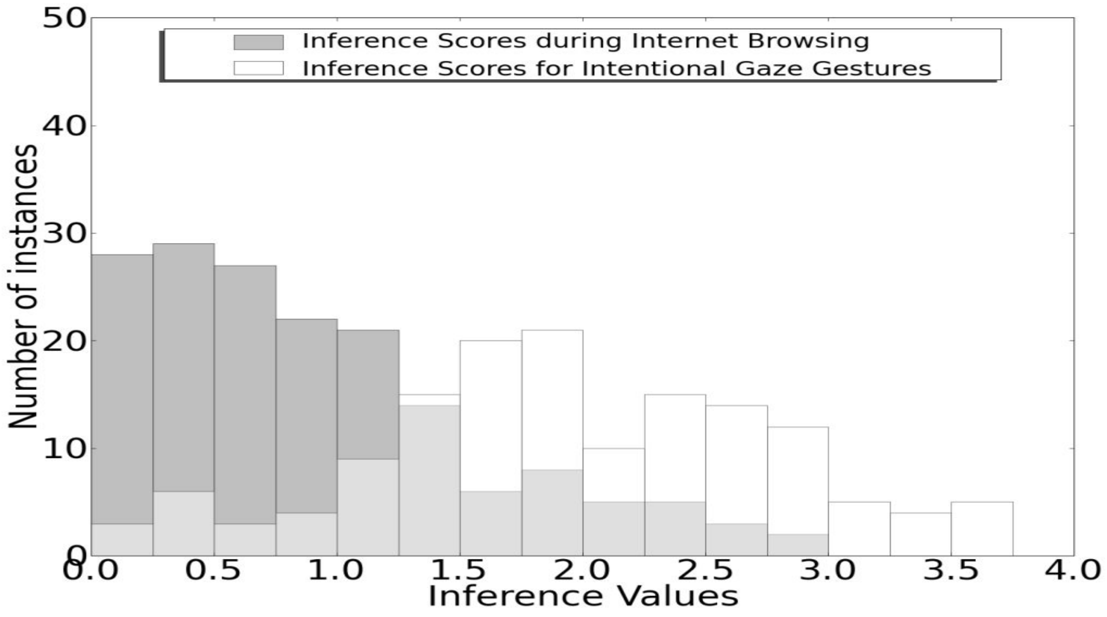


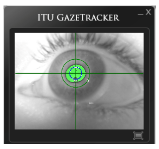
Figure 10.13: **Traditional HTM recognition scores distribution for gaze gestures and standard gaze activity while browsing the Internet.** The dark grey bars correspond to the histogram distribution of inference values generated by a traditional HTM network when fed with normal gaze activity generated during standard user-computer interaction. The white bars correspond to the distribution of inference values generated by the HTM networks when fed with a wide array of consciously performed gaze gestures. Light grey areas correspond to a region of overlap between both distributions.

good performance was due to the inherent ability of dynamic programming algorithms in the top node to warp time and become invariant to sequences that can be performed over different time scales. This translated on a clear distinction of alignment scores obtained by conscious gaze gestures and the rest of gaze activity, as shown in Figure 10.14. The clear partition between both distributions results in a straight forward way to discriminate normal gaze activity from consciously performed gaze gestures by choosing a threshold in the middle of the overlapping range of scores as a way to minimize false positives and maximize the sensitivity of gaze gesture detection.

Additional advantages of the extended HTM over traditional HTM is the recognition accuracy robustness of the former to increasing data set vocabularies, i.e. number of recognition categories, see Figure 10.11, and lesser requirements in terms of number of training instances in the training set, see Figure 10.12.

The performance improvement in terms of TPG after just 5 short (less than 10 minutes each) experimental sessions is clearly visible in Figure 10.10 for gaze gestures with and without dwell time and illustrates the fast assimilation of gaze gestures as a form of output communication.

Even though the saccadic gaze gestures with no dwell time generated some false



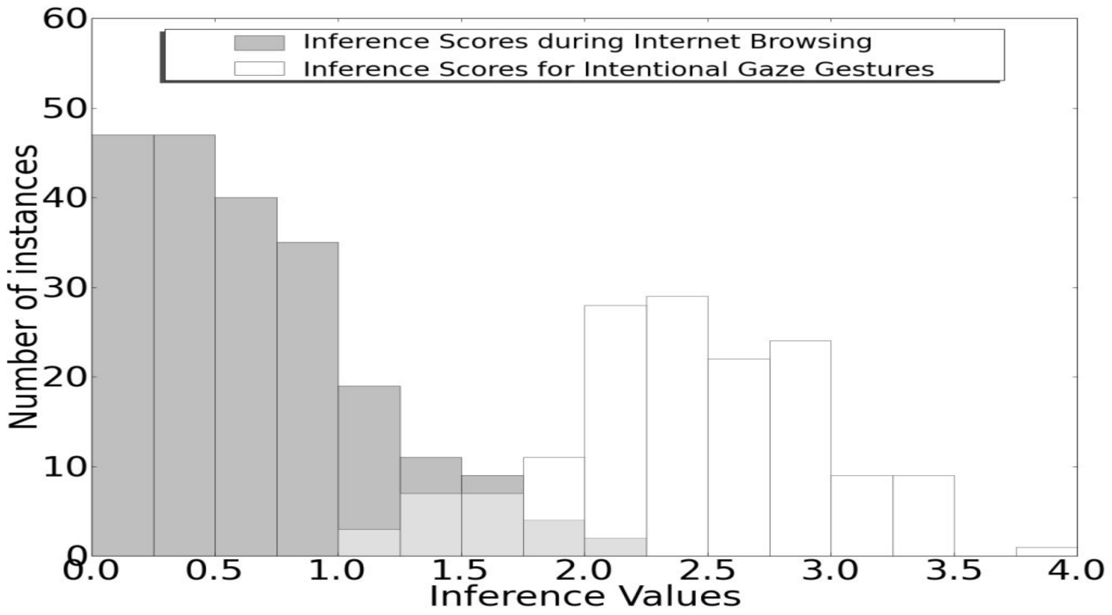


Figure 10.14: **Extended HTM recognition scores distribution for gaze gestures and standard gaze activity while browsing the Internet.** The dark grey bars correspond to the distribution of inference values generated by our extended HTM fed with normal gaze activity generated during standard user-computer interaction. The white bars correspond to the distribution of inference values generated by the extended HTM algorithm when fed with consciously performed gaze gestures. The light grey areas correspond to a minimal region of overlap between both distributions.

positives (Figure 10.8) users reported to prefer not to use fixations because of the faster nature and less stressful condition of not using dwell-activated fixations to signal the start and end of a gesture. The dwell threshold time used in our experiment was fixed. In a real scenario, users should be able to set their preferred dwell time to signal the beginning and end of a gesture.

10.5 Conclusion

In this work we have proposed and extension of the traditional HTM paradigm to improve its performance in the real time recognition of saccadic gaze gestures generated with a low cost and relatively noisy open source eye tracking system. The hardware/software approach used in this work has used extremely low cost hardware and software components, making it easily adoptable for all types of users regardless of financial situation.

The two dimensions used to compare the traditional and extended HTM methods, recognition accuracy of the algorithm and false positives rate show that the extended HTM method outperforms significantly traditional HTMs by being less error prone while preventing accidental algorithmic recognition of unintentional gaze gestures in the context of real time gaze computer interaction.

The motivation for an extended HTM was two-fold. First we wanted to adapt HTM algorithms to problems where the spatio-temporal structure of instances slowly unfolds over time. Second, we wanted to build a system able to robustly recognize gaze gestures in real time. The purpose of such system would be to constitute a communication channel between humans and computers, targeting specifically either people with disabilities or environments where traditional input channels are not suitable or could be augmented.

Last but not least, our extended HTM approach is not specifically designed to deal with gaze gestures specifically, since it is highly independent of hardware and preprocesssing of data. In fact, it is applicable to a wide array of problems whose instances consist of a spatio-temporal structure unfolding in time. Hence, it possesses a high degree of flexibility and can be easily adapted to other machine learning applications where the patterns to be learned are multi-variable time-series unfolding slowly over time.

Part III

Summary and Conclusions

Chapter 11

Summary of Results

In this thesis we have studied and extended an emerging computational paradigm known as hierarchical temporal memory (HTM). In particular we have developed an extension that optimizes the performance of the algorithm for multivariable time series. The specific landmarks and results achieved by this thesis are the following:

- Chapter 6: We have developed an in-house version of HTM by implementing the theoretical principles of the paradigm. We have tested our in-house version on a simple two dimensional image recognition problem to replicate the results of the authors of the theory and as a proof of principle for the validity of our implementation. After training the system with movies of the images moving along the vertical and horizontal directions and presented over different discrete sizes, the system became invariant to translations and size variations. We achieved recognition performance of up to 93% accuracy on a dataset of 48 2-dimensional binary images that underwent distortions, translations, occlusions, noise and slight rotations (The code and data sets employed in this work are available at `http://www.ii.uam.es/~gnb/drfthesis/htmimagerecognition.rar`).

- Chapter 7: We have tested our in-house version, and the standard HTM release from Numenta on a problem of Australian Sign Language recognition using data capture by a data glove. The results of these tests reflected the limitations of HTMs to deal with problems where instances unfold slowly over time. We have developed an extension of the HTM theory to optimize its performance on multivariable time series. Our extension consists on a customized top node that is able to store sequences of input data and compare them using the Needleman-Wunsch algorithm of dynamic programming for sequence alignment. Dynamic programming allows time warping thus providing the algorithm with the ability to recognize sequence generated at different speeds. Our extended HTM algorithm reaches a recognition performance of 91% on a dataset of 95 categories of Australian sign language in comparison to the 61% achieved by traditional HTM. Each category in the vocabulary, was composed of 27 instances, 24 of which were used for training the system and three for testing its inference abilities. Recognition accuracy was similar to well established methods in the literature for sign language recognition such as hidden Markov models and Metafeatures Tclasses. Our algorithm outperform methods in the literature on aspects such as the requirement of a low number of training instances, tolerance to increasing vocabulary sizes, degree of supervision requirements during training, lack of features models to detect and lack of language or grammar models (The code and data sets employed in this work are available at `http://www.ii.uam.es/~gnb/drfthesis/htmslr.rar`).

- Chapter 8: We have also explored a new interaction paradigm with computers through the usage of gaze gestures tracked by low cost and open source eye tracking technology. Due to the spatio-temporal structure that compromises a gaze gesture, HTMs that are specifically designed to capture the spatio-temporal structure of the parameter space that are exposed to during training seemed an appropriate machine learning algorithm to tackle the problem. We have determined that offline gaze gesture recognition is relatively simple to be carried out by traditional HTM algorithms. Real time recognition of gaze gestures is however more challenging due to the difficulty to discriminate consciously performed gaze gestures from standard gaze activity. (The code and data sets employed in this work are available at `http://www.ii.uam.es/~gnb/drfthesis/htmggroffline.rar`).

- Chapter 9: We have explore different modalities of performing a gaze gesture: gliding gestures and saccadic gestures. We have found out through a user study the superiority of the saccadic gestures modality both in terms of user preferences as well as recognition performance and time requirements to carry out the gesture. We have compared the performance of different algorithms for the real time recognition of gaze gestures in head mounted setups. We have illustrated the limitations of traditional HTM to carry out this task. We have also shown how a very simple sequence alignment algorithm using dynamic programming performs very well and it is very light on computer resources for the recognition of gaze gestures in real time reaching up to 95% recognition accuracy in comparison to the 74% accuracy reached by the traditional HTMs (The code and data sets employed in this work are available at `http://www.ii.uam.es/~gnb/drfthesis/htmggr.rar`).

- Chapter 10: We have used our extended HTM algorithm for the successful recognition of real time gaze gestures using a low cost remote setup reaching up to 98% recognition accuracy. The ability of our algorithms to successfully distinguish in real time consciously perform gaze gestures from normal gaze activity during usage of a computer shows the potential of these new form of human computer interaction. Gaze gestures possess several advantages: they do not occupy screen real state, they do not require calibration process, they circumvent the Midas touch problem pervasive to gaze interaction, they eliminate the necessity of dwell time and they increased the interaction vocabulary of gaze interaction (The code and data sets employed in this work are available at `http://www.ii.uam.es/~gnb/drfthesis/htmggr.rar`).

Chapter 12

Conclusions

In this thesis, we have explored an emerging pattern recognition paradigm known as Hierarchical Temporal Memory (HTM). The approach is a connectionist, Bayesian and bio-inspired set of algorithms which try to model the topology and functionality of the neocortex, namely a hierarchical organization, spatio-temporal coding of the features to be learned and the usage of time as a supervisor to cluster instances into a category.

HTMs are structured as a hierarchical network of individual units called nodes. Each node implements the same learning algorithm. A node learns the spatio-temporal patterns of its inputs and clusters them into "causes". A Node's output at one level in the hierarchy is the input to a node in a higher level of the hierarchical structure. Nodes at the bottom level of the hierarchy receive their inputs from sensory systems.

The usage of a hierarchy is important for generalization and storage efficiency. The hierarchy also matches to spatial and temporal hierarchy of the real world [Hawkins, 2006]. A technique of Bayesian theory known as belief propagation [Pearl, 1988] searches for the optimal set of mutually compatible beliefs among nodes. The hierarchical representation also affords a mechanism for attention.

Although, in this thesis, we have just focused on the first two capabilities of the following list, HTMs are able to [Hawkins, 2006]:

1. Discover causes in the world.
2. Infer causes of novel input.
3. Make predictions.
4. Direct behavior.

Hierarchical temporal memory works very well on problems where the spatial structure is fully represented at any time instant. Multivariable time series, where the structure of instances representing categories unfold over time represent a challenging problem for existing HTM Networks.

We have presented an extension of Hierarchical Temporal Memory, by adding and extended top node that is able to store complete sequences of its input. These sequences represent the whole spatio-temporal codification of an instance representing a category from the whole set of categories to be learnt. This extended top node uses the Needleman-Wunsch algorithm to perform sequence alignment and measure sequence similarity in order to cluster sequences belonging to the same category together. The Needleman-Wunsch algorithm is just an instantiation of the wider optimization concept of dynamic programming.

As proof of principle of our extended HTM system, we applied it to the recognition of sign language from data captured with an electronic data glove. This type of glove provided information about hands position in space, wrist angle and finger position as the hands “utters” a word from Sign Language. Our extended HTM matched the recognition performance of state of the art machine learning approaches for the sign language recognition problem: namely, Hidden Markov Models and Metafeatures Tclasses. In some performance aspects, the extended HTM even outperformed existing approaches.

In this thesis we have also explored a new interaction paradigm in the realm of human computer interaction or HCI, namely we have explored the usage of gaze gestures to control a computer within the realm of gaze tracking technology.

Gaze tracking technology is very convenient for substituting the mouse and using gaze as a pointing device. However, the problem of distinguishing whether gaze over an object targets examining the object or interacting with it pervades the field giving rise to the term “Midas touch problem” to refer to the difficulty of discriminating between interaction for examination purposes from intention to generate switching actions.

In our work, we have explored the usage of gaze gestures to control a computer. Offline recognition of gaze gestures is a trivial task with traditional HTM and many other types of machine learning classifiers, with most algorithms getting very consistent and robust results. However, gaze gesture recognition in real time is more challenging. This has to do with the difficulty that intentional gaze gestures have to be distinguish from standard gaze activity during normal gaze computer interaction. Our extended HTM however, is able to handle the spatio-temporal data structure of gaze gestures and properly discriminate them from other types of gaze activity. We obtain high accuracy rates, low false positives and a reportedly ease of use by users

of this innovative interaction paradigm. The ability to recognize gaze gestures in real time using low cost eye tracking technology opens interesting interaction possibilities for the realms of new electronic devices such as smartphones or projected displays.

In this work we have explored and extended a bioinspired pattern recognize engine. We believe it is important to study and apply concepts from information processing extracted from neuroscience to recognition problems where machine learning approaches have traditionally struggle. Beyond the scope of the results of this thesis, the extended HTM can be applied to any type of multivariable time series whose instances are composed of spatio-temporal structure unfolding over time.

Chapter 13

Conclusiones

En esta tesis, hemos explorado un paradigma emergente de reconocimiento de patrones conocido como Memoria Jerárquica Temporal (HTM por sus siglas en inglés). Este paradigma es un algoritmo conexionista, con bases Bayesianas y bioinspirado que trata de modelar la topología y la funcionalidad de la neocorteza. El algoritmo utiliza una organización jerárquica, codificación espacio-temporal de los patrones a aprender y emplea la información temporal para agrupar instancias en grupos representativos de una categoría.

Las HTMs están estructurados como una red jerárquica de unidades individuales llamadas nodos. Cada nodo implementa el mismo algoritmo de aprendizaje. Un nodo aprende el patrón espacio temporal de sus inputs y los agrupa en "causas". El output de un nodo a un nivel de la jerarquía es el input del nodo en un nivel superior de la estructura jerárquica. Los nodos en el nivel inferior de la jerarquía reciben sus inputs de sistemas sensores.

El uso de una jerarquía es importante para la generalización y el almacenamiento eficiente. La jerarquía también se corresponde con la jerarquía espacio-temporal del mundo real. Una técnica de teoría Bayesiana conocida como "propagación de creencias" busca el grupo óptimo de creencias mutualmente compatibles entre los nodos. La representación jerárquica además también permite un mecanismo para la atención.

A pesar de que en esta tesis nos hemos centrado sólo en las dos primeras capacidades de la lista siguiente, los HTM son capaces de:

1. Descubrir causas
2. Inferir causas de inputs novedosos
3. Hacer predicciones
4. Dirigir comportamiento

La memoria jerárquica temporal funciona muy bien en problemas donde la estructura espacial está totalmente representada en cualquier instante de tiempo. Las series temporales multivariables donde la estructura espacial de las instancias que representan a categorías se desarrollan en el tiempo suponen un reto para las HTM tradicionales.

En esta tesis hemos presentado una extensión de la memoria jerárquica temporal, añadiendo un nodo superior extendido que es capaz de almacenar secuencias completas de inputs que componen en su totalidad una categoría. Estas secuencias implementan la codificación espacio temporal completa de una instancia que representa a una categoría del conjunto de categorías a aprender. El nodo superior extendido utiliza el algoritmo de Needleman-Wunsch para realizar alineamiento de secuencias y medir la similitud entre secuencias para agrupar las secuencias que pertenezcan a la misma categoría. El algoritmo de Needleman-Wunsch es simplemente una instancia de un concepto de optimización mas amplio denominado programación dinámica.

Como prueba de concepto de nuestro sistema HTM extendido, hemos estudiado el reconocimiento de lenguaje de signos con datos capturados con un guante electrónico. Este tipo de guante ofrece información sobre la posición de las manos en el espacio, los ángulos de rotación de la muñeca y la posición de los dedos mientras la mano “pronuncia” una palabra de lenguaje de signos. Nuestro sistema HTM extendido alcanza el rendimiento, en términos de reconocimiento, de los métodos de aprendizaje automático que representan el estado del arte para el problema del reconocimiento de lenguaje de signos: Hidden Markov Models y las Metafeatures Tclasses. En algunos aspectos relacionados con la comparación de rendimiento, nuestro sistema HTM extendido supera a dichos métodos.

En esta tesis también hemos explorado un nuevo paradigma de interacción con máquinas electrónicas consistente en el uso de gestos pupilares dentro del campo de seguimiento de la mirada.

El seguimiento de la mirada es muy conveniente para sustituir al ratón como un sistema de puntero. A pesar de todo, esta tecnología tiene un problema a la hora determinar si el usuario utiliza su mirada para examinar un objeto en el interfaz o para interactuar con él.

En nuestro trabajo, hemos explorado el uso de gestos pupilares para controlar un ordenador. El reconocimiento de gestos pupilares offline es trivial tanto para HTM tradicionales como para otros tipos de clasificadores automáticos con muchos algoritmos obteniendo resultados consistentes y robustos. Sin embargo, el reconocimiento

de gestos pupilares en tiempo real es más difícil. Esto es debido a la dificultad de distinguir gestos pupilares de otro tipo de movimientos pupilares realizados durante la interacción normal con un ordenador. Nuestro sistema HTM extendido sin embargo es capaz de lidiar con la estructura de datos espaciotemporal de los gestos pupilares y discriminar correctamente estos de otro tipo de movimientos pupilares. En nuestros experimentos, obtenemos altas tasas de reconocimiento, bajo número de falsos positivos y retroalimentación positiva por parte de los usuarios que afirman estar satisfechos con esta nueva forma de interacción. La capacidad de reconocer gestos pupilares en tiempo real utilizando un sistema de seguimiento pupilar de bajo coste abre interesantes paradigmas de interacción para el campo de los dispositivos electrónicos tales como los smartphones, las tabletas o los displays proyectados.

En este trabajo, hemos explorado y extendido un algoritmo bioinspirado de reconocimiento de patrones. Creemos que es importante estudiar y aplicar conceptos de procesamiento de la información extraídos de la neurociencia a problemas de reconocimiento automático donde los algoritmos tradicionales tienen habitualmente problemas. Más allá de los resultados de esta tesis, el sistema HTM extendido propuesto puede ser aplicado a cualquier tipo de serie temporal multivariable cuyas instancias están compuestas de estructuras espacio-temporales que se desarrollan el tiempo.

Bibliography

M. Abeles. *Corticonics: neural circuits of the cerebral cortex.* Henry Holt and Company, 2004.

M. AL-Rousan, K. Assaleh, and A. Tala'a. Video-based signer-independent arabic sign language recognition using hidden markov models. *Applied Soft Computing*, 9(3):990 – 999, 2009. ISSN 1568-4946. doi: DOI:10.1016/j.asoc.2009.01.002. URL `http://www.sciencedirect.com/science/article/B6W86-4VH4D92-2/2/d6ad78c1a0f8461390422b0f63048336`.

A. Auyeung, P.B. May, J.W. Goh, and B.R. Sastry. Temporal requirements of associative short-term potentiation in ca1 neurons of rat hippocampus. *Neuroscience Letters*, 79(1-2):117–122, 1987.

M. Bear, B. Connors, M. Paradiso, M.F. Bear, B.W. Connors, and M.A. Paradiso. *Neuroscience: Exploring the Brain (Book with CD-ROM).* Lippincott Williams & Wilkins, second edition, March 2002. ISBN 0781739446. URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0781739446`.

N. Bee and E. André. Writing with your eye: A dwell time free writing system adapted to the nature of human eye gaze. In *Proceedings of the 4th IEEE tutorial and research workshop on Perception and Interactive Technologies for Speech-Based Systems*, pages 111–122. Springer-Verlag, 2008.

C.J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, 1998. URL `http://dblp.uni-trier.de/rec/bibtex/journals/datamine/Burges98`.

H.F. Chua, J.E. Boland, and R.E. Nisbett. Cultural variation in eye movements during scene perception. *Proceedings of the National Academy of Sciences of the United States of America*, 102(35):12629–12633, August 2005. doi: 10.1073/pnas.0506162102. URL `http://dx.doi.org/10.1073/pnas.0506162102`.

H. Davson. *Physiology of the eye.* Macmillan, 1990. ISBN 9780333458600. URL `http://books.google.com/books?id=zSfEQgAACAAJ`.

P. Dayan, G.E. Hinton, R.M. Neal, and R.S. Zemel. The Helmholtz Machine. *Neural Computation*, 7(5):889–904, December 2010. doi: 10.1162/neco.1995.7.5.889. URL `http://dx.doi.org/10.1162/neco.1995.7.5.889`.

A. De Luca, R. Weiss, and H. Drewes. Evaluation of eye-gaze interaction methods for security enhanced pin-entry. In *Proceedings of the 19th Australasian conference on Computer-Human Interaction: Entertaining User Interfaces*, OZCHI '07, pages 199–202, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-872-5. doi: http://doi.acm.org/10.1145/1324892.1324932. URL `http://doi.acm.org/10.1145/1324892.1324932`.

L. Dipietro, A.M. Sabatini, and P. Dario. A survey of glove-based systems and their applications. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(4):461 –482, 2008. ISSN 1094-6977. doi: 10.1109/TSMCC.2008.923862.

R.J. Douglas and K.A.C. Martin. Neuronal circuits of the neocortex. *Annual Review of Neuroscience*, 27(1):419–451, 2004. doi: 10.1146/annurev.neuro.27.070203.144152. URL `http://arjournals.annualreviews.org/doi/abs/10.1146/annurev.neuro.27.070203.144152`.

H. Drewes. *Eye Gaze Tracking for Human Computer Interaction.* PhD thesis, Ludwig-Maximilians-Universität München, 2010.

H. Drewes and A. Schmidt. Interacting with the computer using gaze gestures. In *Proceedings of the 11th IFIP TC 13 international conference on Human-computer interaction - Volume Part II*, INTERACT'07, pages 475–488, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 3-540-74799-0, 978-3-540-74799-4. URL `http://portal.acm.org/citation.cfm?id=1778331.1778385`.

A.T. Duchowski. *Eye Tracking Methodology: Theory and Practice.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007. ISBN 1846286085.

R. Ennals. Pamela mccorduck and a.k. peters (eds): Machines who think: 25th anniversary update: Natick, massachusetts, 2004, isbn 1-56881-205-1. *AI Soc.*, 18:382–383, October 2004. ISSN 0951-5666. doi: 10.1007/s00146-004-0307-0. URL `http://portal.acm.org/citation.cfm?id=1030813.1030816`.

G. Fang and W. Gao. A srn/hmm system for signer-independent continuous sign language recognition. *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on*, pages 312 –317, may. 2002. doi: 10.1109/AFGR.2002.1004172.

D.J Felleman and D.C. Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, 1(1):1–47, 1991. URL `http://www.ncbi.nlm.nih.gov/pubmed/1822724`.

S. Fine. The hierarchical hidden markov model: Analysis and applications. In *Machine Learning*, volume 32, pages 41–62, 1998. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.33.2863`.

D. George and J. Hawkins. A hierarchical bayesian model of invariant pattern recognition in the visual cortex. In *Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference on*, volume 3, pages 1812–1817 vol. 3, July-4 Aug. 2005. doi: 10.1109/IJCNN.2005.1556155.

D. George and J. Hawkins. Towards a mathematical theory of cortical micro-circuits. *PLoS Comput Biol*, 5(10):e1000532, 10 2009. doi: 10.1371/journal.pcbi.1000532. URL `http://dx.doi.org/10.1371%2Fjournal.pcbi.1000532`.

D. George and B. Jarosy. The HTM learning algorithms. Technical report, Numenta, 2007.

L. Gupta and M. Suwei. Gesture-based interaction and communication: automated classification of hand gesture contours. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 31(1):114 –120, February 2001. ISSN 1094-6977. doi: 10.1109/5326.923274.

D.W. Hansen and R. Hammoud. Boosting particle filter-based eye tracker performance through adapted likelihood function to reflexions and light changes. *Advanced Video and Signal Based Surveillance, IEEE Conference on*, 0:111–116, 2005. doi: http://doi.ieeecomputersociety.org/10.1109/AVSS.2005.1577252.

D.W. Hansen and Q. Ji. In the Eye of the Beholder: A Survey of Models for Eyes and Gaze. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32 (3):478–500, 2010. ISSN 0162-8828. doi: 10.1109/TPAMI.2009.30. URL `http://dx.doi.org/10.1109/TPAMI.2009.30`.

J. Hawkins. *On Intelligence.* Cambridge University Press, 2004.

J. Hawkins. Hierarchical temporal memory, concepts, theory, and terminology. Technical report, Numenta, 2006.

S. Haykin. *Neural Networks and Learning Machines.* Prentice Hall, 3 edition, November 2008. ISBN 0131471392. URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0131471392`.

D.O. Hebb. *The Organization of Behavior: A Neuropsychological Theory.* Wiley, New York, new edition edition, June 1949. ISBN 0805843000. URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0805843000`.

H. Heikkilä. Speed and accuracy of gaze gestures. *SciencesNew York*, 3(2):1–14, 2009. URL `http://www.jemr.org/download/pictures/68/w9x29ckly31c07g14zw195mxaasba2/heikkila-jemr-2009.pdf`.

H. Heikkilä. *Design and implementation of an object-based drawing application for gaze control.* PhD thesis, University of Tampere, 2011.

C. Hennessey, B. Noureddin, and P. Lawrence. Fixation precision in high-speed noncontact eye-gaze tracking. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 38(2):289 –298, 2008. ISSN 1083-4419. doi: 10.1109/TSMCB.2007.911378.

G.E. Hinton and T.J. Sejnowski. Learning and relearning in Boltzmann machines. pages 282–317, 1986. URL http://portal.acm.org/citation.cfm?id=104279.104291.

Y.C. Ho and D.L. Pepyne. Simple explanation of the no free lunch theorem of optimization. *Cybernetics and Sys. Anal.*, 38:292–298, March 2002. ISSN 1060-0396. doi: 10.1023/A:1016355715164. URL http://portal.acm.org/citation.cfm?id=592657.592825.

M.E. Hoff. *Learning Phenomena in Networks of Adaptive Switching Circuits.* PhD thesis, Technical Report 1554-1, Stanford Electron. Labs., Stanford, CA, July 1962.

E.J. Holden, R. Owens, and G.R. Geoffrey. Adaptive fuzzy expert system for sign recognition. Technical report, in Proceedings of the International Conference on Signal and Image Processing SIP'2000. 1999: Las Vegas, 1999.

E.J Holden, G. Lee, and R. Owens. Australian sign language recognition. *Machine Vision and Applications*, 16:312–320, 2005. ISSN 0932-8092. URL http://dx.doi.org/10.1007/s00138-005-0003-1. 10.1007/s00138-005-0003-1.

J.J. Hopfield and D.W. Tank. Neural computation of decisions in optimization problems. *Biological Cybernetics*, 52(3):141–152, July 1985. doi: 10.1007/BF00339943. URL http://dx.doi.org/10.1007/BF00339943.

I. Infantino, R. Rizzo, and S. Gaglio. A system for sign language sentence recognition based on common sense context. In *Computer as a Tool, 2005. EUROCON 2005.The International Conference on*, volume 2, pages 1421 –1424, 2005. doi: 10.1109/EURCON.2005.1630228.

P. Isokoski. Text input methods for eye trackers using off-screen targets. In *ETRA '00: Proceedings of the 2000 symposium on Eye tracking research & applications*, pages 15–21, New York, NY, USA, 2000. ACM. ISBN 1-58113-280-8. doi: http://doi.acm.org/10.1145/355017.355020.

H. Istance, S. Vickers, and A. Hyrskykari. Gaze-based interaction with massively multiplayer on-line games. In *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, CHI EA '09, pages 4381–4386, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-247-4. doi: http://doi.acm.org/10.1145/1520340.1520670. URL http://doi.acm.org/10.1145/1520340.1520670.

R.J.K. Jacob. The use of eye movements in human-computer interaction techniques: what you look at is what you get. *ACM Trans. Inf. Syst*, 9(2):152–169, 1991.

M.W. Kadous. Australian sign language signs data set. Technical report, UCI Machine Learning Repository, http://archive.ics.uci.edu/ml/datasets/.

M.W. Kadous. *Temporal Classification: Extending the Classification Paradigm to Multivariate Time Series*. PhD thesis, The University of New South Wales, 2002.

E.R Kandel, J.H. Schwartz, and T.M. Jessell. *Principles of Neural Science*, volume 3. McGraw-Hill, 2000. URL `http://www.amazon.com/Principles-Neural-Science-Eric-Kandel/dp/0838577016`.

T. Kapuscinski and M. Wysocki. Using hierarchical temporal memory for recognition of signed polish words. In *Computer Recognition Systems: Advances in Intelligent and Soft Computing (Kurzynski M., Wozniak M, Eds.)*. Springer-Verlag Berlin / Heidelberg 2009, pp 355-362, 2009.

T. Kohonen. *Self-organized formation of topologically correct feature maps*, pages 509–521. MIT Press, Cambridge, MA, USA, 1988. ISBN 0-262-01097-6. URL `http://portal.acm.org/citation.cfm?id=65669.104428`.

A. Krogh. What are artificial neural networks? *Nature biotechnology*, 26(2):195–197, February 2008. ISSN 1546-1696. doi: 10.1038/nbt1386. URL `http://dx.doi.org/10.1038/nbt1386`.

J. Lampinen and A. Vehtari. Bayesian approach for neural networks - review and case studies. *Neural Networks*, 14(3):257–274, 2001. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.16.9915`.

R. Latorre, F.B. Rodriguez, and P. Varona. Signature neural networks: Definition and application to multidimensional sorting problems. *Neural Networks, IEEE Transactions on*, 22(1):8 –23, jan. 2011. ISSN 1045-9227. doi: 10.1109/TNN.2010.2060495.

G. Lavee, E. Rivlin, and M. Rudzsky. Understanding video events: A survey of methods for automatic interpretation of semantic occurrences in video. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 39(5):489 –504, September 2009. ISSN 1094-6977. doi: 10.1109/TSMCC.2009.2023380.

Y. Lecun, S. Chopra, R. Hadsell, F.J. Huang, and M.A. Ranzato. *A Tutorial on Energy-Based Learning.* MIT Press, 2006.

D. Li, J. Babcock, and D.J. Parkhurst. openEyes: a low-cost head-mounted eye-tracking solution. In *Proceedings of the 2006 symposium on Eye tracking research & applications*, pages 95–100, San Diego, California, 2006. ACM. ISBN 1-59593-305-0.

R.H. Liang and Ouhyoung. M. A real-time continuous gesture recognition system for sign language. *Automatic Face and Gesture Recognition, 1998. Proceedings.*

Third IEEE International Conference on, pages 558 –567, apr. 1998. doi: 10.1109/AFGR.1998.671007.

D. Mackay. Dasher - an efficient keyboard alternative. *ACNR*, 3(2):24, 2003. URL `http://www.acnr.co.uk/contents3-2.htm`.

Jennifer Mankoff and Gregory D. Abowd. Cirrin: A word-level unistroke keyboard for pen input. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'98)*, pages 213–214. ACM Press, November 1998.

W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 52:99–115, 1990. ISSN 0092-8240. URL `http://dx.doi.org/10.1007/BF02459570`. 10.1007/BF02459570.

M.L. Minsky and S.A. Papert. *Perceptrons.* The MIT Press, December 1969. ISBN 0262631113. URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0262631113`.

S. Mitra and T. Acharya. Gesture recognition: A survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(3):311 –324, May 2007. ISSN 1094-6977. doi: 10.1109/TSMCC.2007.893280.

M. Mohandes, S.I. Quadri, and M.D. King. Arabic sign language recognition an image-based approach. In *Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 21st International Conference on*, volume 1, pages 272 –276, May 2007. doi: 10.1109/AINAW.2007.98.

E. Mollenbach. *Selection Strategies in Gaze Interaction.* PhD thesis, Loughborough University, 2010.

E. Mollenbach, J.P. Hansen, and Alastair G. Lillholm, M.G. Single stroke gaze gestures. In *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, CHI '09, pages 4555–4560, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-247-4. doi: http://doi.acm.org/10.1145/1520340.1520699. URL `http://doi.acm.org/10.1145/1520340.1520699`.

E. Mollenbach, M. Lillholm, A. Gail, and J.P. Hansen. Single gaze gestures. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*, ETRA '10, pages 177–180, 2010.

V. Mountcastle. An organizing principle for cerebral function: the unit model and the distributed system. In G. Edelman and V. Mountcastle, editors, *The Mindful Brain.* MIT Press, Cambridge, Mass., 1978.

S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443 – 453, 1970. ISSN 0022-2836. doi: DOI:10.1016/0022-2836(70)90057-4. URL `http://www.sciencedirect.com/science/article/B6WK7-4DN8W3K-7X/2/0d99b8007b44cca2d08a031a445276e1`.

J. Newton and M. Sur. Rewiring cortex: Functional plasticity of the auditory cortex during development. In Josef Syka and Michael M. Merzenich, editors, *Plasticity and Signal Representation in the Auditory System*, pages 127–137. Springer US, 2005. ISBN 978-0-387-23181-5. URL `http://dx.doi.org/10.1007/0-387-23181-1_11`.

Numenta. Hierarchical temporal memory - comparison with existing models. Technical report, Numenta, 2006a.

Numenta. Problems that fit htm. Technical report, Numenta, 2006b.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988. ISBN 1558604790.

P. Qvarfordt and S. Zhai. Conversing with the user based on eye-gaze patterns. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 221–230, New York, NY, USA, 2005. ACM. ISBN 1-58113-998-5. doi: http://doi.acm.org/10.1145/1054972.1055004.

M.I. Rabinovich, P. Varona, A.I. Selverston, and Henry D. I. Abarbanel. Dynamical principles in neuroscience. *Reviews of Modern Physics*, 78(4):1213+, 2006. doi: 10.1103/RevModPhys.78.1213. URL `http://dx.doi.org/10.1103/RevModPhys.78.1213`.

B.S. Reddy and O.A. Basir. Concept-based evidential reasoning for multimodal fusion in human-computer interaction. *APPLIED SOFT COMPUTING*, 10(2): 567–577, MAR 2010. ISSN 1568-4946. doi: {10.1016/j.asoc.2009.08.026}.

M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex, 1999. URL `http://www.cog.brown.edu/~{}tarr/pdf/RiPo99.pdf`.

D. Rios. Artificial intelligence, 2007. URL `http://www.learnartificialneuralnetworks.com`.

B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, January 1996. ISBN 0521460867. URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0521460867`.

D.A. Robinson. The mechanics of human smooth pursuit eye movement. *The Journal of Physiology*, 180(3):569–591, October 1965. doi: VL-180. URL `http://jphysiol.highwire.org/content/180/3/569.short`.

F.B. Rodríguez and R. Huerta. Analysis of perfect mappings of the stimuli through neural temporal sequences. *Neural Netw.*, 17(7):963–973, 2004. ISSN 0893-6080. doi: http://dx.doi.org/10.1016/j.neunet.2003.12.003.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

D. Rozado, F.B. Rodriguez, and P. Varona. Gliding and saccadic gaze gesture recognition in real time. *Submitted*, 2010a.

D. Rozado, F.B. Rodriguez, and P. Varona. Optimizing hierarchical temporal memory for multivariable time series. In Konstantinos Diamantaras, Wlodek Duch, and Lazaros Iliadis, editors, *Artificial Neural Networks - ICANN 2010*, volume 6353 of *Lecture Notes in Computer Science*, pages 506–518. Springer Berlin / Heidelberg, 2010b. URL `http://dx.doi.org/10.1007/978-3-642-15822-3_62`. 10.1007/978-3-642-15822-3-62.

D. Rozado, F.B. Rodriguez, and P. Varona. Low cost remote gaze gesture recognition in real time. *Submitted*, 2011a.

D. Rozado, F.B. Rodriguez, and P. Varona. Gaze gesture recognition with hierarchical temporal memory networks. In Joan Cabestany, Ignacio Rojas, and Gonzalo Joya, editors, *Advances in Computational Intelligence*, volume 6691 of *Lecture Notes in Computer Science*, pages 1–8. Springer Berlin / Heidelberg, 2011b.

D. Rozado, F.B. Rodriguez, and P. Varona. Extending the hierarchical temporal memory paradigm for sign language recognition. *Submitted*, 2011c.

S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall series in artificial intelligence. Prentice Hall, 2 edition, December 2002. ISBN 0137903952. URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0137903952`.

J. San Agustin. *Off-the-Shelf Gaze Interaction*. PhD thesis, IT University of Copenhagen, 2010.

J. San Agustin, H. Skovsgaard, E. Mollenbach, M. Barret, M. Tall, D.W. Hansen, and J.P. Hansen. Evaluation of a low-cost open-source gaze tracker. In *ETRA '10: Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*, pages 77–80, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-994-7. doi: http://doi.acm.org/10.1145/1743666.1743685.

A.K. Sarkaleh, A. Karami, and B. Zanj. Persian sign language (PSL) recognition using wavelet transform and neural networks. *Expert Systems with Applications*, August 2010. ISSN 09574174. doi: 10.1016/j.eswa.2010.08.056. URL `http://dx.doi.org/10.1016/j.eswa.2010.08.056`.

G.M. Shepherd. *The synaptic organization of the brain*, volume 5th ed. Oxford University Press, 2004. URL `http://www.amazon.com/dp/019515956X`.

N.E. Sondak and V.K. Sondak. Neural networks and artificial intelligence. *SIGCSE Bull.*, 21(1):241–245, 1989. ISSN 0097-8418. doi: 10.1145/65294.71221. URL `http://dx.doi.org/10.1145/65294.71221`.

D. Specht. Probabilistic neural networks. *Neural Networks*, 3(1):109–118, 1990. ISSN 08936080. doi: 10.1016/0893-6080(90)90049-Q. URL `http://dx.doi.org/10.1016/0893-6080(90)90049-Q`.

T. Starner, J. Weaver, and A. Pentland. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:1371–1375, 1998.

P. Vamplew. Recognition of sign language gestures using neural networks. In *Australian Journal of Intelligent Information Processing Systems*, pages 94–102, 1998.

S. Vickers, H.O Istance, R. Bates, and A. Hyrskykari. Snap clutch, a moded approach to solving the midas touch problem. In *ETRA &039;08: Proceedings of the 2008 symposium on Eye Tracking Research Applications*, Savannah, GA, March 2008. ACM Press. URL `http://doi.acm.org/10.1145/1344471.1344523`.

S. Vickers, H. Istance, A. Hyrskykari, L. Immonen, and S. Mansikkamaa. Designing gaze gestures for gaming: an investigation of performance. In *Proceedings of the 2010 symposium on Eye Tracking Research &amp; Applications; ETRA 2010*, Austin, TX, 2010. ACM Press. URL `http://doi.acm.org/10.1145/1743666.1743740`.

L. Wiskott and T.J. Sejnowski. Slow feature analysis: unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, April 2002. ISSN 0899-7667. doi: http://dx.doi.org/10.1162/089976602317318938. URL `http://dx.doi.org/10.1162/089976602317318938`.

J.O. Wobbrock, J. Rubinstein, M. Sawyer, and A.T. Duchowski. Gaze-based creativity not typing but writing: Eye-based text entry using letter-like gestures. In *The 3rd Conference on Communication by Gaze Interaction - COGAIN 2007*, 2007.

J.O. Wobbrock, J. Rubinstein, M.W. Sawyer, and A.T. Duchowski. Longitudinal evaluation of discrete consecutive gaze gestures for text entry. In *Proceedings of the 2008 symposium on Eye tracking research &#38; applications*, ETRA '08, pages 11–18, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-982-1. doi: http://doi.acm.org/10.1145/1344471.1344475. URL `http://doi.acm.org/10.1145/1344471.1344475`.

X. Yao. A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems*, 4:539–567, 1993. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.52.9598`.

Y. Zhou, X. Chen, D. Zhao, H. Yao, and W. Gao. Adaptive Sign Language Recognition With Exemplar Extraction and MAP/IVFS. *IEEE Signal Processing Letters*, 17, March 2010. doi: 10.1109/LSP.2009.2038251.

The End