

Advances in ubiquitous sensing
applications for healthcare

Volume **Nine**

Series Editors:

Nilanjan Dey, Amira S. Ashour
Simon James Fong

ADVANCES IN UBIQUITOUS COMPUTING

CYBER-PHYSICAL SYSTEMS, SMART CITIES AND ECOLOGICAL MONITORING

Volume Editor:
Amy Neustein



Advances in Ubiquitous Computing

Cyber-Physical Systems, Smart
Cities and Ecological Monitoring

This page intentionally left blank

Advances in Ubiquitous Sensing
Applications for Healthcare

Advances in Ubiquitous Computing

Cyber-Physical Systems, Smart
Cities and Ecological Monitoring

Volume Nine

Volume Editor

Amy Neustein

*Founder and CEO, Linguistic Technology Systems,
Fort Lee, NJ, United States*

Series Editors

Nilanjan Dey

Amira S. Ashour

Simon James Fong



ELSEVIER



ACADEMIC PRESS

An imprint of Elsevier

Academic Press is an imprint of Elsevier
125 London Wall, London EC2Y 5AS, United Kingdom
525 B Street, Suite 1650, San Diego, CA 92101, United States
50 Hampshire Street, 5th Floor, Cambridge, MA 02139, United States
The Boulevard, Langford Lane, Kidlington, Oxford OX5 1GB, United Kingdom

© 2020 Elsevier Inc. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or any information storage and retrieval system, without permission in writing from the publisher. Details on how to seek permission, further information about the Publisher's permissions policies and our arrangements with organizations such as the Copyright Clearance Center and the Copyright Licensing Agency, can be found at our website: www.elsevier.com/permissions.

This book and the individual contributions contained in it are protected under copyright by the Publisher (other than as may be noted herein).

Notices

Knowledge and best practice in this field are constantly changing. As new research and experience broaden our understanding, changes in research methods, professional practices, or medical treatment may become necessary.

Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information, methods, compounds, or experiments described herein. In using such information or methods they should be mindful of their own safety and the safety of others, including parties for whom they have a professional responsibility.

To the fullest extent of the law, neither the Publisher nor the authors, contributors, or editors, assume any liability for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions, or ideas contained in the material herein.

Library of Congress Cataloging-in-Publication Data

A catalog record for this book is available from the Library of Congress

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library

ISBN: 978-0-12-816801-1

For information on all Academic Press publications
visit our website at <https://www.elsevier.com/books-and-journals>

Publisher: Mara Conner
Acquisitions Editor: Chris Katsaropoulos
Editorial Project Manager: John Leonard
Production Project Manager: Selvaraj Raviraj
Cover Designer: Alan Studholme

Typeset by SPi Global, India



Working together
to grow libraries in
developing countries

www.elsevier.com • www.bookaid.org

Contents

Contributors	xi
Foreword	xiii
Introduction.....	xv

PART 1 Wireless sensor networks and cyber-physical systems: New approaches and methods

CHAPTER 1 Routing protocols for wireless sensor networks: A survey	3
Bahae Abidi, Abdelillah Jilbab and Mohamed El Haziti	
1.1 Introduction.....	3
1.2 Application of WSNs.....	5
1.3 Characteristics and constraints of WSNs	6
1.4 Network topology of WSNs	7
1.4.1 Bus topology	7
1.4.2 Tree topology	7
1.4.3 Star topology.....	7
1.4.4 Mesh topology	7
1.5 Routing protocol for WSNs.....	8
1.5.1 Hierarchical routing protocol	8
1.5.2 Data-centric routing protocol	10
1.5.3 Location-based protocol	11
1.6 Conclusion.....	13
References.....	13
CHAPTER 2 Replay attack detection using excitation source and system features	17
Madhusudan Singh and Debadatta Pati	
2.1 Introduction.....	17
2.2 Replay speech signal	21
2.3 Features used for replay detection.....	22
2.3.1 LP residual-based implicit source features	22
2.3.2 Explicit source features	26
2.3.3 System features.....	27
2.4 Experimental study	28
2.4.1 Databases	28
2.4.2 Experimental setup	31

2.4.3	Evaluation process	33
2.4.4	Experimental results and discussion	34
2.5	Summary and future scope	40
	Acknowledgments	41
	References.....	41
CHAPTER 3 Hypergraph-based type theory for software development in a Cyber-Physical context		45
Nathaniel Christen		
3.1	Hub applications and gatekeeper code	48
3.1.1	Gatekeeper code	55
3.1.2	Fragile code	59
3.1.3	Core language vs. external tools	60
3.2	Case studies.....	63
3.2.1	How “Internet of Things” interoperability affects data modeling priorities.....	69
3.2.2	Linguistic case study	72
3.2.3	Proactive design.....	75
3.3	Directed Hypergraphs and generalized lambda calculus.....	77
3.3.1	Generalized lambda calculus.....	78
3.3.2	Directed Hypergraphs and “channel abstractions”	81
3.3.3	Channelized hypergraphs and RDF	85
3.3.4	Procedural input/output protocols via type theory.....	93
3.4	Modeling procedures via channelized hypergraphs.....	98
3.4.1	Initializing function-typed values.....	98
3.4.2	Dependent types and co-constructors.....	104
3.5	Channels and carriers.....	111
3.5.1	Carrier transfers	113
3.5.2	Channelized-type interpretations of larger-scale source code elements	118
3.6	Conclusion.....	126
	References.....	129
CHAPTER 4 A new method of power efficient speech transmission over 5G networks using new signaling techniques		139
Javaid A. Sheikh, Sakeena Akhtar, Arshid Iqbal Khan, Shabir A. Parah, G.M. Bhat and Amy Neustein		
4.1	Introduction	139
4.2	Related work	140
4.3	Ultra-filter bank multicarrier modulation.....	141
4.4	Space-time codes	143
4.4.1	Bit error rate analysis using Alamouti coding	144

4.4.2 Orthogonal space-time block code	146
4.4.3 Maximal ratio combining	147
4.5 Bit error rate (BER) analysis of BPSK modulated system.....	148
4.6 Bit error rate (BER) analysis of a QAM modulated system	149
4.7 Proposed technique	150
4.8 Simulation results	152
4.9 Conclusion.....	157
References.....	158

PART 2 Smart cities/smart homes/smart communities

CHAPTER 5 Study of robust language identification techniques for future smart cities..... 163

Ravi Kumar Vuddagiri, Krishna Gurugubelli,
Ramakrishna Thirumuru and Anil Kumar Vuppala

5.1 Introduction	163
5.2 Related works.....	165
5.3 Database	166
5.4 Baseline LID system.....	167
5.4.1 The i-vector-based LID system	167
5.4.2 LID system using DNN.....	167
5.4.3 LID system using DNN with attention	169
5.5 Performance of LID in mismatched environments.....	171
5.5.1 Language identification in noisy conditions	171
5.5.2 Language identification in a mobile environment.....	171
5.6 Proposed approaches for improving performance language identification in mismatched conditions	172
5.6.1 Improving the performance of LID systems by vowel region-based front-end system	172
5.6.2 Improving the performance of LID systems by enhancing the speech signals	177
5.6.3 Improving the performance of LID systems by using CL strategies	179
5.7 Conclusion and future scope	180
Acknowledgment	181
References.....	181

CHAPTER 6 Effective natural interaction with our sensorized smart homes..... 185

António Teixeira, Nuno Almeida, Maksym Ketsmur and
Samuel Silva

6.1 Introduction.....	185
6.2 Related work	187

6.2.1 Home sensors and actuators	187
6.2.2 High level handling of sensor networks	187
6.2.3 “Smart” appliances	188
6.2.4 Ubiquitous computing, interaction, and multidevice contexts	188
6.2.5 Human building interaction.....	189
6.2.6 Interaction with smart homes	190
6.2.7 Assistants	192
6.2.8 Design challenges in IoT and data contexts	195
6.3 An integrated vision for human-home interaction.....	196
6.4 Integrated ubiquitous distributed solution	
for a smart home	197
6.4.1 Smart home’s information structuring	198
6.4.2 Users and context	199
6.4.3 Home control	200
6.4.4 Multimodal interaction support.....	200
6.4.5 Smart home test bench	201
6.5 Ubiquitous human-home interaction	203
6.5.1 Multimodal interaction	203
6.5.2 Conversational capabilities.....	204
6.6 Illustrative scenarios	208
6.6.1 Chatting with the smart home	208
6.6.2 Sending an email to the home.....	212
6.6.3 Assisting the very young and the elderly.....	213
6.6.4 Multimodal interaction across rooms.....	214
6.7 Conclusion.....	216
Acknowledgment	217
References.....	217
CHAPTER 7 A study on the emotional state of a speaker in voice bio-metrics	223
K.N.R.K. Raju Alluri and Anil Kumar Vuppala	
7.1 Introduction	223
7.2 Emotional database	225
7.3 Baseline emotional SR system	225
7.4 Analysis of SR system performance in emotional conditions	227
7.4.1 Fundamental frequency	228
7.4.2 Strength of excitation	228
7.4.3 Energy of excitation	228
7.4.4 Duration	228

7.5	Proposed strategies for SR in emotional conditions	230
7.5.1	SR system with emotional UBM	230
7.5.2	SR system with emotional data.....	231
7.5.3	Selection of speaker model based on emotion recognition	232
7.6	Summary and conclusions	233
	Acknowledgments	234
	References.....	234
	Further reading	236

PART 3 Ecological monitoring

CHAPTER 8 Ubiquitous computing and biodiversity monitoring 239

Todor D. Ganchev

8.1	Marine biodiversity monitoring.....	240
8.2	Terrestrial biodiversity monitoring.....	242
8.2.1	The ARBIMON acoustics project	244
8.2.2	The AMIBIO project	247
8.3	Pest control.....	252
8.4	Health and disease transmission.....	254
8.5	Monitoring of urban ecosystems	255
8.6	Discussion and future trends	257
	References.....	257
	Further reading	258

CHAPTER 9 The use of WSN (wireless sensor network) in the surveillance of endangered bird species 261

Amira Boulmaiz, Noureddine Doghmane, Saliha Harize,
Nasreddine Kouadria and Djemil Messadeg

9.1	Introduction.....	261
9.2	Study area.....	264
9.3	Study bird species	266
9.3.1	The white-headed duck	266
9.3.2	The ferruginous duck.....	268
9.4	Measurement and data collection	270
9.5	Habitat monitoring and wildlife information gathering.....	271
9.5.1	Conventional observation method.....	271
9.5.2	Mobile devices.....	271
9.5.3	Fixed devices	272

9.5.4 Comparison	272
9.6 Wireless sensor networks	273
9.6.1 WSNs principles	273
9.6.2 Wireless sensor node structure.....	274
9.6.3 WSNs for wildlife monitoring	276
9.7 Methodology	276
9.7.1 Environmental noise	277
9.7.2 Limited computational capacity of wireless sensor nodes.....	279
9.8 Bird species recognition systems	280
9.8.1 Acoustic bird recognition system: Proposition 1.....	280
9.8.2 Acoustic bird recognition system: Proposition 2.....	286
9.8.3 Alternative acoustic features used in birdsong/speech recognition systems	291
9.9 Experimental evaluations.....	293
9.9.1 Data description (database) and experimental setup	293
9.9.2 Results and discussion	294
9.10 Conclusion.....	299
References.....	300
Further reading	306
Index	307

Contributors

Bahae Abidi

LRIT—CNRST URAC No. 29, Rabat IT Center, Faculty of Sciences, Mohammed V University in Rabat, Rabat, Morocco

Sakeena Akhtar

PG Department of Electronics and IT, University of Kashmir, Srinagar, India

K.N.R.K. Raju Alluri

Speech Processing Laboratory, LTRC, KCIS, International Institute of Information Technology, Hyderabad, India

Nuno Almeida

Department of Electronics Telecommunications and Informatics/IEETA, University of Aveiro, Aveiro, Portugal

G.M. Bhat

College of Engineering, University of Kashmir, Srinagar, India

Amira Boulmaiz

Electronics Department, Faculty of Engineering Sciences, LERICA Laboratory, Badji Mokhtar University, Sidi Amar, Annaba, Algeria

Nathaniel Christen

Linguistic Technology Systems, Fort Lee, NJ, United States

Noureddine Doghmane

Electronics Department, Faculty of Engineering Sciences, Laboratory of Automatic and Signals of Annaba (LASA), Badji Mokhtar University, Sidi Amar, Annaba, Algeria

Todor D. Ganchev

Department of Computer Science and Engineering, Technical University of Varna, Varna, Bulgaria

Krishna Gurugubelli

Speech Processing Laboratory, LTRC, KCIS, International Institute of Information Technology, Hyderabad, India

Saliha Harize

Electronics Department, Faculty of Engineering Sciences, Laboratory of Automatic and Signals of Annaba (LASA), Badji Mokhtar University, Sidi Amar, Annaba, Algeria

Mohamed El Haziti

High School of Technology, Mohammed V University in Rabat, Rabat, Morocco

Abdelillah Jilbab

ENSET, Mohammed V University in Rabat, Rabat, Morocco

Maksym Ketsmur

Department of Electronics Telecommunications and Informatics/IEETA,
University of Aveiro, Aveiro, Portugal

Arshid Iqbal Khan

PG Department of Electronics and IT, University of Kashmir, Srinagar, India

Nasreddine Kouadria

Electronics Department, Faculty of Engineering Sciences, Laboratory of
Automatic and Signals of Annaba (LASA), Badji Mokhtar University, Sidi Amar,
Annaba, Algeria

Djemil Messadeg

Electronics Department, Faculty of Engineering Sciences, Laboratory of
Automatic and Signals of Annaba (LASA), Badji Mokhtar University, Sidi Amar,
Annaba, Algeria

Amy Neustein

Founder and CEO, Linguistic Technology Systems, Fort Lee, NJ, United States

Shabir A. Parah

PG Department of Electronics and IT, University of Kashmir, Srinagar, India

Debadatta Pati

Department of Electronics and Communication Engineering, National Institute of
Technology Nagaland, Dimapur, India

Javaid A. Sheikh

PG Department of Electronics and IT, University of Kashmir, Srinagar, India

Samuel Silva

Department of Electronics Telecommunications and Informatics/IEETA,
University of Aveiro, Aveiro, Portugal

Madhusudan Singh

Department of Electronics and Communication Engineering, National Institute
of Technology Nagaland, Dimapur, India

António Teixeira

Department of Electronics Telecommunications and Informatics/IEETA,
University of Aveiro, Aveiro, Portugal

Ramakrishna Thirumuru

Speech Processing Laboratory, LTRC, KCIS, International Institute of Information
Technology, Hyderabad, India

Ravi Kumar Vuddagiri

Speech Processing Laboratory, LTRC, KCIS, International Institute of Information
Technology, Hyderabad, India

Anil Kumar Vuppala

Speech Processing Laboratory, LTRC, KCIS, International Institute of Information
Technology, Hyderabad, India

Foreword

Computer technology is increasingly ubiquitous, with most devices in our lives and businesses driven by digital processing. The processing power of devices connected to the Internet is increased beyond that of the device through cooperation with other devices and computing systems. The exponential growth of processing power in general is multiplied by the number of devices that can cooperate in using that power. Ubiquitous computing will expand “computer intelligence”—what these devices can do for us—at an ever-increasing rate.

Technologies such as speech recognition can tighten the connection between humans and this network, providing technologies such as digital assistants that can communicate with other devices and cloud computing systems to do our bidding. Networks of devices can also cooperate to create business and industrial systems with sensors and controllers that optimize operations, creating major economic benefits.

The software in a single digital device can be complex in itself. With ubiquitous computing, that complexity is magnified. Amy Neustein, editor of *Advances in Ubiquitous Computing*, has collected deep insights into some of the challenges created by ubiquitous computing and how leading researchers are attempting to understand and tame those challenges.

Ubiquitous computing isn’t just a challenging research area. It is a trend that will significantly impact our society and economy at an accelerating pace. The research presented in this fine collection is an attempt to make sure we control that computing power rather than it controlling us.

— William Meisel, Ph.D., author of *Computer Intelligence: With Us or Against Us?*

This page intentionally left blank

Introduction

Advances in Ubiquitous Computing: Cyber-Physical Systems, Smart Cities, and Ecological Monitoring debut some of the newest methods and approaches to multimodal user-interface design, safety compliance, formal code verification and deployment requirements, as they pertain to cyber-physical systems, smart homes/smart cities, and biodiversity monitoring. The contributors, who are distinguished members of universities and research centers in Europe, South Asia, Africa, and the United States, have been hand-selected by the editor of this volume because of their outstanding work in pervasive computing. In this anthology the authors assiduously examine a panoply of topics related to wireless sensor networks (WSNs). Those topics include interacting with smart-home appliances and biomedical devices, and designing multilingual speech recognition systems that are robust to vehicular, mechanical, and other noises common to large metropolises. The authors also recognize that in building smart cities there is a pressing need for voice biometrics through which one can validate a person's identity over a wireless medium. However, in order to build a robust speaker recognition system that functions in a noisy environment, they examine, in particular, new methods to control the emotion-state of the speaker which can easily impede speaker verification.

This volume recognizes that any discussion of pervasive computing in smart cities must not end there. We've witnessed in recent decades the perilous effects of climate change, which attests to the fact that our lives are not circumscribed by the geographically sculpted boundaries of cities, counties, countries, or continents. For this reason, in this book we address present and emerging technologies of scalable biodiversity monitoring: pest control, disease transmission, environmental monitoring, and habitat preservation. The common thread is the need to collect, store, process, and interpret vast amounts of data originating from sources spread over large areas and for prolonged periods of time. This requires immediate data storage and processing, reliable networking and solid communication infrastructure, along with intelligent data analysis and interpretation methods that can resolve contradictions and uncertainty in data. Undeniably, ubiquitous computing is one of the essential instruments in achieving these goals, combining advances in Computational Bioacoustics, Eco-acoustics, Remote Sensing, and related technology in support of the habitat preservation efforts.

As an illustration, contributors analyze how ubiquitous computing is used in wetlands which are home to impressive number of fauna and flora, some of which are threatened with extinction. They show, for example, how advanced WSN-based monitoring of birds in their natural habitat is able to identify and recognize different bird species instantly from the detection and processing of their vocalizations (call and/or song) using wireless sensor nodes. In fact, sensor networks are proving to be ideal platforms for recording and processing such data because of their characteristic compliance with the requirements of a project such as energy independence, the

average financial cost, wide geographic coverage, and the preservation of the environment. This is why a volume on advances in ubiquitous computing must consider the use of WSNs in those regions most vulnerable to the perils of biodiversity threats, as it is an undeniable truth that such threats have a cascading effect on the global world writ large.

The volume opens with a section entirely devoted to the exposition of some of the most fascinating new approaches and methods of ubiquitous computing, beginning with a survey of routing protocols for WSNs. The authors, in presenting a review of routing protocols that are developed in the field of the WSNs, examine the evolution of the relevant technologies for WSNs, such as component miniaturization, along with the concomitant evolution of microelectronics and the increase of computational capabilities, which have allowed important advances in the development of WSN technologies and applications in the first place. They show how routing protocols for WSNs are responsible for maintaining the routes in the network and how such WSNs ensure reliable communication under specific conditions. In short, they provide a survey of routing protocols for optimization energy consumption in WSNs.

This overview chapter of routing protocols is followed by a very interesting study on different excitation features for automatic speaker verification and anti-spoofing. For this purpose, the authors used three linear prediction (LP) residual-based implicit source features namely, residual mel frequency cepstral coefficient (RMFCC), LP residual Hilbert envelope mel frequency cepstral coefficient (LPRHEMFCC), and residual phase cepstral coefficient (RPCC) for replay detection. The authors proposed, in addition, two explicit excitation source features namely, pitch contour and glottal flow derivative cepstral coefficient (GFDCC). Performing experimental analysis using GMM-UBM ASV experiments (on IITG-MV replay database) and spoof detection experiments (on ASVspoof 2017 database), the authors showed that source features complement the vocal-tract information-based spoof-detection systems. As such, these study results illustrate the significance of joint use of source and system features for the development of generalized countermeasures to replay attacks.

The chapter that follows presents an exposition of Hypergraph-Based Type Theory and Requirements Engineering for Software Development in a Cyber-Physical Context. The chapter explores the integration of several methodologies related to source code analysis and software Requirements Engineering. In so doing, the author provides an extensive review of graph-based representations of source code, alongside applied type theory (for expressing programming languages' type systems) and systematic accounts of foundational programming elements such as functions/procedures, function calls, and inter-procedure information flows. The new representational device described by the author involves a theory of "channels" which permits graph-based code models to be integrated with type theories and lambda calculi structured around modern programming paradigms. The proposed techniques in this thought-provoking chapter support documentation and verification of procedural, data type, and holistic specifications; implementation-assumptions on procedures and/or modeling assumptions on types, along with larger-scale inter-type

relationships. The chapter employs real-world Cyber-Physical case studies to illustrate data models which call for advanced code-documentation techniques, insofar as Cyber-Physical software must prioritize safety and reliability. And, that need to prioritize safety and reliability can be most evident in the health-care field. In fact, one of the lacunae wisely pointed out by the author is that as software becomes an increasingly central feature of the biomedical ecosystem, there is yet no corresponding *oversight framework* that exists in the software world, which can be especially important for the safety and reliability of those biomedical devices.

That is, biomedical IT regulations tend to be ad hoc and narrowly domain-focused. The author of this chapter points to the fact that codebases in the United States which manage HL-7 data (the current federal Electronic Medical Record format) must meet certain requirements, but yet there is no comparable framework for software targeting other kinds of health-care information. So, for example, any data resulting from new medications, new treatments, new devices, new imaging techniques, etc., is not likely to be covered under the HL-7 standardization. The author illustrates this point by looking at the data generated from Whole Slide Imaging—microscope positioning and image-segment data—a new method of histological analysis invented by an NYU-incubated company. But since there is no way for such a data model of tissue analysis to be standardized in HL-7, this important data remains outside the electronic health record standardization. Similarly, the author points to Montefiore Medical Center Eran Bellin’s “Clinical Looking Glass” (CLG)—which represents novel methodologies for defining patient cohorts, where data from multiple databases can be pooled—meaning data sharing is part of the CLG methodology. Nonetheless, since there is no standardization of “consuming and querying” CLG data sets comparable to HL-7, this represents yet another illustration of important data that cannot be standardized because, as in this instance, CLG constitutes a new method which has not yet had time to be canonically defined within HL-7.

The author probes why this is happening, and suggests that this is not only—or not primarily—an issue of lax government oversight. The author opines that the deeper problem is that we do not have a clear picture, in the framework of computer programming and software development, of what a robust regulatory framework would look like: what kinds of questions it would ask; what steps a company could follow to demonstrate regulatory compliance; what indicators the public should consult to verify that any software that could affect their medical outcome has been properly vetted. And, outside the medical arena, similar observations could be made regarding software in Cyber-Physical settings like transportation, energy (power generation and electrical grids), physical infrastructure, environmental protections, government and civic data, and so forth—settings where software errors may cause personal and/or property damage. These are important questions which stimulate much needed discussion.

Finally, the section on new approaches and methods of ubiquitous computing is rounded out by an illuminating chapter on a new method of power-efficient speech transmission over 5G Networks using new signaling techniques. The authors

consider a new Radio Access Technology (RAT) after LTE (a 4G wireless communications standard) to meet the strict 5G requirements. They research new waveforms, new multiple access schemes, alternate channel coding schemes and, most importantly, a flexible frame structure. They amply demonstrate the merits of a new multicarrier modulation scheme (Ultra-Filter Bank Multicarrier UFMC) as a new candidate modulation for 5G communication systems.

In so doing, the authors show the recent advances in UFMC Modulation for 5G networks by comparing this modulation scheme with the conventional cyclic-prefix Orthogonal Frequency Division Multiplexing (CP-OFDM) approach. They point out the important issue of sub-channel equalization as demonstrated by the joining of UFMC with offset Quadrature Amplitude Modulation (OQAM). This chapter wisely places emphasis on the bit error rate (BER) analysis of a test recorded speech channel for Binary Phase Shift Keying (BPSK) Modulation, Quadrature Amplitude Modulation (QAM) at different diversity orders, using Alamouti-OSTBC and Maximum Ratio Combining (MRC) techniques. In addition, a new modulation technique known as Universal Filtered Multi-Carrier (UFMC) is implemented and the spectrum analysis of a speech signal is analyzed, using FBMC compared with OFDM. Moreover, in the proposed work a thorough comparison of various combining techniques for different modulations techniques is done, proving that UFMC modulation technique outperforms all other modulation schemes considered in this area of work.

In laying a foundation for the new methods and approaches to pervasive computing, the reader is then able to understand the true challenges in designing smart cities/smart homes/smart communities and in the architecture of a system for performing high-level ecological monitoring, many of which are elucidated in the remainder of the book. The contributors have labored through many years of data collection and analysis, to be able to present their research findings in one cohesive text. I am deeply honored as the editor of *Advances in Ubiquitous Computing* to have been personally invited by the editors of the Elsevier series, *Advances in Ubiquitous Sensing Applications for Healthcare*, to put together a volume for their illustrious collection. And I cherish the privilege of having worked closely with the authors who have contributed to this volume. Their work is novel and forward-looking.

The voices in this volume capture not only the technology of WSNs but the zeitgeist. In that spirit, the authors show the wondrous benefits of a cyber-physical world that complies with the highest level of safety and reliability, and how a meticulously constructed software development ecosystem is imperative in achieving those goals. On a personal note, I cannot end this preamble without acknowledging the sacrifices made by the contributors themselves.

Some were battling politically motivated internet connectivity privations and restrictions while trying desperately to ferry their work through publishing portals; some were battling grievous losses of family members and friends; and others were saddled with episodic health problems and financial hardship. One would never know the personal struggles of these contributors, for their work resonates a deep and unalterable commitment to the preservation and protection of both human and zoological populations. Years from now we will realize the prescience of these

scientists whose work fills the pages of this book, because they are the ones who cherished the environment well enough to probe deeper and deeper into new ways to use WSNs to improve our biosphere for all of nature's flora, fauna, and humankind. Thank you Elsevier and your team of production editors, headed by the very patient and caring John Leonard, for encouraging the publication of this important work!

Amy Neustein

This page intentionally left blank

PART

Wireless sensor networks and cyber-physical systems: New approaches and methods

1

This page intentionally left blank

Routing protocols for wireless sensor networks: A survey

1

Bahae Abidi^a, Abdelillah Jilbab^b, Mohamed El Haziti^c

^aLRIT—CNRST URAC No. 29, Rabat IT Center, Faculty of Sciences, Mohammed V University in Rabat, Rabat, Morocco ^bENSET, Mohammed V University in Rabat, Rabat, Morocco ^cHigh School of Technology, Mohammed V University in Rabat, Rabat, Morocco

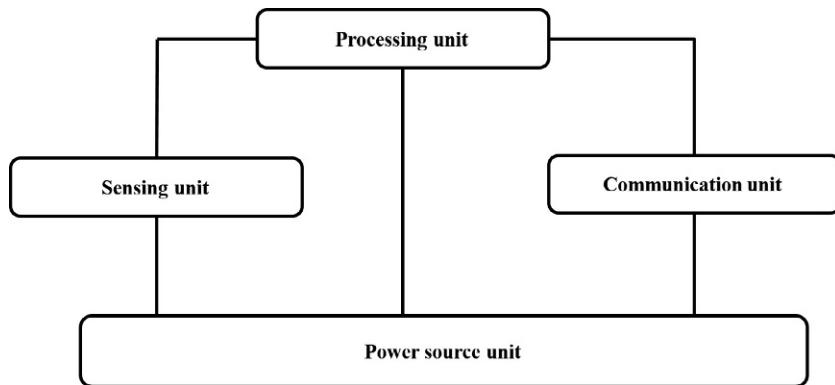
1.1 Introduction

In recent years, communication technology has been an exponential progress from wired to wireless communication. Recent advances in wireless communication, embedded computing, and digital electronics give rise to small autonomous components that are able to provide physical and environmental measurements. These nodes allowed the creation of a new category of networks known as wireless sensor networks (WSNs) [1]. WSNs are a group of specialized sensors with a communications infrastructure to monitor physical or environmental conditions. A WSN is a self-organized network composed of a large number of micro-sensors; the deployment of the sensor node can be random, regular or mobile to produce high-quality information through a wireless communication [2]. The sensor nodes are small components that are mainly intended to obtain physical measurements [3]. The nature of the collected information strongly depends on the application in which the nodes are used. They are designed to perform some processing and cooperatively pass their data with the use of the wireless medium to a more powerful node called the base station (BS). This provides higher capacities for storing and processing than the other normal nodes. Multiple features allow the nodes to operate in an autonomous way without human intervention for long periods, for example:

- *Capacity to be auto-organized*: Each node is able to detect its neighbors and relay data to the BS without any prior topology.
- *Fault tolerance*: If a node fails, the other nodes can continue to do their tasks without affecting the operation of the network.

The four parts [4] that form the sensor node are shown in Fig. 1.1.

- *Sensing unit*: This unit is responsible for collecting measurements and then converts the analog signal to a digital value. The measurements collected by this unit will then be processed by the processing unit.

**FIG. 1.1**

Architecture of a wireless sensor node.

- *Processing unit*: Characterized by the ease of programming and low power consumption, composed of a microprocessor, it performs process data and controls the operation of the other units of the sensor node. The amount of the processed data by this unit depends on the size of the captured data by the node and that received from its neighbors. To reduce the volume of the transmitted data by the sensor node, new aggregation algorithms have been designed.
- *Communication unit*: In terms of energy, this unit is the highest consumption component. It is composed of a radio transceiver that is responsible for transmission and reception of data, which provides four modes: transmission, reception, idle, and sleep. Putting the transceiver in sleep mode reduces the energy consumption and the required energy to communicate data depends on the distance between the receiver and sender, as well as the number of intermediate nodes that relay the message.
- *Power supply unit*: This is an irreplaceable battery with limited capacity and it is responsible for providing power to all the other units.

Sensor node can optionally integrate two other units: the geo-location unit and the energy harvesting unit. The first one is responsible for providing the position of sensors because in some cases we need to know the position in order to use the information coming from the network more effectively. The main role of the second optional unit is increasing node lifetime. It is equipped with a different technical way to recharge the battery through environmental energy sources by converting the different energy such as solar energy, electromagnetic waves, and vibration to electric power.

The small sizes of sensors, their availability, and the decrease in cost of micro-sensors have expanded the domains of application of sensor networks. They include different domains [5].

The chapter is organized as follows: [Section 1.2](#) introduces the different application of WSN, followed by the characteristics and constraints of WSN in [Section 1.3](#). [Section 1.4](#) discusses the network topology, then [Section 1.5](#) presents routing protocol for WSNs. Finally, we summarize with a conclusion in [Section 1.6](#).

1.2 Application of WSNs

The applications of WSN can be classified according to their objectives [6] or the topology used in the network [7]. It can have a deterministic or random deployment depending on the application. These WSNs can be deployed in hostile places where human intervention is very risky and sometimes impossible. The low cost of the node and the plasticity to be deployed in the different area enable the WSN application fields to be very diverse. The sensor networks consist of different types of sensor which are able to monitor a wide variety of conditions such as temperature, humidity, vehicular movement, pressure, noise levels, and the absence or presence of some objects. The concept of micro-sensing and wireless connection of these nodes promise many new applications areas, which can be categorized into military, environmental [8], health [9], and home applications [10].

The rapid deployment, self-organization, and fault tolerance characteristics of sensor networks make them a very promising sensing technique for the military. They can be an integral part of military command, control, communications, computing, and surveillance systems. The destruction of some nodes by hostile actions does not affect military operations as much as the destruction of a traditional sensor, which makes the sensor networks concept a better approach for battlefields. Some military applications [11] are battlefield surveillance, reconnaissance of opposing forces and terrain, battle damage assessment, attack detection, and reconnaissance. The environmental applications of sensor networks [12] include tracking the movements of small animals, birds, and insects plus monitoring the environmental conditions that affect the crops, the biological earth, environmental monitoring in marine, the detection of forest fires, the mapping of the environment, and pollution study.

The health applications for sensor networks are providing interfaces for the disabled, integrated patient monitoring, and the telemonitoring of human physiological data by collecting physiological data to be used for medical exploration. These small sensor nodes can also detect elderly people's behavior and allow doctors to identify predefined symptoms earlier. The health monitoring applications can facilitate the quality of life for the patients [13]. The doctors may carry a sensor node which allows other doctors to locate them within the hospital. Patients may have small sensor nodes attached to them which have specific tasks like detecting heart rate or detecting blood pressure.

As technology advances, smart sensor nodes can be used for home applications. These sensor nodes inside domestic devices like microwaves and refrigerators can

interact with each other and with the external network via the internet or satellite. They allow users to manage home devices locally and remotely more easily.

1.3 Characteristics and constraints of WSNs

The realization and conception of the WSN are influenced by many parameters [14]. These factors serve as guidelines for the development of the algorithms and protocols used in WSNs. The main factors and constraints can be summarized as follows:

- *Limited lifetime:* The sensor node has limited energetic resources. It is usually powered by a thin battery and it is often very difficult or even impossible to charge or recharge it when the WSN is deployed in inaccessible areas, so the sensor's lifetime depends on that of their battery. The scarcity of energetic resources may result in a short lifetime, so energy management in this kind of network is crucial. The limited lifetime is the most important constraint that WSN communication protocols should deal with it.
- *Auto-configuration and auto-organization:* Sensors are usually deployed in the monitored area, either randomly or manually. Once deployed, these sensors have the ability to be auto-configured. In addition, they have the ability to organize themselves autonomously into a communication network and reconfigure their connectivity in the event of topology changes and node failures, in order to collaborate with each other and accomplish their different tasks.
- *Low capacities of processing and storing:* This limitation is due to two constraints: low cost and low energy consumption. Usually, applications need to scatter thousands of sensors, so the fabrication cost of these components should remain in a reasonable range; therefore, reducing the cost of sensor nodes is important and will result in the cost reduction of the whole network. Second, these components should respect the limited energetic constraint, so they are fabricated using low power consumption-based electronic circuits.
- *Scalability:* Usually a high number of nodes are scattered in the monitored area. A WSN can contain up to 1000 of sensor nodes due to their ability to collaborate with each other and accomplish the application requirement.
- *Reliability:* Network protocols designed for sensor networks must provide error control and correction mechanisms to ensure reliable data delivery over noisy, error-prone, and time-varying wireless channels.
- *Fault tolerance:* Along the network life cycle, some sensors can get dysfunctional, due to a lack of energy or material fault, etc. These failures should not affect the network operation. In addition, in case of adding new nodes, they should be easily integrated into the network. Therefore, a WSN should have the capacity to continue operating without any interruption of its tasks.

1.4 Network topology of WSNs

The easy deployment and developments of WSNs have taken the network topology in new different orientations.

1.4.1 Bus topology

Alternatively referred to as a line topology, this is the simplest of network topology. In this type of topology [15], all the nodes are connected to the single cable, called a bus. It is the easiest network topology for connecting nodes in a linear fashion. The bus network works best with a limited number of nodes. If more than a few nodes are added to a network, performance problems will likely result and it can be difficult to identify the problems.

1.4.2 Tree topology

This is also called a cascaded star topology. It is a special type of structure in which many connecting elements are arranged like the branches of a tree [16]. The network uses a central hub called a root node as the main communication router. In this topology, there can be only one connection between any two connected nodes, because any two nodes can have only one mutual connection. The main advantage of the tree topology is that the expansion of a network can be easily possible, and also error detection becomes easy. The disadvantage with this network is that it relies heavily on the bus cable; if it breaks, all the network will collapse.

1.4.3 Star topology

A star topology is a communication topology, where each node connects directly to a gateway. A single gateway can send or receive a message to a number of remote nodes. In a star topology, the nodes are not permitted to send messages to each other. This allows low-latency communications between the remote node and the gateway (BS). Due to its dependency on a single node to manage the network, the gateway must be within the radio transmission range of all the individual nodes. The advantage includes the ability to keep the remote node's power consumption to a minimum and simply under control. The size of the network depends on the number of connections made to the hub.

1.4.4 Mesh topology

Mesh topologies allow transmission of data from one node to another, which is within its radio transmission range. If a node wants to send a message to another node, which is out of radio communication range, it needs an intermediate node to forward the message to the desired node [17]. The advantages of the mesh

topology include easy isolation and detection of faults in the network. The disadvantages are that the network is large and requires a huge investment.

1.5 Routing protocol for WSNs

1.5.1 Hierarchical routing protocol

The special feature of the hierarchical routing protocols is that it provides self-organization capabilities to allow large-scale network deployment. Basically, in a hierarchical architecture, some nodes take responsibility for performing high-energy transmission while the rest perform the normal task. The hierarchical routing protocol can be categorized into two types based on the topology management:

- *Cluster-based hierarchical routing protocol*: Sensor nodes are grouped into clusters and each of these clusters is led by a cluster head (CH), which is one of the nodes. This node acts as an intermediate node between cluster members and the BS.
- *Chain-based hierarchical routing protocol*: All the nodes in the field are connected in a chain structure. The node with a maximum energy is chosen as the chain leader to mediate the data transmission from normal nodes and the BS.

The main goal of the hierarchical routing protocol [18] is to maintain efficiently the energy consumption of sensor nodes by performing data aggregation and fusion in order to decrease the number of transmitted data to the BS.

1.5.1.1 Low energy adaptive clustering hierarchy

Low energy adaptive clustering hierarchy (LEACH) is the most popular energy efficient cluster-based hierarchical routing protocol [19] for WSNs that was proposed for reducing energy consumption. It uses localized coordination to enable scalability and robustness for dynamic networks, and to reduce the amount of information that must be transmitted to the base station, it incorporates data fusion into the routing protocol.

Consuming lower energy is the main goal of the LEACH protocol to create and maintain clusters in order to improve the lifetime of a WSN. In the LEACH protocol, most nodes transmit to cluster heads, and the cluster head aggregate and compress the data and forward them to the BS, and each node uses at each round a specific probability to determine whether it will become a cluster head in this round. Randomized rotation of the CH and corresponding clusters are the key features of LEACH.

The main advantages of the LEACH protocol are as follows:

- Scalability: by limiting most of the communication inside the different clusters of the network.
- Single hop routing: from the sensor node to the cluster head.
- Reduces the amount of traffic within the network: the cluster head aggregates or fuses the information that has been collected by the sensor nodes.
- Increases the lifetime of the network in three phases.

- Does not require the information of localization of the sensor nodes in the network to create the clusters.

However, LEACH also has some disadvantages:

- LEACH does not work well with the applications that require large area coverage.
- Requires high range of transmission power in the network because there is no intercluster communication in the network to communicate.
- Leach cluster head is not uniformly distributed within the cluster.

1.5.1.2 Power efficient gathering in sensor information systems

Power efficient gathering in sensor information systems (PEGASIS) is a chain-based hierarchical protocol [20], developed version of the LEACH protocol, instead of grouping sensor nodes into clusters. PEGASIS applies chain forming at the beginning of each round. Upon deployment, each node is assumed to know its neighbors. The algorithm is used to select a start node, which will initiate the chain-forming process. This node is normally the farthest node from the BS. The start node will then select the closest neighbor to which it will set a connection path. The second node that joins the chain then will select its closest neighbor and set a new connection path. The process goes on until all nodes have joined the chain.

The main advantages of PEGASIS are as follows:

- Collaborative technique: to increase the lifetime of each sensor.
- Avoids the cluster formation: uses only one node in a chain to transmit to the BS instead of multiple nodes.
- Reduces the power required to transmit data per round as the power draining is spread uniformly over all the sensor nodes.

The disadvantages of PEGASIS are as follows:

- Assumes that each node is able to communicate with the BS directly and in major cases, sensor nodes use multihop communication to reach the BS.
- Assumes that all nodes have the same level of energy and are likely to die at the same time.
- Introduces excessive delay for distant sensor node on the chain.

1.5.1.3 Threshold sensitive energy efficient sensor network protocol

Threshold sensitive energy efficient sensor network (TEEN) protocol is designed to be responsive of the sudden changes in the sensed attributes [21]. The architecture of sensor network in TEEN is based on hierarchical grouping where closer nodes form clusters, and this process goes on the second level until the BS is reached. The sensor nodes sense the medium continuously but the data transmission is done less frequently. TEEN uses LEACH's strategy to form a cluster. The whole network consists of two levels: the first level cluster heads are formed away from the BS and the second level cluster heads are formed near to the BS [22].

The main advantages of TEEN are as follows:

- The threshold can be varied: This depends on the criticality of the sensed attribute and the application.
- The data reach the user almost instantaneously.
- A smaller value of the threshold can provide more accurate exploration or information of the network.

The disadvantages of TEEN are as follows:

- Cluster head always wait for data from the nodes by maintaining the transmitters on.
- A sensor node may wait for a time slot for data transmission and the time slot may be wasted if a sensor node has no data for transmission.

1.5.1.4 Hybrid energy efficient distributed clustering

Hybrid energy efficient distributed clustering (HEED) periodically selects cluster heads according to two parameters, a residual energy of each node and the proximity to its neighbors, also called node degree. In HEED [23], the proposed algorithm periodically selects a cluster head according to a combination of the two parameters. The clustering process at each node requires several rounds and each round is long enough to receive data from any neighbor within the cluster range. This protocol is most suitable for prolonging the network lifetime.

The main advantages of HEED are as follows:

- Distribution of energy extends the lifetime of the sensor nodes within the network and stabilizing the neighboring sensor nodes.
- Even when nodes are not synchronized, it operates correctly.
- Each sensor node requires only local information to form the cluster.

The disadvantages of HEED are as follows:

- The periodic cluster head rotation needs extra energy to reconstruct the clusters.
- The random selection of the cluster heads may cause higher communication.

1.5.2 Data-centric routing protocol

In many sensor network applications, with a large number of nodes, it is not possible to assign global identifiers to each node. This absence of global identifiers with the random deployment of sensor nodes makes it difficult to select a set of sensor nodes to interrogate. Therefore, the data are typically transferred to a deployment region with significant redundancy. All this led to the data-centric routing protocol [24].

1.5.2.1 Sensor protocol for information via negotiation

Sensor protocol for information via negotiation (SPIN) is the first data-centric protocol designed for WSNs [25]. SPIN uses three types of messages: ADV, REQ, and DATA. The ADV messages are broadcasted by a node that has some data; this

message contains information about the type of data contained by the advertising node. Interested nodes that got the ADV message send a REQ message requesting data. The node having the data sends the data to the interested nodes. The nodes, after receiving data, send an ADV message and the process continues until the data reach the BS. In this protocol, every node will have a resource manager, which will inform each node about the amount of various resources left in the node. The node can make a decision regarding whether it can be as forwarding node or not. The main advantages of SPIN are as follows:

- Reduction of redundant data: Because nodes negotiate before the data transmitting.
- Minimizing the network overhead and the energy consumption.

The disadvantages of SPIN protocol are as follows:

- The data advertisement cannot guarantee delivery of data, thus it does not fit in applications requiring the reliability of data.
- Not good for high-density distribution of nodes.

1.5.2.2 Directed diffusion

This is an important protocol in the data-centric protocol for WSNs [26]. It consists of several elements: interests, data messages, gradients, and reinforcements. The interest message is a query which specifies what a user wants. Each interest contains a description of a sensing task that is supported by a sensor network for acquiring data. The data in sensor networks are the collected information of a physical phenomenon. It can be an event, which is a short description of the sensed phenomenon. The data are named using attribute-value pairs in directed diffusion. A sensing task is disseminated throughout the sensor network as an interest of named data. This dissemination sets up gradients within the network designed to draw events. Specifically, a gradient is a direction state created in each node that receives an interest. The gradient direction is set toward the neighboring node from which the interest is received. Events start flowing toward the originators of interests along multiple gradient paths. The sensor network reinforces one or a small number of these paths.

Directed diffusion supports the multirouting but it is not suitable for applications that require continuous data delivery.

1.5.3 Location-based protocol

The location-based protocol or geographic protocol uses location information to formulate an efficient route search toward the destination [27]. It is a very suitable protocol for sensor networks, where data aggregation is a useful technique to minimize the number of transmissions toward the BS by eliminating redundancy among packets from the different sources. It is very attractive for large multihop wireless networks in which the nodes are not reliable and their network topology is frequently changing. The location-based protocol requires only the propagation of single hop

topology information, like the best neighbor to make correct forwarding decisions. There are various types of geographic routing protocols in WSNs. Each has a different feature.

1.5.3.1 Geographic adaptive fidelity

Geographic adaptive fidelity (GAF) is a location-aware routing protocol that divides the network into grids and makes one node active at a time from each grid to sense data [28]. Proper grid size is very important in this protocol, because it directly affects the network connectivity and also energy consumption. A smaller grid consumes more energy than a larger grid. The size of grid depends on the transmission range of nodes; a node can be able to communicate with other active nodes in the adjacent grids. GAF uses the location information of the nodes to send data. It works in three states:

- Discovery: search the next active node.
- Active: participates in communication.
- Sleep: saves the nodes' energy by turning radio off.

At the time of discovery, all the nodes are awake and transmit discovery messages to select the higher energy node to become active. This process will take T_d time. After time T_d , the selected node enters into active state and the rest go to sleep state. Time is defined for each state in advance. The node will remain in active state for T_a time and in sleep state for T_s time. The T_a time must be more than T_s time to avoid the loss of communication.

1.5.3.2 Geographical and energy-aware routing

Geographical and energy-aware routing (GEAR) is an energy-aware protocol [29]. In this protocol, interest is sent to a certain region which means packets are routed to a particular node based on a destination node ID in the packet. This protocol keeps two types of cost, estimated cost and learning cost, and there are two phases in GEAR. The first phase entails a geographical and energy-aware neighbor selection algorithm being used to forward packets toward the target area. In the second phase, a recursive geographic forwarding algorithm is used to disseminate packets to nodes within the target area.

1.5.3.3 Minimum energy communication network

Minimum energy communication network (MECN) establishes and maintains a minimum energy network for wireless networks by utilizing low power GPS. This protocol [30] has two phases. The first phase takes the positions of the two-dimensional plane and constructs a thin graph, which consists of all the enclosures of each transmit node in the graph. The enclosed graph contains finest links in terms of energy consumption. The second phase finds the best possible links on the enclosure graph. The MECN protocol utilizes the distributed shortest path algorithm with power consumption as a cost metric.

Table 1.1 Comparison of routing protocols in WSNs.

	Classification	Mobility	Negotiability	Aggregation	Multiway	Power usage
LEACH	Hierarchical	Fixed	No	Yes	No	Maximum
PEGASIS			No	No	No	
TEEN			No	Yes	No	
HEED			No	Yes	No	
SPIN	Centric	Possible	Yes	Yes	Yes	Limited
Directed diffusion			Yes	Yes	Yes	
GEAR	Geocast	Limited	No	No	No	
GAF			No	No	No	
MECN		No	No	Yes	No	Maximum

The advantages of geographic routing protocol are as follows:

- Reduces the need for maintaining routing tables.
- Reduces the control overhead.
- Does not require flooding.
- The mobility support can be facilitated.

There are several routing protocols in the WSN. Each protocol has its own features which make them different from others; they try to reduce the overhead and make the network faster. On the basis of above explanation [Section 1.5](#), the [Table 1.1](#) shows the basic comparison of different routing protocols on the basis of many factors, that is, classification, mobility, negotiation, aggregation, multiway, and power usage.

1.6 Conclusion

Research into WSNs is very dynamic. Concerning the applications about these files, they have high expectations regarding applications. Each application differs in features and requirements to support the diversity of these domains. Development of new algorithms, communications protocols, and services is needed to improve the quality of research. This chapter presents a survey about the routing protocol for the optimization of energy consumption to maximize the network lifetime in a WSN. We have firstly presented the different application of the WSNs, then the characteristics and constraints that they face, then the network topology of these WSNs, and finally, we have summarized and compared some different proposed routing protocols.

References

- [1] Q. Wang, I. Balasingham, Wireless sensor networks—an introduction, in: *Wireless Sensor Networks: Application—Centric Design*, InTech, 2010, pp. 1–14.

- [2] B. Abidi, A. Jilbab, M. El Haziti, Optimization of energy consumption with the gateway nodes in wireless sensor networks, *Int. J. Sensors Wirel. Commun. Control* 7 (2) (2017) 152–160.
- [3] G.J. Pottie, W.J. Kaiser, Wireless integrated network sensors, *Commun. ACM* 43 (5) (2000) 51–58.
- [4] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: a survey, *Comput. Netw.* 38 (4) (2002) 393–422.
- [5] K. Romer, F. Mattern, The design space of wireless sensor networks, *IEEE Wirel. Commun.* 11 (6) (2004) 54–61.
- [6] J. Yick, B. Mukherjee, D. Ghosal, Wireless sensor network survey, *Comput. Netw.* 52 (12) (2008) 2292–2330.
- [7] M.P. Jurisic, Z. Tafa, G. Dimic, V. Milutinovic, A survey of military applications of wireless sensor networks, in: 2012 Mediterranean Conference on Embedded Computing (MECO)IEEE, 2012, pp. 196–199.
- [8] P. Cheong, K.-F. Chang, Y.-H. Lai, S.-K. Ho, I.-K. Sou, K.-W. Tam, A ZigBee-based wireless sensor network node for ultraviolet detection of flame, *IEEE Trans. Ind. Electron.* 58 (11) (2011) 5271–5277.
- [9] J. Halamka, Early experiences with positive patient identification, *J. Healthc. Inf. Manag.* 20 (1) (2006) 25–27.
- [10] Y.Q. Ni, Y. Xia, W.Y. Liao, J.M. Ko, Technology innovation in developing the structural health monitoring system for Guangzhou new TV tower, *Struct. Control Health Monit.* 16 (1) (2009) 73–98.
- [11] Th. Arampatzis, J. Lygeros, S. Manesis, A survey of applications of wireless sensors and wireless sensor networks, in: Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and AutomationIEEE, 2005, pp. 719–724.
- [12] K. Sohraby, D. Minoli, T. Znati, *Wireless Sensor Networks: Technology, Protocols, and Applications*, John Wiley & Sons, 2007.
- [13] B. Abidi, A. Jilbab, M. El Haziti, A new generation of wireless sensors networks: wireless body area networks, in: World Conference on Information Systems and TechnologiesSpringer, 2017, pp. 384–393.
- [14] J. Zheng, A. Jamalipour, *Wireless Sensor Networks: A Networking Perspective*, John Wiley & Sons, 2009.
- [15] F.L. Lewis, Wireless sensor networks, *Smart Environ. Technol. Protocols Appl.* 11 (2004) 46.
- [16] N. Xu, S. Rangwala, K.K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, D. Estrin, A wireless sensor network for structural monitoring, in: Proceedings of the Second International Conference on Embedded Networked Sensor SystemsACM, 2004, pp. 13–24.
- [17] C. Buratti, A. Conti, D. Dardari, R. Verdome, An overview on wireless sensor networks technology and evolution, *Sensors* 9 (9) (2009) 6869–6896.
- [18] S. Sharma, S.K. Jena, A survey on secure hierarchical routing protocols in wireless sensor networks, in: Proceedings of the 2011 International Conference on Communication, Computing and SecurityACM, 2011, pp. 146–151.
- [19] M. Aslam, N. Javaid, A. Rahim, U. Nazir, A. Bibi, Z.A. Khan, Survey of extended LEACH-based clustering routing protocols for wireless sensor networks, in: 2012 IEEE 14th International Conference on High Performance Computing and Communication,

- and 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS)IEEE, 2012, pp. 1232–1238.
- [20] F. Sen, Q. Bing, T. Liangrui, An improved energy-efficient PEGASIS-based protocol in wireless sensor networks, in: 2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), vol. 4 IEEE, 2011, pp. 2230–2233.
 - [21] N.A. Pantazis, S.A. Nikolidakis, D.D. Vergados, Energy-efficient routing protocols in wireless sensor networks: a survey, *IEEE Commun. Surv. Tutorials* 15 (2) (2013) 551–591.
 - [22] S. Waware, N. Sarwade, P. Gangurde, A review of power efficient hierarchical routing protocols in wireless sensor networks, *Int. J. Eng. Res. Appl.* 2 (2) (2012) 1096–1102.
 - [23] Y. Zhou, X. Wang, T. Wang, B. Liu, W. Sun, Fault-tolerant multi-path routing protocol for WSN based on HEED, *Int. J. Sensor Netw.* 20 (1) (2016) 37–45.
 - [24] S.D. Muruganathan, D.C.F. Ma, R.I. Bhasin, A.O. Fapojuwo, A centralized energy-efficient routing protocol for wireless sensor networks, *IEEE Commun. Mag.* 43 (3) (2005) S8–S13.
 - [25] Z. Rehena, S. Roy, N. Mukherjee, A modified SPIN for wireless sensor networks, in: 2011 Third International Conference on Communication Systems and Networks (COMSNETS)IEEE, 2011, pp. 1–4.
 - [26] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, F. Silva, Directed diffusion for wireless sensor networking, *IEEE/ACM Trans. Netw.* 11 (1) (2003) 2–16.
 - [27] L. Blazevic, J.-Y. Le Boudec, S. Giordano, A location-based routing method for mobile ad hoc networks, *IEEE Trans. Mob. Comput.* 4 (2) (2005) 97–110.
 - [28] S.K. Singh, M.P. Singh, D.K. Singh, Routing protocols in wireless sensor networks—a survey, *Int. J. Comput. Sci. Eng. Surv.* 1 (63–83) (2010) 29–31.
 - [29] Y. Yu, R. Govindan, D. Estrin, Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks, (2001).
 - [30] D. Goyal, M.R. Tripathy, Routing protocols in wireless sensor networks: a survey, in: 2012 Second International Conference on Advanced Computing and Communication Technologies (ACCT)IEEE, 2012, pp. 474–480.

This page intentionally left blank

Replay attack detection using excitation source and system features

2

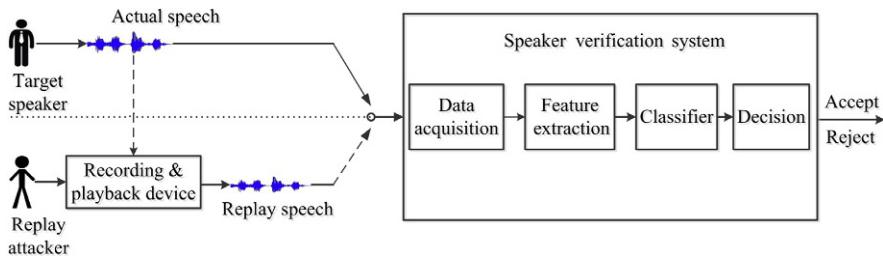
Madhusudan Singh, Debadatta Pati

Department of Electronics and Communication Engineering, National Institute of Technology Nagaland, Dimapur, India

2.1 Introduction

Automatic speaker verification (ASV) systems accept/reject claimed identities on the basis of provided speech samples [1, 2]. Due to significant advancement in speaker verification (SV) technology, the modern ASV systems have achieved verification accuracy close to perfection. In fact, the modern ASV systems are vulnerable to spoof attacks [3]. Consequently, speech research communities are actively participating in developing efficient spoof detection algorithms for ASV antispoofering over the last 5 years, for example, ASVspoof 2015 [4] and ASVspoof 2017 [5] challenges. A spoof attack is an attempt to alter ASV system decisions by providing artificial speech samples of any target speaker [6]. Generally four types of spoof attack methods are addressed in the literature, named as impersonation, replay, speech synthesis (SS), and voice conversion (VC). *Impersonation* is a technique where an impostor sounds like a target speaker through voice mimicry [7, 8]. *Replay attack* involves presentation of prerecorded target speaker's speech samples through a playback device in front of the ASV system [9, 10]. SS generates artificial target speech for a given input text [11, 12]. VC converts the impostor's voice characteristics to that of the target speaker [13, 14].

Impersonation is marked as a low-risk attack, as its effectiveness naturally depends on the skill of the impersonator and similarity of the impersonator's voice to that of the target speaker [3]. On the other hand, replay, SS, and VC attacks are equally highly effective. SS and VC attacks are high technology attacks. Replay attack is a simple low technology attack. It can be implemented using only a high-quality recording and playback device, and hence may pose a greater risk to ASV system [3]. In reply, the research community has paid serious attention to the development of countermeasures to replay attacks, like SS and VC attacks. Fig. 2.1 shows a common ASV system with replay spoofing scenario. In a genuine attempt, a speech sample is directly presented to the ASV system, and is called *actual speech*. In the case of a replay attempt, the speech sample is passed through the attacker's recording and playback devices before being presented to the ASV system, and is known as *replay speech*. As a consequence, replay speech gets affected by the

**FIG. 2.1**

Example ASV system demonstrating the difference between a genuine and replay attempt.

attacker's recording and playback device characteristics. Thus, detection of replay attacks relies mainly on tracing the recording and playback device artifacts present in the replay signals.

Various approaches have been proposed for replay detection in the literature. Most of them have utilized acoustic level differences between actual and replay signals caused by the playback devices, the recording devices, and the environment acoustic conditions of the surroundings. In [15], increased spectral flatness of the replay signal due to inclusion of noise and reverberations from the surroundings was used to detect replay attacks using spectral ratio and modulation index-based features. In [16], the differences between the channel pattern noise computed from original and replay recordings were used for detecting replay signals. A study [17] explored the distortion that occurs in the high-frequency regions due to the additional antialiasing filtering process during rerecording via microphone for replay detection. In [18], a multiclass deep neural network (DNN) was trained to discriminate between different available channel conditions under a replay attacks scenario. It was shown that multiclass DNN back-end gives superior performance over two-class GMM back-end. The work in [19] captures the changes in energy levels at low signal-to-noise ratio (SNR) time instant for detecting replay signals using single frequency filtering cepstral coefficients (SFFCC) approach. In [20], source features, namely epoch feature (EF) and peak-to-side lobe ratio mean and skewness (PSRMS), were proposed to capture the variations in the source information for detecting replay attacks. Moreover, a comprehensive study on a set of different conventional and non-conventional features for the development of replay detection systems was reported in [21]. In the ASVspoof 2017 challenge, the best-reported replay detection system [22] (equal error rate, EER = 6.73%, S01) is based on convolutional neural networks (CNN) and recurrent neural networks (RNN) learning approaches. Moreover, a few more recent works have been proposed in Interspeech 2018 related to replay attacks detection. Jelil et al. [23] have proposed the compressed integrated linear prediction residual (CILPR) feature for discriminating between genuine and replayed signals. The CILPR feature models the changes in the temporal dynamics of the integrated LP residual signal between two glottal closure instants (GCIs). In [24], two features,

namely mel-scale relative phase (Mel-RP) and phase-based source-filter vocal tract (PBSFVT), are combined at score level for replay detection. It has been shown that the phase-based features outperform conventional magnitude-based features in a replay detection context. In [25], the empirical mode decomposition cepstral coefficient (EMDCC) and linear frequency modified group delay cepstral coefficient (LFMGDCC) have been proposed for detecting replay attacks. In [26], high-frequency regions have been exploited using amplitude and frequency modulation (AM and FM)-based temporal cepstral features for detecting replay attacks. The features are derived from the Convolutional Restricted Boltzmann Machine (ConvRBM) using an auditory filterbank learning approach. In [27], energy separation algorithm (ESA)-based features, namely instantaneous amplitude (IA) and instantaneous frequency (IF) cosine coefficients (i.e., ESA-IACC and ESA-IFCC), have been proposed to distinguish between actual and replay signals. The features are extracted using linearly spaced Gabor filtersbank to capture useful informative evidence for spoof detection from the entire spectrum. Suthokumar et al. [28] have explored increments in the noise and reverberations from replay transmission channels for detecting replay attacks using two novel features, namely modulation centroid frequency cepstral coefficient (MCFCC) and modulation static energy cepstral coefficient (MSECC). **Table 2.1** provides a brief summary of recent prior works related to replay attack detection proposed in *Interspeech 2017–18*.

This work is the extension of our recent prior work [30], published in *Interspeech 2018*. In that work, two source features, namely residual mel-frequency cepstral coefficient (RMFCC) and a novel feature LP residual Hilbert envelope mel-frequency cepstral coefficient (LPRHEMFCC), were proposed for a replay detection task. The RMFCC feature is obtained by short-term processing of the LP residual and provides compact modeling of excitation source information using mel-cepstral analysis. The LPRHEMFCC feature is obtained by short-term precessing of Hilbert envelope of the LP residual, and provides compact representation of the source

Table 2.1 A brief summary of recent prior works related to replay attack detection proposed in *Interspeech 2017–18*.

Study	Features	Classifier	EER (%)
Lavrentyeva et al. [22]	FFT features	CNN, RNN	6.73 (S01)
Ji et al. [29]	MFCC, CQCC	GMM classifiers	12.34 (S02)
Nagarsheth et al. [18]	HFCC, CQCC	DNN-SVM	11.50
Jelil et al. [20]	EF, PSRMS, CQCC	GMM	17.61
Jelil et al. [23]	CILPR, CQCC	GMM	9.41
Sailor et al. [26]	AM and FM features	GMM	8.89
Kamble et al. [27]	ESA-IACC, ESA-IFCC	GMM	9.64
Suthokumar et al. [28]	MCFCC, MSECC	GMM	6.54

information present specially around GCIs using mel-cepstral analysis. As an extension, in this work another source feature called residual phase cepstral coefficient (RPCC) has been proposed along with LPRHEMFCC for detecting replay attacks. The RPCC feature is obtained by short-term processing of the residual phase, and provides compact representation of perceptually meaningful excitation source-like information using mel-cepstral analysis. From the signal perspectives, the Hilbert envelope of the LP residual mainly contains the amplitude information of LP residual samples. However, information about the sequence of the LP residual samples is lost in the Hilbert envelope of the LP residual. It is the residual phase that represents the excitation source information present in the sequence of the LP residual samples [31]. Thus, the Hilbert envelope and residual phase can be considered as two components of the LP residual signal. The Hilbert envelope is the magnitude function of the analytical signal derived using LP residual and Hilbert transform [31]. The residual phase is the cosine of the phase function of the analytic signal. Even though the analytical signal is common, LPRHEMFCC and RPCC features are derived from the magnitude and phase function, respectively, of the analytical signal using mel-cepstral analysis. Accordingly, the source information representations of LPRHEMFCC and RPCC features may be different. The RPCC feature explored in this work has been previously used for an SV task [32]. The novelty of this work lies in exploring RPCC and LPRHEMFCC features together for detecting replay attacks. The LPRHEMFCC and RPCC together with CQCC provide 8.86% EER, and hence approaches to state-of-the-art (see Table 2.1). Further, intuitively, it can be predicted that the combination of individual source representations from the Hilbert envelope and residual phase would be comparable to stand-alone source representation from the raw LP residual signal. In this direction, a combination of the Hilbert envelope (LPRHEMFCC) and residual phase (RPCC) is compared with the raw LP residual (RMFCC) signal. With this motivation, a comparative analysis has been performed to investigate the usefulness of processing the Hilbert envelope and residual phase (together) over direct processing of the raw LP residual signal. This is achieved through Gaussian mixtures model-universal background model (GMM-UBM) ASV experiments and spoof detection experiments using a self-developed IITG-MV replay database and standard ASVspoof 2017 database, respectively. Further to, contrary to current trends on development of stand-alone countermeasures, the proposed work aims to reject replay trials directly on the ASV system. The genuine trials and either zero-effort impostor trials or replay trials are classified using an EER decision threshold of an ASV system. The significant point of interest here is that the proposed approach (in an IITG-MV database context) does not require any separate countermeasure for detecting replay attacks and hence valuable contribution of this work.

In addition to LP residual-based implicit excitation source features, this work has also explored explicit excitation source information such as fundamental frequency (F_0) and glottal flow derivative cepstral coefficients (GFDCC) for replay detection task. The F_0 or pitch contour obtained by zero band filtering (ZBF) approach [33] is used as feature for replay detection in this work. The GFDCC

feature is obtained by short-term processing of the glottal flow derivative (GFD) signal and provides compact representation of glottal information using mel-cepstral analysis. GFD signal is obtained from speech signals using the iterative adaptive inverse filtering (IAIF) method [34]. As per the authors' knowledge, pitch contour and GFD signals are explored for the first time in this work for a replay detection task, thus adding novelty to this work. The GFDCC together with CQCC provide 9.34% EER, and hence approach state-of-the-art spoof detection results (see [Table 2.1](#)).

The rest of the chapter is organized as follows: [Section 2.2](#) describes the effect of intermediate devices and transmission medium over replay signal characteristics. Different source and vocal-tract features used for a replay detection task are given under [Section 2.3](#). The ASV and spoof detection experimental study, followed by useful discussion, is provided in [Section 2.4](#). The summary and future scope of the work are reported in [Section 2.5](#). The acknowledgment and list of refereed articles are reported at the end.

2.2 Replay speech signal

Replay recordings are simply the playback of prerecorded original speech samples. Consequently, amplitude-frequency characteristics of replay signals are affected by recording and playback device response. In addition, environmental acoustic conditions may also produce distortions in the replay signals. This section describes the effect of a recording device, a playback device, and the environmental acoustic conditions on the replay signal characteristics.

The recording and playback device setup contains a chain of different components, such as a microphone, loudspeakers, amplifiers etc., and performs a sequence of transformation on the input speech signal. Recording device components mainly include a microphone that converts acoustic waveforms into electrical waveforms and an analog-to-digital converter for storing digital waveforms. The playback device components include a digital-to-analog converter and amplifiers to drive the loudspeaker, which converts stored digital waveforms back into acoustic pressure waves. Each component of the chain acts as a system that may cause distortion in the replay signal characteristics. For example, the following main flaws may be introduced during replay signal generation.

- Low-frequency distortions during reproduction of the replay speech from the loudspeaker.
- High-frequency imperfections around Nyquist frequency due to additional antialiasing filtering process while rerecording through a microphone [17].
- Reflection and reverberation noise from the environment acoustic conditions during collection of speech through the microphone.

All the above-mentioned points show the effects of different intermediate audio components on the replay signal characteristics. Mathematically, replay signal $s_{replay}(n)$

can be expressed as convolution (“*”) of actual speech $s(n)$ and impulse response of the replay configuration $h_{RC}(n)$, and is given by

$$s_{replay}(n) = s(n) * h_{RC}(n) \quad (2.1)$$

where $h_{RC}(n)$ represents the convolution of impulse responses of recording device $h_{mic}(n)$, recording environment $a(n)$, playback device $h_{pd}(n)$, playback environment $b(n)$, and is given by

$$h_{RC}(n) = h_{mic}(n) * a(n) * h_{pd}(n) * b(n) \quad (2.2)$$

According to LP theory, the actual speech signal $s(n)$ is expressed as

$$s(n) = \hat{s}(n) + r(n) \quad (2.3)$$

and

$$\hat{s}(n) = \sum_{k=1}^p a_k s(n-k) \quad (2.4)$$

where $\hat{s}(n)$ models the vocal-tract component of the actual speech signal in terms of LP coefficients “ $a_k, k = 1, 2, \dots, p$,” and error in the prediction $r(n)$, called as LP residual models the excitation component [35, 36]. By employing Eqs. (2.1), (2.3), (2.4), the replay speech signal $s_{replay}(n)$ is expressed as

$$s_{replay}(n) = \left[\sum_{k=1}^p a_k s(n-k) + r(n) \right] * h_{RC}(n) \quad (2.5)$$

$$\implies s_{replay}(n) = \sum_{k=1}^p a_k s(n-k) * h_{RC}(n) + r(n) * h_{RC}(n) \quad (2.6)$$

Eq. (2.6) shows that both vocal-tract (system) and excitation (source) components of input speech signal $s(n)$ are affected by $h_{RC}(n)$. Thus, together they can be used to develop generalized countermeasures to replay attacks.

2.3 Features used for replay detection

This section describes different excitation source (implicit or explicit) and system features for building replay detection systems and the motivation behind using them.

2.3.1 LP residual-based implicit source features

In the LP model of speech, a speech sample $s(n)$ is predicted as a linear weighted combination of previous p speech samples, and is expressed as

$$\hat{s}(n) = - \sum_{k=1}^p a_k s(n-k) \quad (2.7)$$

where $\hat{s}(n)$ is the predicted speech sample, a_k s are LP coefficients (LPCs), and p is the order of prediction. The difference (error) between the actual and predicted speech signal is known as LP residual $r(n)$ and is given by

$$r(n) = s(n) - \hat{s}(n) = s(n) + \sum_{k=1}^p a_k s(n-k) \quad (2.8)$$

The complex analytic signal $r_a(n)$ corresponding to $r(n)$ is given by [31]

$$r_a(n) = r(n) + j r_h(n) \quad (2.9)$$

where $r_h(n)$ is the Hilbert transform of LP residual $r(n)$.

The Hilbert envelope $h(n)$ is the magnitude of analytical signal and is expressed as

$$h(n) = \sqrt{r^2(n) + r_h^2(n)} \quad (2.10)$$

The residual phase is defined as the cosine of the phase function of the analytical signal and is given by

$$\cos\theta(n) = \frac{\text{real}(r_a(n))}{h(n)} = \frac{r(n)}{h(n)} \quad (2.11)$$

2.3.1.1 RMFCC feature

The *RMFCC feature* is extracted from the LP residual $r(n)$ via short-term mel-cepstral analysis, as shown in Fig. 2.2. The preprocessing stage involves the frame splitting of speech signal with 20 ms framesize and 10 ms frameshift. No voice-activity detection

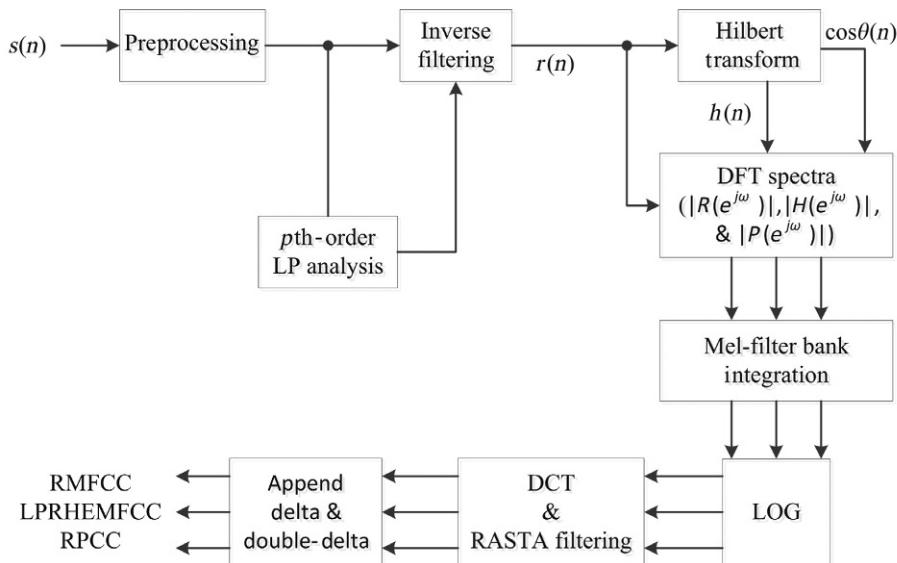


FIG. 2.2

Block diagram for RMFCC, LPRHEMFCC, and RPCC features extraction process. No. of mel-filters = 24, framesize = 20 ms, frameshift = 10 ms. ($p = 10$) for 8 kHz, ($p = 18$) for 16 kHz.

(VAD) has been performed as unvoiced frames may also persist useful channel information for replay detection [21]. Then, the LP residual signal is obtained using the LP analysis method. Discrete fourier transform (DFT) is performed to obtain the LP residual spectrum. The magnitude of LP residual spectra is passed through nonuniform triangular band pass filter banks placed on the mel-frequency scale. At the end, discrete cosine transform (DCT) and RASTA (relative spectra) filtering is applied on the logarithm of the subband energies obtained from mel-filters bank to obtain the RMFCC features. If $R(e^{j\omega})$ is the spectrum of the LP residual $r(n)$, the magnitude of which is passed through mel-filters bank (M_{el}) for subband energy calculations, then the RMFCC feature is computed as

$$RMFCC = DCT[\log(M_{el}(|R(e^{j\omega})|))] \quad (2.12)$$

The source feature “RMFCC” involves frame-based cepstral domain processing of the raw LP residual using 20 ms framesize and 10 ms overlap, and thereby models the glottal information averaged over two to three pitch periods [37].

2.3.1.2 LPRHEMFCC feature

The *LPRHEMFCC feature* involves short-term mel-cepstral processing of the Hilbert envelope of the LP residual (Fig. 2.2). If $H(e^{j\omega})$ is the spectrum of the Hilbert envelope $h(n)$ of the LP residual, then the LPRHEMFCC feature is computed in the following way:

$$LPRHEMFCC = DCT[\log(M_{el}(|H(e^{j\omega})|))] \quad (2.13)$$

The Hilbert envelope of the LP residual represents amplitude information of the LP residual samples. In comparison with the LP residual, the amplitudes are emphasized in the Hilbert envelope at each epoch location (instant of major excitations) in a pitch period. As a result, the high-amplitude values at the epochs in the signal dominate the computation of subband energies obtained from the mel-filters banks [38]. Due to this, the obtained LPRHEMFCC feature mainly represents the source information especially at the epoch locations averaged over two to three pitch periods.

2.3.1.3 RPCC features

The source feature “RPCC” is obtained from short-term mel-cepstral domain processing of the residual phase ($\cos\theta(n)$). It represents source information present in the sequence of LP residual samples, and provides perceptually meaningful excitation specific evidences for replay detection. The feature extraction steps to obtain the RPCC feature are given in Fig. 2.2. If $P(e^{j\omega})$ is the spectrum of the residual phase, then the RPCC feature is computed in the following way:

$$RPCC = DCT[\log(M_{el}(|P(e^{j\omega})|))] \quad (2.14)$$

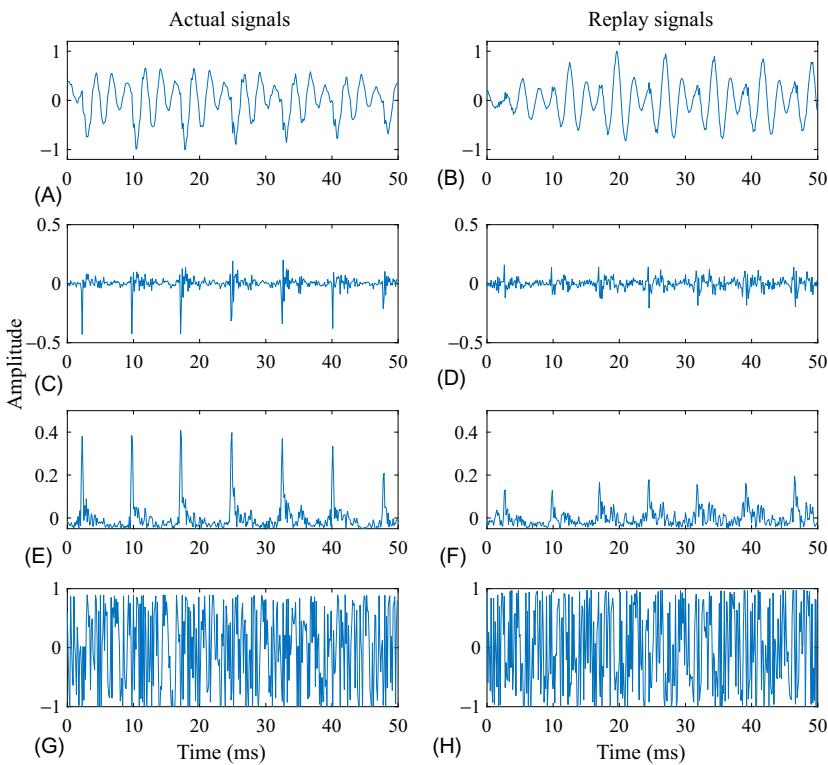


FIG. 2.3

(A, B) Actual and replay speech signals, (C, D) corresponding LP residuals, (E, F) Hilbert envelopes of the LP residuals, and (G, H) residual phase signals. The LP residuals are estimated using 10th-order LP analysis at 8 kHz sampling frequency.

The speech files are taken from the ASVspoof 2017 database, named T_1000196.wav and T_1003010.wav.

2.3.1.4 Usefulness of LP residual features for replay detection

The actual and corresponding replay parts of a speech signal, its LP residual, and the Hilbert envelope of the LP residual and residual phase are shown in Fig. 2.3. It may seem challenging to discriminate between actual and replay speech signals from the time domain representations (Fig. 2.3A and B). However, large fluctuations in amplitude levels occur in the LP residual of actual signals compared to replay signals, especially around GCIs (Fig. 2.3C and D). This corresponds to higher SNR around GCIs for actual speech compared to replay speech. The similar patterns are observed in the case of Hilbert envelopes of the LP residual (Fig. 2.3E and F). Further, it is difficult to see any discriminative feature from the plots of the residual phase (Fig. 2.3G and H). It represents the source like information present in the sequence of the LP residual samples. However, keen observations show that the residual phase corresponding to the replay signal looks denser compared to the actual case. The

proposed LP residual-based features are able to provide compact modeling of these discriminative characteristics from their respective residual signals using melcepstral analysis, and hence would be useful in a replay detection context.

2.3.2 Explicit source features

2.3.2.1 Pitch contour feature

The rate of vibration of the vocal folds is referred to as fundamental frequency (F_0), which contributes to the perceived pitch of the voiced speech [39]. The playback devices, specifically loudspeakers, contain diaphragms which have to move a relatively longer distance to reproduce the low-frequency components of the signal. Intuitively, some inertia may be associated with the mechanical movement of the loudspeaker diaphragm, which may result in a slighter shift in fundamental frequency F_0 during reproduction of speech. In addition, pitch dynamics (i.e., rate of change of pitch with respect to time) may also be affected. Hence, the pitch contour contains pitch and its dynamics can be useful for detecting replay speech signals. Fig. 2.4 shows pitch contours for an actual and corresponding replay speech signal. The pitch contour of an actual signal is almost smooth in nature, whereas the pitch values show a slighter shift in the case of replay signals from their original values. This distinction between the pitch contours of actual and replay signals can be helpful for developing replay detection systems.

2.3.2.2 GFDCC feature

The modulated air flow due to the vibration of vocal folds is termed glottal flow [40]. The glottal flow, also called “excitation” acts as an input to vocal tract system. The output of the vocal tract system is filtered by lip radiation during speech production. Lip radiation is equivalent to first-order differentiation in signal processing terms, and is typically included in the excitation (glottal flow) function [40]. Therefore, the input to the vocal tract becomes derivative of the glottal flow,

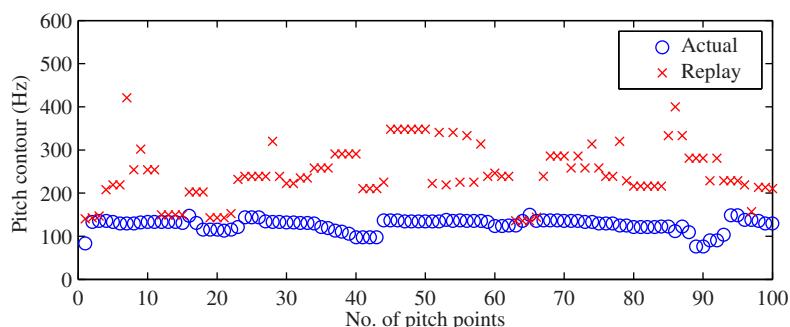


FIG. 2.4

Pitch contours for an actual and corresponding replay speech signals.

which we termed as a GFD signal. The intermediate devices in replay generation setup contain a chain of different components, such as a microphone, loudspeakers, amplifiers, input/output filters, etc., and perform a sequence of transformation on the input speech signal. In the event of replay simulation, the input signal may possibly undergo additional differentiation/integration operations (due to input/output filtering), which may change the shape of the GFD signal. Thus, it can be predicted that the shape and parameters of the GFD signal may vary from actual speech to replay speech, and hence the GFD signal can be explored for the replay detection task.

The *GFDCC feature* is extracted from the GFD signal $g(n)$ via standard short-term mel-cepstral analysis. The GFD signal is obtained from speech signals using the IAIF method [34]. Then, DFT of the GFD signal is computed and passed through a bank of 27 mel-spaced triangular band-pass filters to obtain subband energies. DCT is applied to logarithmic values of subband energies to obtain cepstral coefficients appended with their delta and double delta coefficients. If $G(e^{j\omega})$ is the spectrum of the GFD signal $g(n)$, the magnitude of which is passed through mel-filters bank (M_{el}) for subband energy calculations, then the GFDCC feature is computed as

$$GFDCC = DCT[\log(M_{el}(|G(e^{j\omega})|))] \quad (2.15)$$

2.3.3 System features

In this work, conventional MFCC and state-of-the-art CQCC derived from speech signals are explored as system features for replay detection tasks.

2.3.3.1 MFCC feature

The MFCC feature has been extracted from speech signals using standard short-term mel-cepstral analysis. It is based on conventional Fourier transform and the most common parametrization used in speech and speaker recognition. A bank of 24 triangular mel-filters is used to compute the subband energies. Then, DCT with RASTA filtering is applied to logarithmic values of these subband energies to obtain the MFCC feature vectors. The MFCC provides high low-frequency spectral resolution, thus it can be useful to capture information from the low-frequency regions for spoof detection tasks.

2.3.3.2 CQCC feature

The CQCC feature is based on the constant-Q transform (CQT) instead of the conventional Fourier transform [41, 42]. CQCC feature extraction applies geometrically spaced frequency bins, which leads to higher spectral resolution at lower frequencies and higher temporal resolution at higher frequencies. This pattern of frequency spacing represents the human perception system more closely [41, 42].

Mathematically, the CQT of discrete time sequence $x(n)$ is computed as follows [41, 42]:

$$X^{CQ}(k,n) = \sum_{j=n-N_k/2}^{n+N_k/2} x(j)b_k^*(j-n+N_k/2) \quad (2.16)$$

where $k = 1, 2, \dots, K$ is the index of the frequency bins, b_k^* denotes the complex conjugate of basis functions, b_k , and N_k represent variable window lengths.

The obtained CQT is then converted to a linear space and the standard CQCC feature is extracted via conventional cepstral analysis, and is given by

$$CQCC(q) = \sum_{l=1}^L \log |X^{CQ}(l)|^2 \cos \left[\frac{p(l-\frac{1}{2})\pi}{L} \right] \quad (2.17)$$

where $q = 0, 1, 2, \dots, L - 1$, and where l is newly resampled frequency bins [41, 42].

2.4 Experimental study

This section demonstrates the usefulness of proposed residual-based source features in detecting replay signals through ASV and spoof detection experiments.

2.4.1 Databases

The following two sections provide detailed description of the databases used in this study.

2.4.1.1 IITG-MV

In this study, the replay database is manually developed by using the publicly available Indian Institute of Technology Guwahati Multi-Variability (IITG-MV) speaker recognition database [43]. The Phase-I (office) and Phase-II (laboratory) datasets of IITG-MV database are collected using five different microphone sensors in multiple environment conditions and in different sessions. Therefore, they are suitable for robust SV, to design a database for replay attacks and antispoofing studies like the RSR database [44]. The speech samples are collected in two modes: In *read speech* mode, the participant reads predefined paragraphs of about 3–5 min duration. In *conversational speech* mode, the participant speaks freely about any topic for 10–15 min. In order to create replay recordings, we prefer the latter mode for the following reasons. The speaker characteristics including behavioral traits are relatively better manifested in conversational speech, and in general the replay attacker preferably acquires the speech samples (secretly) while the target is in conversation with others.

The Phase-I and Phase-II datasets of the IITG-MV database contain 148 (112 males and 36 females) nonnative English speakers' speech samples, recorded at the rate of 16,000 samples/s. The duration of the speech samples per speaker varies

from 10 to 15 min. For this experimental study, we consider 76 (45 males and 31 females) speakers' speech data and segregate the data into two groups: Dataset-I and Dataset-II. Dataset-I includes 5 male and 6 female speakers' speech data, amounting to 1 h from each gender for building gender-dependent UBM models. Dataset-II is developed with 65 speakers speech data (comprising 40 males and 25 females) for evaluation purposes. Each speaker's first 2 min of speech data are used for enrollment. The remaining data are converted into several segments of 30 s duration and used for test trials. Each test segment of each speaker is used as a genuine trial for the same target model and an impostor trial against other speakers model of the same gender. This resulted in a huge number of trials. The detailed statistics are summarized in [Table 2.2](#). Altogether, there are 42,440 trials, which include 1274 genuine and 41,166 impostor trials. Spoofing an ASV system via a replay attempt requires speech recordings from the target claimants only. Hence, the number of replay trials is equal to the number of target genuine trials.

The replay speech samples are generated manually by replaying the original data through a high-quality CREATIVE-SBS-A35 loudspeaker (frequency response 100–15,000 Hz) almost in acoustically controlled environment (i.e., inside a closed room with no fan and air condition noise) and rerecorded through an in-built microphone of HD Webcam C270-Logitech at a sampling rate of 16,000 samples/s. We put very careful effort into acquiring good-quality replay speech samples in order to provide a more challenging scenario. To verify the quality of the replay data, the original and replay recordings were played in front of a few participants. They could hardly differentiate between them. This confirms that the collected replay recordings are of good quality.

The quality of the replay recordings can also be verified by estimating the distortion between actual and corresponding replay recordings using the cepstral distance method [45]. Cepstral distance (CSD) is an estimate of measuring distortion in the replay recordings from their corresponding actual signals. It represents the average Euclidean distance between the two recordings and is estimated using standard short-term cepstral analysis with a Hamming window of duration 20 and 10 ms overlap. The DC coefficient " c_0 " is omitted. Low CSD values characterize high-quality replay recordings. The mean and standard deviation of CSD values, estimated for 1274 trials (males and females), are given in [Table 2.3](#). This table also contains two additional columns, representing the CSD values for C1 and

Table 2.2 Summary of the dataset used in this work for genuine, impostor, and replay trials.

Statistics	Male	Female	Total
Background speakers	05	06	11
Target speakers	40	25	65
Genuine trials	706	568	1274
Impostor trials	27,534	13,632	41,166
Replay trials (target claimants)	706	568	1274

Note: The speech samples are taken from IITG-MV database [43].

Table 2.3 Quality verification of the developed IITG-MV replay database with respect to the standard ASVspoof 2017 database using following parameters: number of genuine and replay trials, mean and standard deviation of cepstral distance (CSD) between actual and replay recordings, and quality of playback and recording device.

Parameters	Replay database		
	IITG-MV	ASVspoof 2017	
		C1	C3
No. of genuine trials	1274	1438	2363
No. of replay trials	1274	1438	2363
CSD (μ)	0.80	0.79	0.77
CSD (σ)	0.16	0.16	0.28
Playback device quality	H	L	L/M
Recording device quality	M	L/M	L/M

Note: L=low, M=medium, H=high.

C3 out of six evaluation conditions (C1–C6) of the ASVspoof 2017 database (see Table 5 and Fig. 2 in [5]). Conditions C1 and C3 represent comparatively low category replay trials with substantial spectral distortion. It can be observed that CSD values for the trials of the IITG-MV database are in closed matching with CSD values of the trials under either C1 or C3 evaluation conditions of the ASVspoof 2017 database. Although, both C1 and C3 are of low category but show wide variations in replay detection performance among top 10 systems, thereby simulates challenging evaluation conditions. From this aspect, the developed IITG-MV replay database provides relatively homogeneous but challenging evaluation conditions similar to either the C1 or C3 category of ASVspoof 2017 database. Hence, it can be considered a useful database for ASV as well as replay spoofing and countermeasure studies. In addition, it facilitates the vulnerability study of ASV systems to replay attacks gender-wise.

2.4.1.2 ASVspoof 2017

The ASVspoof 2017 database involves speech files from the original *RedDots* corpus and corresponding replayed versions. The original speech corpus is replayed and rerecorded through various different kind of playback devices and microphones, and in different acoustic environments. Thus, it provides highly varying replay attack conditions. The database consists of three nonoverlapping subsets: train, development, and evaluation; the details are given in Table 2.4. The speech files and corresponding replay recordings are collected at 16 kHz with resolution 16-bit per sample.

Table 2.4 Details of the ASVspoof 2017 database used for replay detection experiments.

Database statistics	Number of speakers	Speech files	
		Actual	Replay
Train set	10	1508	1508
Development set	8	760	950
Evaluation set	24	1298	12,008

2.4.2 Experimental setup

This section describes the details of features, classifiers, and experimental setups used for SV and spoof detection experiments in this study.

2.4.2.1 Features

Three LP residual-based features—*RMFCC*, *LPRHEMFCC*, and *RPCC*—have been extracted using short-term mel-cepstral analysis as shown in Fig. 2.2. Thus, all three features are derived from all types of speech frames as unvoiced frames also contribute toward the replay detection task along with voiced frames [21, 46]. The *GFDCC* features are derived from the voiced speech frames using short-term cepstral analysis. *Pitch contour*, a one-dimensional feature vector, is obtained from the voiced speech frames using the ZBF approach [33]. The *MFCC* feature has been extracted from all speech frames (voiced or unvoiced) using standard short-term mel-cepstral analysis and 24 channel triangular filter banks. The *CQCC* [42] feature has been extracted by keeping the same baseline configuration provided in the ASVspoof 2017 challenge. It is based on the CQT, which results in variable time-frequency resolution across the entire speech spectrum and thus captures useful informative characteristics for spoof detection. The CQCC feature is postprocessed by the cepstral mean variance normalization technique.

2.4.2.2 Classifiers

In this work, classical GMM-UBM is used at model level for ASV experiments on the IITG-MV database. State-of-the-art modeling techniques, such as i-vector and related frameworks, are available but require a large amount of data for training. In contrast, GMM-UBM is a standard and popular modeling approach, and works satisfactorily with a relatively small amount of training data [47]. GMM-UBM outperformed i-vector particularly for unknown types of spoof attacks as reported in the study by Hanilçi et al. [48]. Further, present work focuses on exploration of discriminatory evidence more at feature level than model level. From these aspects, use of the GMM-UBM classifier at model level seems more suitable for this study. For spoof detection experiments on the ASVspoof 2017 database, a two-class GMM-classifier is used to discriminate between actual and replay speech samples.

2.4.2.3 Setups for ASV and spoof detection experiments

In ASV studies, the speech signals are generally processed at sampling frequency, $F_s = 8$ kHz [37]. We therefore performed the ASV experiments on the IITG-MV database at $F_s = 8$ kHz. In general, as a rule of thumb ($F_s + 2$, where “ F_s ” is in kilohertz), order is followed in the literature for LP analysis of the speech signal [35, 36, 38]. Accordingly, 10th-order LP analysis is used to extract LP residual from speech signals sampled at 8 kHz. The MFCC feature is extensively used as a state-of-the-art for ASV task and thus considered as a baseline feature for ASV experiments. As per our experience in ASV studies, 39-dimensional (13 static, excluding $c_0 + \Delta + \Delta\Delta$) feature vectors provide good speaker recognition accuracy and hence are considered for our experiments.

The current stand-alone spoof detection methodology involves processing of speech files at their original $F_s = 16$ kHz for experiments using the ASVspoof 2017 database. Therefore, we perform the spoof detection experiments with $F_s = 16$ kHz and accordingly 18th-order LP analysis is employed. This facilitates comparison of our results with the recent prior works related to the replay detection task. Further, we experimentally experience that the improvements are comparatively significant for 57-dimensional features and considered in most of the spoof detection studies. Accordingly, 57-dimensional (19 static, excluding $c_0 + \Delta + \Delta\Delta$) feature vectors are used to conduct spoof detection experiments. Table 2.5 provides a summary of individual systems and their configurations used for ASV and spoof detection experiments. The number of Gaussian mixtures is chosen with optimal choice.

Table 2.5 Summary of the experimental setups of the systems for ASV and spoof detection experiments.

Databases	Systems (classifiers)	Features	Dimension
IITG-MV	ASV (GMM-UBM) 256 Gaussian components	MFCC RMFCC LPRHEMFCC RPCC GFDCC	(13-S+13- Δ +13- $\Delta\Delta$)
ASVspoof 2017	Spoof detection (GMM) 512 Gaussian mixtures 2 Gaussian mixtures	MFCC RMFCC LPRHEMFCC RPCC GFDCC CQCC Pitch contour	(19-S+19- Δ +19- $\Delta\Delta$) 1-dimensional

S , Δ , and $\Delta\Delta$ stand for static (excluding first energy term c_0), delta, and delta-delta coefficients, respectively.

2.4.3 Evaluation process

The following evaluation metrics have been used for evaluating the features' performances in this study.

2.4.3.1 Equal error rate and detection error tradeoff

The SV performance is generally shown by a detection error tradeoff (DET) curve and measured in terms of EER, where the false rejection rate (FRR) and false acceptance rate (FAR) are equal [49]. In false rejection, a genuine is classified as an impostor while in false acceptance, an impostor is accepted as a genuine speaker. Under a replay spoofing scenario, the replay attackers usually aim at specified (target) speakers and thereby increase the FAR of the system. Thus, under replay attacks, the FAR is a more relevant measuring parameter for evaluating the system performance. Accordingly, we have used two metrics to evaluate the system performance under both the baseline and spoofing test conditions: zero-effort false acceptance rate (ZFAR) and replay attack false acceptance rate (RFAR). ZFAR and RFAR are related to zero-effort impostor trials and replay trials, respectively.

The baseline ZFAR or equivalently the EER performance is computed by pooling all genuine and zero-effort impostor trials together. The RFAR performance is

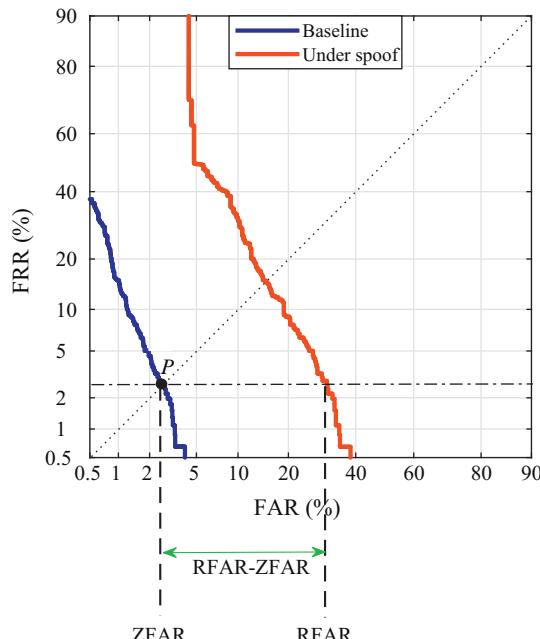


FIG. 2.5

Synthetic example showing computation of ZFAR and RFAR using a decision threshold fixed at EER point (P) of the baseline system.

computed using the replay trials only, under replay attacks. The computation of RFAR is based on the fixed decision threshold (at EER point) of the baseline system, as shown in Fig. 2.5. As the same baseline ASV system is used for both ZFAR and RFAR computation, the difference $RFAR-ZFAR$ directly indicates system vulnerability to replay attacks [3]. In a positive sense, it represents ASV systems' capability to resist spoof attacks. A smaller value of $RFAR-ZFAR$ indicates better replay detection accuracy. For a foolproof spoof-resistant ($RFAR = 0$) ASV system, the $RFAR-ZFAR$ will approach to (-ZFAR). Moreover, since the same ASV system is used for both baseline and spoofing tests, the scores and decisions for all genuine trials will remain unaffected. Consequently, the FRR will remain constant, under both test conditions. Altogether, ZFAR, RFAR, and their difference $RFAR-ZFAR$ can be used as evaluation metrics to compare the different ASV systems' performance under a replay spoofing scenario.

2.4.3.2 Tandem-detection cost function

Tandem-detection cost function (t-DCF) is a recently proposed performance evaluation scheme [50]. It is an extension of conventional DCF used in ASV research into scenarios involving spoofing attacks. The present work is related to detection of replay trials directly on ASV systems *without any countermeasures*. Accordingly, the suitable empirical formula for t-DCF can be expressed as

$$\begin{aligned} t - DCF(t) = & C_{miss}^{ASV} \cdot \pi_{tar} \cdot P_{miss}^{ASV}(t) + C_{fa}^{ASV} \cdot \pi_{non} \cdot P_{fa}^{ASV}(t) \\ & + C_{fa,spoof}^{ASV} \cdot \pi_{spoof} \cdot (1 - P_{miss,spoof}^{ASV}(t)) \end{aligned} \quad (2.18)$$

where

$$\pi_{tar} + \pi_{non} + \pi_{spoof} = 1 \quad (2.19)$$

- C_{miss}^{ASV} is the cost of ASV system rejecting a target trial.
- C_{fa}^{ASV} is the cost of ASV system accepting a nontarget trial.
- $C_{fa,spoof}^{ASV}$ is the cost of ASV system accepting a spoof trial.
- Other parameters $P_{miss}^{ASV}(t)$, $P_{fa}^{ASV}(t)$, $P_{miss,spoof}^{ASV}(t)$, π_{tar} , π_{non} , and π_{spoof} have their usual meanings as given in [50].

Eq. (2.18) represents that the t-DCF performance is computed by pooling all genuine, zero-effort impostor trials and replay trials together. Hence, t-DCF given in Eq. (2.18) can be used as a standard measure to evaluate the ASV performance involving replay attacks. This is very much consistent with the work method presented in this chapter.

2.4.4 Experimental results and discussion

This section presents the experimental results for IITG-MV replay database and standard ASVspoof 2017 database, followed by useful discussions based on the obtained results.

Table 2.6 ASV results for different features and their combinations using GMM-UBM classifier.

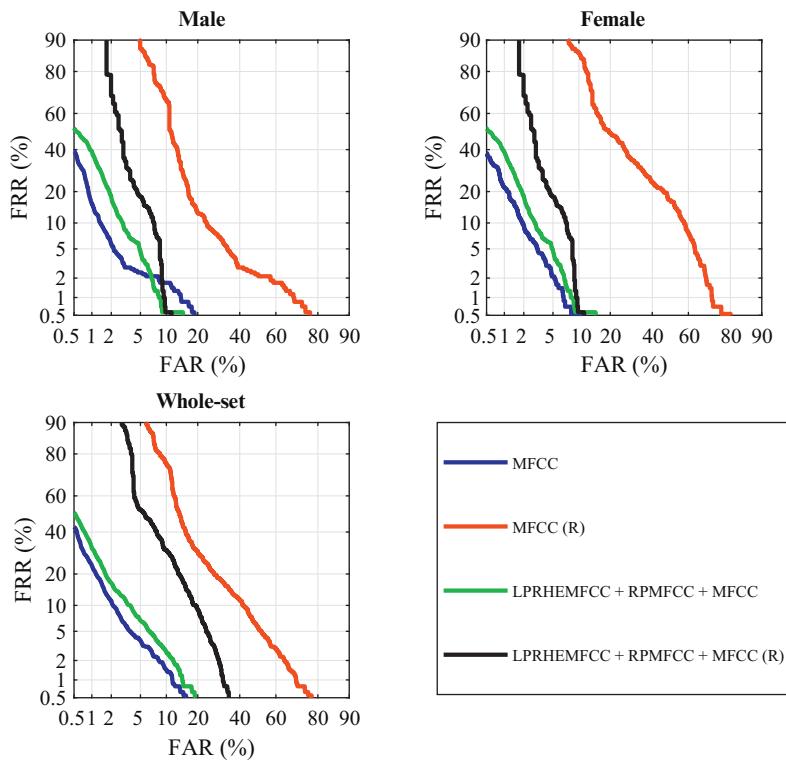
Features	ZFAR (%)	RFAR (%)	Difference (RFAR-ZFAR)	t-DCF ($\pi_{spoof} = 0.05$)
Male				
MFCC	2.97	38.81	35.84	0.225
RMFCC	5.38	15.72	10.34	0.136
LPRHEMFCC	12.62	7.93	-4.69	0.170
RPCC	12.74	7.51	-5.23	0.170
(LPRHEMFCC + RPCC)	8.22	7.79	-0.43	0.124
GFDCC	4.39	14.02	9.63	0.117
RMFCC + MFCC	2.97	29.46	26.49	0.158
(LPRHEMFCC + RPCC) + MFCC	4.82	8.50	3.68	0.094
GFDCC + MFCC	2.69	21.95	19.26	0.138
Female				
MFCC	3.69	65.14	61.45	0.364
RMFCC	5.46	51.40	45.94	0.314
LPRHEMFCC	10.78	22.00	11.22	0.220
RPCC	9.68	56.34	46.66	0.382
(LPRHEMFCC + RPCC)	8.45	34.86	26.41	0.262
GFDCC	6.51	52.99	46.48	0.332
RMFCC + MFCC	3.69	61.44	57.75	0.335
(LPRHEMFCC + RPCC) + MFCC	5.28	47.18	41.90	0.291
GFDCC + MFCC	4.40	62.50	58.10	0.359
Whole-set				
MFCC	4.24	54.08	49.84	0.314
RMFCC	5.65	31.16	25.51	0.214
LPRHEMFCC	13.20	8.00	-5.20	0.177
RPCC	12.23	27.63	15.40	0.265
(LPRHEMFCC + RPCC)	9.81	18.52	8.71	0.194
GFDCC	6.28	30.69	24.41	0.219
RMFCC + MFCC	3.80	41.20	37.51	0.237
(LPRHEMFCC + RPCC) + MFCC	5.80	23.00	17.20	0.175
GFDCC + MFCC	4.55	40.97	36.42	0.252

Notes: The features are combined at score level using a simple weighted linear combination scheme with equal weights. The italic results are given for quick comparison of combinations RMFCC+MFCC, (LPRHEMFCC+RPCC)+MFCC, and GFDCC+MFCC.

2.4.4.1 Results on the IITG-MV database

Table 2.6 shows the results of ASV experiments conducted on the IITG-MV database using different features and their combinations under both the baseline and replay spoofing test conditions. In the case of zero-effort impostor trials, the ASV performance is expressed in terms of ZFAR. Under replay attacks the performance is expressed in terms of RFAR. The t-DCF performance considers pooling of all genuine, zero-effort impostor and replay trials together. With reference to ZFAR values, the corresponding RFAR values are higher for MFCC, RMFCC, and GFDCC features in all cases. In contrast, opposite patterns are obtained for LPRHEMFCC and RPCC features particularly in male speakers' case. Usually, an ASV system shows a higher RFAR value (under replay attacks) compared to the respective baseline ZFAR value, which is not the case with LPRHEMFCC- and RPCC-based ASV systems. For a foolproof spoof-resistant ASV system, the RFAR value should be zero (see [Section 2.4.3.1](#)). In this regard, a smaller value of RFAR with respect to corresponding ZFAR for these features shows their strong caliber toward the detection of replay samples. This can also be validated through comparing respective computed t-DCF values across different features given in the last column of [Table 2.6](#).

The contribution of this work is better reflected as (source features + MFCC) combinations. From the fourth and fifth columns of [Table 2.6](#), the (RFAR-ZFAR, t-DCF) performances of (26.49%, 0.158), (3.68%, 0.094), and (19.26%, 0.138) are achieved for RMFCC + MFCC, (LPRHEMFCC + RPCC) + MFCC, and GFDCC + MFCC combinations, respectively, in the case of male speakers. Similarly, (57.75%, 0.335), (41.90%, 0.291), and (58.10%, 0.359) are achieved for RMFCC + MFCC, (LPRHEMFCC + RPCC) + MFCC, and GFDCC + MFCC combinations, respectively, in the case of female speakers. For the whole set, the respective values are (37.51%, 0.237), (17.20%, 0.175), and (36.42%, 0.252). The substantial improvements in the (RFAR-ZFAR, t-DCF) performances have been recorded for the (source features + MFCC) combination over the stand-alone MFCC feature. Thus, source features complement the MFCC feature under the replay spoofing scenario. Further, the relative (RFAR-ZFAR) improvements of 86.10% (males), 27.45% (females), and 54.14% (whole-speaker-set) have been achieved for the (LPRHEMFCC + RPCC) + MFCC combination compared to the RMFCC + MFCC combination, whereas in terms of the t-DCF metric, the obtained relative improvements are 40.50% (males), 13.13% (females), and 26.16% (whole-set). This indicates the usefulness of processing LP residual in terms of its components (Hilbert envelope and residual phase) rather than direct processing of the raw LP residual signal for the replay detection task. Moreover, the performance for female speakers is worse under replay attacks compared to male speakers. This may be due to less spectral (harmonics) distortion caused by playback devices in the case of female speech as they fewer harmonics than male speech [51]. The DET plots in [Fig. 2.6](#) show ASV performance of the (LPRHEMFCC + RPCC) + MFCC combination and the stand-alone MFCC feature under baseline and spoof attacks scenarios. A small gap in the baseline performance and a large gap in the performance under spoof attacks reflect the significance of the

**FIG. 2.6**

DET plots showing ASV performance under baseline and replay attacks ("R") test scenario for the (LPRHEMFCC + RPCC) + MFCC combination and the stand-alone MFCC feature in all cases, that is, male, female, and whole-set. Here, baseline evaluation involves pooling of genuine trials and zero-effort impostor trials while evaluation in the spoofing scenario involves pooling of genuine trials and replay trials (as impostor trials).

fusion of source features along with MFCC in developing a spoof-resistant ASV system without using any stand-alone classifier for replay detection.

2.4.4.2 Results on ASVspoof 2017 database

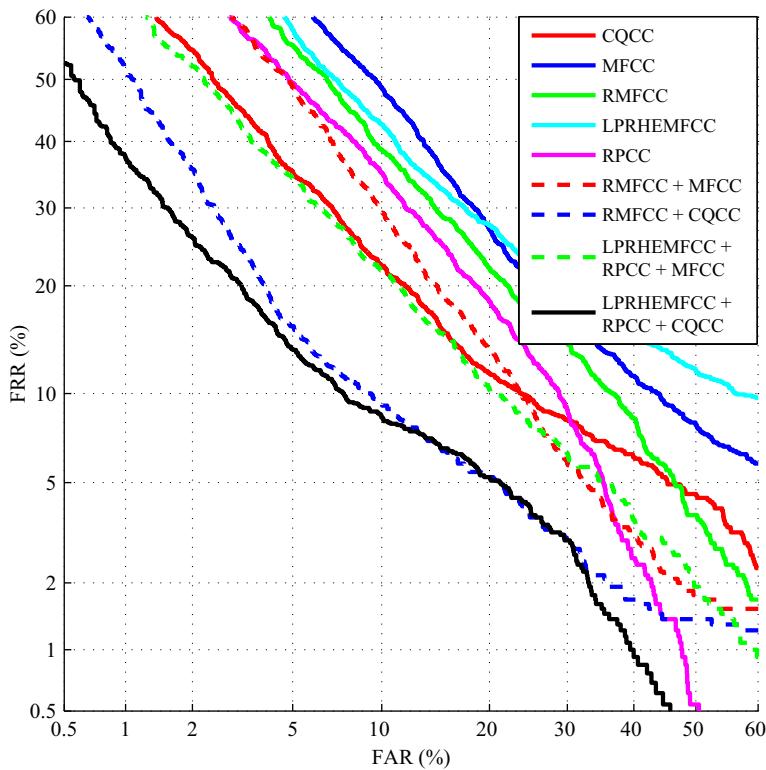
The obtained SV results on the IITG-MV database show the significance combined use of source and system features for the development of replay attack countermeasures. Nevertheless, the efficacy of the proposed systems should be validated under diverse replay attacks scenarios. In this direction, the experiments for the proposed features set are repeated using the pooled ASVspoof 2017 database. The systems are trained on a pooled (train + development) set and tested using the evaluation-set. The spoof detection results are presented in [Table 2.7](#). The source features RMFCC, (LPRHEMFCC + RPCC), Pitch contour, and GFDCC provides 20.88%, 17.48%,

Table 2.7 Spoof detection results for different features and their combinations using GMM classifier.

Features	EER (%)
MFCC	22.73
CQCC	15.16
RMFCC	20.88
LPRHEMFCC	23.65
RPCC	19.00
(LPRHEMFCC + RPCC)	17.48
Pitch contour	31.07
GFDCC	16.06
RMFCC + MFCC	16.80
(LPRHEMFCC + RPCC) + MFCC	14.83
Pitch Contour + MFCC	18.26
GFDCC + MFCC	14.80
RMFCC + CQCC	9.50
(LPRHEMFCC + RPCC) + CQCC	8.86
Pitch Contour + CQCC	11.16
GFDCC + CQCC	8.30
CILPR [23]	15.76
CILPR + CQCC [23]	9.41

The features are combined at score level using *bosaris_toolkit* [52]. The italic result is the best one for this work on ASVspoof 2017 database.

31.07%, and 16.06% EER values. When combined with MFCC, the obtained respective EER values are 16.80%, 14.83%, 18.26%, and 14.80%. Similarly, when combined with CQCC, the obtained respective EER values are 9.50%, 8.86%, 11.16%, and 8.30%. Thus, substantial improvements have been observed for (source + system) feature combinations compared to a stand-alone system features-based replay detection system. This signifies the usefulness of the combined use of source and system information-based features for developing a generalized replay attack detection system. The DET plots corresponding to EER performances for individual features and combinations are given in Figs. 2.7 and 2.8. The (LPRHEMFCC + RPCC) + MFCC system gives better performance compared to the RMFCC + MFCC system. Likewise, the (LPRHEMFCC + RPCC) + CQCC combination outperforms the RMFCC + CQCC combination. These observations confirm joint use of the Hilbert envelope and residual phase in rejecting replay spoofing trials over the raw LP residual signal. Further, Table 2.7 compares the proposed source features set in this work with the recent source features-based work [23]. Even though the stand-alone performance of the proposed (LPRHEMFCC + RPCC) and GFDCC features is less than CILPR, the (LPRHEMFCC + RPCC) + CQCC (EER = 8.86%) and GFDCC + CQCC (EER = 8.30%) combinations outperform with respect to the CILPR + CQCC (EER = 9.41%) combination.

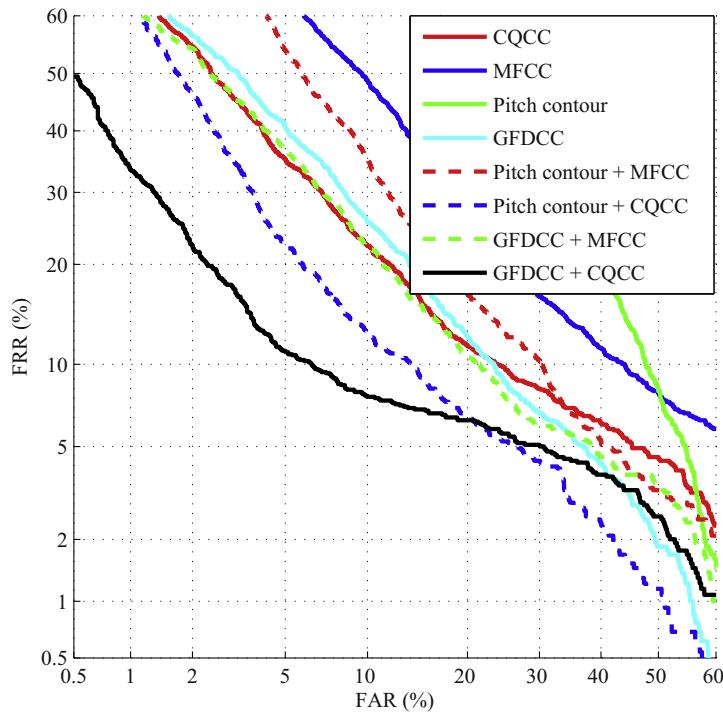
**FIG. 2.7**

DET plots representing EER performances of implicit LP residual-based source features in isolation and in combination with system features MFCC and CQCC.

2.4.4.3 Discussions

Useful remarks from the experimental outcomes are as follows:

- The (LPRHEMFCC + RPCC) feature shows better performance compared to the RMFCC feature. Thus, combination of independently processed LP residual components, that is, the Hilbert envelope and residual phase, is more beneficial compared to directly processed raw LP residual, in a replay detection context.
- Combining evidence from the different source features along with MFCC improves RFAR-ZFAR performance by a considerable amount with respect to the stand-alone MFCC feature.
- It is easy to fool female speakers compared to male speakers.
- (LPRHEMFCC + RPCC) + CQCC, GFDCC + CQCC, and RMFCC + CQCC combinations provide 8.86%, 8.30%, and 9.50% EERs, respectively. The obtained results are almost at par with the results reported for state-of-the-art techniques.

**FIG. 2.8**

DET plots representing EER performances of explicit source features in isolation and in combination with system features MFCC and CQCC.

- Pitch contour provides 31.07% EER in isolation. On fusion with MFCC (CQCC), the EER value is reduced from 22.73% (15.16%) to 18.26% (11.16%) for the MFCC (CQCC) feature. The improvement is substantial, thus it should be explored further, along with system features for developing replay countermeasures.
- Altogether, exploration of source information with suitable signal processing may evolve as a good candidate for developing generalized countermeasures to replay attacks near future. Further, as complementary information the excitation source features together with well-performed system features may contribute more toward the detection of replay signals.

2.5 Summary and future scope

This work demonstrates the use of implicit and explicit excitation source information in performance improvement of MFCC (CQCC)-based replay detection systems. For the IITG-MV database male dataset, the RFAR-ZFAR and t-DCF performances for MFCC feature are 35.84% and 0.225, respectively. When combined with LPRHEMFCC and RPCC, the achieved RFAR-ZFAR and t-DCF performances

are (3.68%, 0.094). For GFDCC + MFCC and RMFCC + MFCC combinations, the achieved (RFAR-ZFAR, t-DCF) performances are (19.26%, 0.138) and (26.49%, 0.158), respectively. Similar behavior is observed for the female dataset. However, performances are relatively poor compared to the male dataset. This indicates that spoofing female speakers is relatively easier.

The outcomes of ASV experiments on the IITG-MV database have been validated through spoof detection experiments on the standard ASVspoof 2017 database. The MFCC (CQCC) feature provides 22.73% (15.16%) EER, which is reduced to 14.83% (8.86%) when fused with (LPRHEMFCC + RPCC) at score level, and 14.80% (8.30%) when fused with the GFDCC feature. This indicates that the excitation source features complement the filter-based spoof detection systems significantly, and hence should be explored further. Thus, excitation-based solutions together with filter-based features are essential to develop generalized countermeasures against a wide variety of replay attacks. On the ASVspoof 2017 database, the LRHEMFCC + RPCC (EER = 17.48%) outperforms the RMFCC feature (EER = 20.88%) by 3.40%. This reflects the usefulness of processing the LP residual signal in the form of combination (Hilbert envelope + residual phase) over direct processing of the raw LP residual samples in rejecting replay spoofing trials.

The presented work in this chapter provides a motivation toward the development of a spoof resistance ASV system which will handle the spoofing and ASV problems together. Further, a self-developed IITG-MV replay database can be a useful contribution for the speech research community toward the development of a spoof-resistant ASV system. Compared to conventional MFCC features' extraction steps, feature extraction techniques used here require only a few additional signal processing steps, which are computationally inexpensive. Hence, the proposed method in this work can easily be adapted in hand-held mobile devices for real-time applications. A future plan is to carry out deeper investigations on excitation source information and their joint use with various novel system features for further enhancement in ASV performance under replay attacks scenarios.

Acknowledgments

This research work is funded by Ministry of Electronics and Information Technology (MeitY), Government of India through the project "Development of Excitation Source Features Based Spoof Resistant and Robust Audio-Visual Person Identification System." The research work is carried out in the Speech Processing and Pattern Recognition (SPARC) laboratory at the National Institute of Technology Nagaland, Dimapur, India.

References

- [1] J.P. Campbell Jr, Speaker recognition: a tutorial, Proc. IEEE 85 (9) (1997) 1437–1462.
- [2] T. Kinnunen, H. Li, An overview of text-independent speaker recognition: from features to supervectors, Speech Commun. 52 (2010) 12–40.

- [3] Z. Wu, N. Evans, T. Kinnunen, J. Yamagishi, F. Alegre, H. Li, Spoofing and counter measures for speaker verification: a survey, *Speech Commun.* 66 (2015) 130–153.
- [4] Z. Wu, T. Kinnunen, N. Evans, J. Yamagishi, C. Hanilçi, Md. Sahidullah, A. Sizov, ASVspoof 2015: the first automatic speaker verification spoofing and countermeasures challenge, in: *Proceedings of Interspeech*, 2015, pp. 2037–2041.
- [5] T. Kinnunen, Md. Sahidullah, H. Delgado, M. Todisco, N. Evans, J. Yamagishi, K. A. Lee, The ASVspoof 2017 challenge: assessing the limits of replay spoofing attack detection, in: *Proceedings of Interspeech*, 2017, pp. 2–6.
- [6] N. Evans, T. Kinnunen, J. Yamagishi, Spoofing and countermeasures for automatic speaker verification, in: *Proceedings of Interspeech*, 2013, pp. 925–929.
- [7] R.G. Hautamäki, T. Kinnunen, V. Hautamäki, T. Leino, A.-M. Laukkonen, I-vectors meet imitators: on vulnerability of speaker verification systems against voice mimicry, in: *Proceedings of Interspeech*, 2013, pp. 930–934.
- [8] R.G. Hautamäki, T. Kinnunen, V. Hautamäki, A.-M. Laukkonen, Automatic versus human speaker verification: the case of voice mimicry, *Speech Commun.* 72 (2015) 13–31.
- [9] J. Lindberg, M. Blomberg, Vulnerability in speaker verification—a study of technical impostor techniques, in: *Proceedings of EUROSPEECH*, 1999, pp. 5–9.
- [10] J. Villalba, E. Lleida, Speaker verification performance degradation against spoofing and tampering attacks, in: *FALA 10 Workshop*, 2010, pp. 131–134.
- [11] P.L. De Leon, V.R. Apsingekar, M. Pucher, J. Yamagishi, Revisiting the security of speaker verification systems against imposture using synthetic speech, in: *Proceedings of ICASSP*, 2010, pp. 1798–1801.
- [12] P.L. De Leon, M. Pucher, J. Yamagishi, Evaluation of the vulnerability of speaker verification to synthetic speech, in: *Proceedings of Odyssey: The Speaker and Language Recognition Workshop*, 2010, p. 28.
- [13] J.-F. Bonastre, D. Matrouf, C. Fredouille, Artificial impostor voice transformation effects on false acceptance rates, in: *Proceedings of Interspeech*, 2007, pp. 2053–2056.
- [14] T. Kinnunen, Z.-Z. Wu, K.A. Lee, F. Sedlak, E.S. Chng, H. Li, Vulnerability of speaker verification systems against voice conversion spoofing attacks: the case of telephone speech, in: *Proceedings of ICASSP*, 2012, pp. 4401–4404.
- [15] J. Villalba, E. Lleida, Preventing replay attacks on speaker verification systems, in: *Proceedings of the International Carnahan Conference on Security Technology (ICCST)*, 2011, pp. 1–8.
- [16] Z.F. Wang, G. Wei, Q.H. He, Channel pattern noise based playback attack detection algorithm for speaker recognition, in: *Proceedings of the International Conference on Machine Learning and Cybernetics (ICMLC)*, 2011, pp. 1708–1713.
- [17] M. Witkowski, S. Kacprzak, P. Zelasko, K. Kowalczyk, J. Galka, Audio replay attack detection using high-frequency features, in: *Proceedings of Interspeech*, 2017, pp. 27–31.
- [18] P. Nagarseth, E. Khoury, K. Patil, M. Garland, Replay attack detection using DNN for channel discrimination, in: *Proceedings of Interspeech*, 2017, pp. 97–101.
- [19] K.N.R.K. Raju Alluri, A.K.V. Gangashetty, SFF Anti-spoof: IIIT-H submission for automatic speaker verification spoofing and countermeasures challenge 2017, in: *Proceedings of Interspeech*, 2017, pp. 107–111.
- [20] S. Jelil, R.K. Das, S.R.M. Prasanna, R. Sinha, Spoof detection using source, instantaneous frequency and cepstral features, in: *Proceedings of Interspeech*, 2017, pp. 22–26.

- [21] R. Font, J.M. Espin, M.J. Cano, Experimental analysis of features for replay attack detection-results on the ASVspoof 2017 Challenge, in: Proceedings of Interspeech, 2017, pp. 7–11.
- [22] G. Lavrentyeva, S. Novoselov, E. Malykh, A. Kozlov, O. Kudashev, V. Shchemelinin, Audio replay attack detection with deep learning frameworks, in: Proceedings of Interspeech, 2017, pp. 82–86.
- [23] S. Jelil, S. Kalita, S.R.M. Prasanna, R. Sinha, Exploration of compressed ILPR features for replay attack detection, in: Proceedings of Interspeech, 2018, pp. 631–635.
- [24] D. Li, L. Wang, J. Dang, M. Liu, Z. Oo, S. Nakagawa, H. Guan, X. Li, Multiple phase information combination for replay attacks detection, in: Proceedings of Interspeech, 2018, pp. 656–660.
- [25] P. Tapkir, H. Patil, Novel empirical mode decomposition cepstral features for replay spoof detection, in: Proceedings of Interspeech, 2018, pp. 721–725.
- [26] H. Sailor, M. Kamble, H. Patil, Auditory filterbank learning for temporal modulation features in replay spoof speech detection, in: Proceedings of Interspeech, 2018, pp. 666–670.
- [27] M. Kamble, H. Tak, H. Patil, Effectiveness of speech demodulation-based features for replay detection, in: Proceedings of Interspeech, 2018, pp. 641–645.
- [28] G. Suthokumar, V. Sethu, C. Wijenayake, E. Ambikairajah, Modulation dynamic features for the detection of replay attacks, in: Proceedings of Interspeech, 2018, pp. 691–695.
- [29] Z. Ji, Z.-Y. Li, P. Li, M. An, S. Gao, D. Wu, F. Zhao, Ensemble learning for countermeasure of audio replay spoofing attack in ASVspoof2017, in: Proceedings of Interspeech, 2017, pp. 87–91.
- [30] M. Singh, D. Pati, Linear prediction residual based short-term cepstral features for replay attacks detection, in: Proceedings of Interspeech, 2018, pp. 751–755.
- [31] K.S.R. Murty, B. Yegnanarayana, Combining evidence from residual phase and MFCC features for speaker recognition, IEEE Signal Process. Lett. 13 (1) (2006) 52–55.
- [32] J. Wang, M. Johnson, Residual phase cepstrum coefficients with application to cross-lingual speaker verification, in: Proceedings of Interspeech, 2012.
- [33] K.T. Deepak, S.R.M. Prasanna, Epoch extraction using zero band filtering from speech signal, Circuits Syst. Signal Process. 34 (7) (2015) 2309–2333.
- [34] P. Alku, Glottal wave analysis with pitch synchronous iterative adaptive inverse filtering, Speech Commun. 11 (2–3) (1992) 109–118.
- [35] J. Makhoul, Linear prediction: a tutorial review, Proc. IEEE 63 (4) (1975) 561–580.
- [36] S.R.M. Prasanna, C.S. Gupta, B. Yegnanarayana, Extraction of speaker-specific excitation information from linear prediction residual of speech, Speech Commun. 48 (2006) 1243–1261.
- [37] R.K. Das, S.R.M. Prasanna, Exploring different attributes of source information for speaker verification with limited test data, J. Acoust. Soc. Am. 140 (1) (2016) 184–190.
- [38] V.C. Raykar, B. Yegnanarayana, S.R.M. Prasanna, R. Duraiswami, Speaker localization using excitation source information in speech, IEEE Trans. Speech Audio Process. 13 (5) (2005) 751–761.
- [39] B. Yegnanarayana, K.S.R. Murthy, Event-based instantaneous fundamental frequency estimation from speech signals, IEEE Trans. Audio Speech Lang. Process. 17 (4) (2009) 614–624.

- [40] M.D. Plumpe, T.F. Quatieri, D.A. Reynolds, Modelling of glottal flow derivative waveform with application to speaker identification, *IEEE Trans. Speech Audio Process.* 7 (5) (1999) 569–586.
- [41] M. Todisco, H. Delgado, N. Evans, A new feature for automatic speaker verification anti-spoofing: constant Q cepstral coefficients, in: *Speaker Odyssey Workshop*, Bilbao, Spain, vol. 25, 2016, pp. 249–252.
- [42] M. Todisco, H. Delgado, N. Evans, Constant Q cepstral coefficients: a spoofing countermeasure for automatic speaker verification, *Comput. Speech Lang.* 45 (2017) 516–535.
- [43] B.C. Haris, G. Pradhan, S.R.M. Prasanna, R.K. Das, R. Sinha, Multivariable speaker recognition database in Indian Scenario, *Int. J. Speech Technol.* 15 (4) (2012) 441–453.
- [44] A. Larcher, K.A. Lee, B. Ma, H. Li, RSR2015: database for text-dependent speaker verification using multiple pass-phrases, in: *Proceedings of Interspeech*, 2012, pp. 1580–1583.
- [45] N. Nocerino, F. Soong, L. Rabiner, D. Klatt, Comparative study of several distortion measures for speech recognition, in: *Proceedings of ICASSP*, vol. 10 1985, pp. 25–28.
- [46] C. Hanilçi, Linear prediction residual features for automatic speaker verification anti-spoofing, *Multimed. Tools Appl.* 77 (2017) 1–13.
- [47] D.A. Reynolds, T.F. Quatieri, R.B. Dunn, Speaker verification using adapted Gaussian mixture models, in: *Digital Signal Processing*, vol. 10, 2000, pp. 19–41.
- [48] C. Hanilçi, T. Kinnunen, Md. Sahidullah, A. Sizov, Classifiers for synthetic speech detection: a comparison, in: *Proceedings of Interspeech*, 2015, pp. 2057–2061.
- [49] A. Martin, G. Doddington, T. Kamm, M. Ordowski, M. Przybocki, The DET curve in assessment of detection task performance, in: *Proceedings of the European Conference on Speech Communication Technology*, Rhodes, Greece, vol. 4 , 1997, pp. 1895–1898.
- [50] T. Kinnunen, K.A. Lee, H. Delgado, N. Evans, M. Todisco, Md. Sahidullah, J. Yamagishi, D.A. Reynolds, t-DCF: a detection cost function for the tandem assessment of spoofing countermeasures and automatic speaker verification, in: *Proceedings of Odyssey: The Speaker and Language Recognition Workshop*, 2018, pp. 312–319.
- [51] E. Pépiot, Male and female speech: a study of mean F0, F0 range, phonation type and speech rate in Parisian French and American English speakers, in: *Speech Prosody* 7, 2014, pp. 305–309.
- [52] The Bosaris toolkit [software package], <https://sites.google.com/site/bosaristoolkit>, 2013. Accessed 10 December 2013.

Hypergraph-based type theory for software development in a Cyber-Physical context

3

Nathaniel Christen

Linguistic Technology Systems, Fort Lee, NJ, United States

We take the view that all design has an impact on sustainability and all software has an impact on the world. Therefore, it is the responsibility of those who are involved in the creation of software to consider this impact carefully. The various efforts described above tackle a wide range of different research questions, often with very little overlap ... Some seek to encourage reductions in consumption of energy and material goods, or to support changes in purchasing behavior. Others seek to use software capabilities to build smarter (lower impact) infrastructure. However, there is a lack of common understanding of the fundamental concepts of sustainability and how they apply, and a need for a common ground and consistent terminology. As such, persistent misperceptions occur, as researchers and practitioners disagree over whether we're even asking the right questions [...] We lack a coherent framework with sound theoretical basis that can provide a well-understood trans-disciplinary basis for sustainability design. Sustainability Design and Software: The Karlskrona Manifesto [1, p. 5]

It is possible to study Cyber-Physical systems at the level of individual devices. Each device has its own mechanical properties, generates its own kind of data, and may require its own software to interpret and understand that data. As devices proliferate, so does the diversity of software via which users may access whatever data they generate.

It is also possible to see Cyber-Physical systems in a more holistic way. As Cyber-Physical networks proliferate, we can envision a rise in technologies that merge and integrate data from many kinds of devices and many different vendors.

Such an eventuality has already been contemplated, including in this volume. Teixeira et al., for example, argue that

Overall, the increase in sensors, devices, and appliances, in our homes, has transformed it into a rather complex environment with which to interact. This characteristic cannot be merely addressed by a matching set of device-dependent applications, turning the smart home into a set of isolated interactive artifacts. Hence, there is a strong need to unify this experience, blending this diversity into a unique interactive ecosystem. This can be tackled, to a large extent by the proposal of a unique, integrated, ubiquitous distributed smart home application capable of handling a dynamic set of sensors and devices and providing the different house occupants (e.g., children, teenagers, young adults, and elderly) with natural and simple ways of controlling and accessing information. However, the creation of such application presents a challenge, particularly due to the need to support natural and adaptive forms of interaction beyond a simple home dashboard application.

I will use the term *hub application* to describe software meeting the requirements of what Teixeira et al. call an “application capable of handling a dynamic set of sensors and devices.”

Hub applications must receive data from many kinds of devices. They must also respond properly to data once it is received. Each kind of device should thereby have a corresponding software component built around data generated by *that particular* device. For the sake of discussion, I will call such device-specific software components a *hub library*. Hub libraries need to bridge the low-level realm of Cyber-Physical signals (and the networks that carry them) with the high-level realm of software engineering: GUI components, data validation, fluid and responsive User Experience, and so forth. We may assume that hub applications will feature many hub libraries, and that implementing hub libraries will become an integral step in the process of deploying Cyber-Physical instruments.

Hub applications, in short, serve as central loci organizing collections of hub libraries. In this role, they have a significance beyond just supplying a User Interface to Cyber-Physical data. Hub libraries would provide a concrete artifact that engineers may consult to obtain information about device properties and expected behavior. Compared to (as Teixeira et al. put it) “a set of isolated interactive artifacts,” I believe hub applications and hub libraries, with a centralized architecture, are more conducive to rigorous, concrete representation of Cyber-Physical devices as technical products. This rigor can make Cyber-Physical systems more secure and trustworthy—hub applications may be used for testing and prototyping devices and their supporting code, even before they are brought to market.

This chapter is not explicitly about hub applications; here I will examine coding and code documentation techniques which are applicable in many contexts. However, Cyber-Physical hubs are a good case study in programming contexts where *Requirements Engineering* is an intrinsic architectural feature. I write this chapter, then, from the perspective of a programmer creating a hub library for some form of Cyber-Physical input. This programmer needs to express in code the physical and computational details specific to the device’s signals, functionality, and capabilities. The hub library should serve as a reference point, a proxy for the device itself, in that engineers may study the library as an indirect way of coming to understand the device. Given

these requirements, hub libraries need an especially rigorous development methodology, one that emphasizes strict documentation and verification of coding requirements.

I claim, also, that hub libraries are operationally similar to a different genre of software components: code libraries providing access to scientific datasets. In practice, most Cyber-Physical devices are products of Research and Development cycles that emanate from scientific and technological advances. Research data generated during an R&D phase may therefore be an originating source for data models which ultimately govern the deployed Cyber-Physical software. As a result, code libraries which systematically access research data may be seen as ancestral versions of hub libraries—even though hub libraries prototypically work with real-time input, whereas datasets are curated information spaces that are frozen in time, and potentially reused in multiple research projects.

In essence, libraries for accessing research datasets—belonging to what may be called “dataset applications”—are analogous to hub libraries whose real-time networking logic is subtracted out; their input data comes from static files, not from any kind of decentralized or wireless networks. Hub libraries actually have two roles: they are low-level drivers attuned to network signals, and also high-level processors transforming raw data into analyzable and visualizable representations. Hub libraries can inherit the logic for this second, high-level role from R&D data-set libraries. This means that published datasets, and their accompanying code, are an important foundation for establishing data models and coding practices that may propagate through Cyber-Physical systems’ subsequent deployment phases.

So, although I claimed to write this chapter from the perspective of a programming writing a hub library, it is more accurate to say that I am writing from the perspective of a programmer composing dataset applications—because that *is* my perspective in real life. I contend that dataset applications are a reasonable proxy for hub libraries, so that these perspectives will coincide for many practical purposes.

I will orient this chapter’s discussion toward the C++ programming language, which is arguably the most central point from which to consider the integration of concerns—GUI, device networking, analytics—characteristic of Cyber-Physical hub software. C++ is unique in having extensive resources traversing various programming domains, like native GUI components alongside low-level networking and logically rigorous data verification. For this reason, C++ is a reasonable default language for examining how these various concerns interoperate.

The (predominantly C++) demo code for this chapter includes several republished datasets in various technical and Cyber-Physical domains (bioacoustics, speech samples, and parsed language samples) together with “dataset applications” to access research data and model its properties. I hope these demonstrations serve to illustrate how future Cyber-Physical datasets might be organized. I have also supplemented the datasets with code bases operationalizing many of the theoretical paradigms I present in the second half of this chapter. For example, the code provides a modest but demonstrative scripting platform via a hypergraph-based Intermediate Representation based on (what I call) Channel Algebra. A few code samples are drawn from the demo and published here to illustrate some basic patterns in these hypergraph structures; interested readers may find these sources (perhaps in a revised form) discussed in much greater detail in the documentation for the demo code.

The first two sections in this chapter, however, are less theoretical and less code-oriented. These sections discuss hub applications and Cyber-Physical systems on a more practical level: what are representative examples of data structures and coding requirements that hub libraries will need to encapsulate? The subsequent three sections will then turn to computer code at a more theoretical level, outlining certain representational paradigms, such as Directed Hypergraphs, which I believe can yield more expressive and comprehensive models of coding structures and requirements. The conclusion will briefly discuss data-sharing initiatives, and how these may benefit from code-modeling paradigms such as I outline in this chapter.

3.1 Hub applications and gatekeeper code

To begin, I will speak in general terms about hub applications and about the characteristic coding challenges which derive from Cyber-Physical technologies' unique networking and safety requirements. Implementing software hubs introduces technical difficulties which are distinct from manufacturing Cyber-Physical instruments themselves—in particular, devices are usually narrowly focused on a particular kind of data and measurement, while software hubs are multi-purpose applications that need to understand and integrate data from many different kinds of devices. Hub applications also present technical challenges that are different from other kinds of software, even if these hubs are one specialized domain in the larger class of user-focused software.

Any software application provides human users with tools to access data and computer files interactively and visually, either locally (data encoded on the “host” computer running the software) or remotely (data accessed over a network). Computer programs can be generally classified as *applications* (which are designed with a priority to User Experience) and *background processes* (which often start and maintain their state automatically and have little input or visibility to human users, except for special troubleshooting circumstances). Applications, in turn, can be generally classified as “web applications” (where users typically see one resource at a time, such as a web page, and where data is usually stored on remote servers) and “native applications” (characterized by more complex GUI components, and by the ability to work with “local” data—data stored on users’ computers or accessible on a local network—instead of or in addition to data acquired from a web service).

From a software engineering point of view, I believe we should conceptualize hub software as native, desktop-style applications which leverage native GUI features. Hub applications therefore embody a fundamentally different User Experience than other kinds of Cyber-Physical access points, like touch screens or phone apps.

To cite a concrete example, Teixeira et al. describe a refrigerator “that notifies the user if the door stays accidentally open” and moreover “knows what is inside the refrigerator and [its] expiration dates.” Consider, then, how we would design a User Interface networking with “smart” refrigerators. A simple indicator showing whether doors are open is straightforward, but an interface listing the items inside is much more complicated. Once a refrigerator can detect signals emanating from food

items/containers, we can envision a list of items presented as a GUI component, perhaps one food item per line (each line, say, showing a picture, price, text description, and expiration date). This would require cross-referencing numeric codes, which might be broadcast by the items *inside* the refrigerator, against a database that would load their images, descriptions, dates, and prices.

One question is then where this database would be hosted, and how it would be updated (insofar as food companies develop new products fairly often; any static database could quickly get outdated). Food companies (or some middleware agent) would have to agree on a common format so that the refrigerator's access software can integrate data from many brands. Since a refrigerator can hold many items, the GUI would also need enough screen space (and maybe a multi-level design) for users to browse many artifacts comfortably: perhaps a line-by-line window supplemented with separate dialog windows for each item.

On the other hand, a phone app interfacing with that same data would be constrained by its lesser screen real estate and limited interactive modalities. With no room on-screen to show a complete list of items—and no obvious gesture to navigate between individual and multi-item views—such an app might choose to list only those items nearing their expiry date. In general, when adapting to phone-like usage patterns (smaller screen, brief but frequent user engagement), designers have to offer compact but curated snippets of information. That is, software is forced to anticipate what info carries the most user interest—expiry dates are probably most important to users when items are near perishing. This means that data mining, Artificial Intelligence, and other techniques for anticipating users' needs become proportionately more consequential: if the whole GUI design is premised on AI, then AI ceases to be just a useful tool, augmenting software's analytic reach—it becomes instead a make-or-break User Experience necessity.

Conversely, if we assume that hub applications will adopt the “look and feel” of native desktop applications, then they can present more holistic information—taking advantage of larger screens, with secondary application windows and other interactive features that we associate with native GUI components. It is a reasonable hypothesis that this renders AI less important: the more data that can be shown, the less need for software to filter information on users’ behalf. As Teixeira et al. put it, “some authors argue that the number of interactions between users and the smart home must be kept to a minimum,” but “to remove obstacles in the adoption of smart home systems [...] preserving the autonomy of the user may seem like the most sensible course of action.” In that spirit, investments in AI solutions might be redirected to HCI, securitization, data transparency, and other software virtues which customers may value more than “smart” software that actually strips them of control.

Hub applications could therefore exemplify what Teixeira et al. call “user-centric” design. In this guise, hubs have at least three key responsibilities:

1. To present device and system data for human users, in graphical, interactive formats suitable for people to oversee the system and intervene as needed.
2. To validate device and system data, ensuring that the system is behaving correctly and predictably.
3. To log data (in whole or in part) for subsequent analysis and maintenance.

Once software receives device data, it needs to marshal the information between different formats, exposing data in the different contexts of GUI components, database storage, and analytic review, to confirm proper operation of devices and their handler code.

The more rigorously that engineers understand and document the morphology of information across these different software roles, the more clearly we can define protocols for software design and user expectations. Careful design requires answering many technical questions: how should the application respond if it encounters unexpected data? How, in the presence of erroneous data, can we distinguish device malfunction from coding error? How should users and/or support staff be notified of errors? What is the optimal Interface Design for users to identify anomalies, or identify situations needing human intervention, and then be able to perform the necessary actions via software? What kind of database should hold system data retroactively, and what kind of queries or analyses should engineers be able to perform so as to study system data, to access the system's past states and performance?

Because Cyber-Physical devices are intrinsically *networked*—whether over special wireless networks or the World Wide Web—there is an enlarged “surface area” for vulnerability. Moreover, because they are often worn by people or used in a domestic setting, they tend to carry personal (e.g., location) information, making network security protocols especially important [2–7]. In brief, the dangers of coding errors and software vulnerabilities, in Cyber-Physical Systems like the Internet of Things (IoT), are even more pronounced than in other application domains. While it is unfortunate if a software crash causes someone to lose data, it would be even more serious if a Cyber-Physical “dashboard” application were to malfunction and leave physical, networked devices in a dangerous state.

It is helpful at this point to distinguish *cyber security* from *safety*. When these concepts are separated, *security* generally refers to preventing *deliberate, malicious* intrusion into Cyber-Physical networks. Cyber *safety* refers to preventing unintended or dangerous system behavior due to innocent human error, physical malfunction, or incorrect programming. Malicious attacks—in particular the risks of “cyber warfare”—are prominent in the public imagination, but innocent coding errors or design flaws are equally dangerous. Incorrect data readings, for example, led to recent Boeing 737 MAX jet accidents causing over 300 fatalities (plus the worldwide grounding of that airplane model and billions of dollars in losses for the company). Software failures in either runtime maintenance or anticipatory risk-assessment have been identified as contributing factors to high-profile accidents like Chernobyl [8] and the Fukushima nuclear reactor meltdown [9]. A less tragic but noteworthy case was the 1999 crash of NASA’s US\$125 million Mars Climate Orbiter. This crash was caused by software malfunctions which in turn were due to intercommunicating software components producing incompatible data—in particular, employing incompatible scales of measurement (resulting in an unanticipated mixture of imperial and metric units). In general, it is reasonable to assume that coding errors are among the deadliest and costliest sources of man-made injury and property damage.

Given the risks of undetected data corruption, seemingly mundane questions about how Cyber-Physical applications verify data—and respond to apparent anomalies—become essential aspects of planning and development. Consider even a simple data aggregate like blood pressure (combining systolic and diastolic measurements). Empirically, systolic pressure is always greater than diastolic. Software systems need commensurately to agree on a protocol for encoding the numbers to ensure that they are in the correct order, and that they represent biologically plausible measurements. How should a particular software component test that received blood pressure data is accurate? Should it always test that the systolic quantity is indeed greater than the diastolic, and that both numbers fall in medically possible ranges? How should the component report data that fail this test? If such data checking is not performed—on the premise that the data will be proofed elsewhere—then how can this assumption be justified?

In general, how can engineers identify, in a large and complex software system, all the points where data is subject to validation tests, and then by modeling the overall system in terms of these checkpoints ensure that all needed verifications are performed at least one time? Continuing the blood-pressure example, how would a software procedure that *does* check the integrity of the systolic/diastolic pair indicate for the overall system model that it performs that particular verification? Conversely, how would a procedure that does *not* perform that verification indicate that this verification must be performed elsewhere in the system, to guarantee that the procedure's assumptions are satisfied?

These questions are important not only for objective, measurable assessments of software quality, but also for people's more subjective trust in the reliability of software systems. In the modern world, we allow software to be a determining factor in systems' behavior, in places where malfunction can be fatal—airplanes, hospitals, electricity grids, trains carrying toxic chemicals, highways and city streets, etc. Consider the model of “Ubiquitous Computing” pertinent to the series to which this volume (and hence this chapter) belongs. As explained in the series introduction^a:

U-healthcare systems [...] will allow physicians to remotely diagnose, access, and monitor critical patient's symptoms and will enable real time communication with patients. [This] series will contain systems based on the four future ubiquitous sensing for healthcare (USH) principles, namely i) proactiveness, where healthcare data transmission to healthcare providers has to be done proactively to enable necessary interventions, ii) transparency, where the healthcare monitoring system design should transparent, iii) awareness, where monitors and devices should be tuned to the context of the wearer, and iv) trustworthiness, where the personal health data transmission over a wireless medium requires security, control and authorize access.

^aSee <https://sites.google.com/view/series-title-ausah/home?authuser=0>.

Observe that in this scenario, patients will have to place a level of trust in Ubiquitous Health technology comparable to the trust that they place in human doctors and other health professionals.

All of this should cause software engineers and developers to take notice. Modern society places trust in doctors for well-rehearsed and legally scrutinized reasons: physicians have to prove their competence before being allowed to practice medicine, and this right can be revoked due to malpractice. Treatment and diagnostic clinics need to be licensed, and pharmaceuticals (as well as medical equipment) are subject to rigorous testing and scientific investigation before being marketable. Notwithstanding “free market” ideologies, governments are aggressively involved in caretaking their healthcare systems; commercial activities (like marketing) are regulated, and operational transparency (like reporting adverse outcomes) is mandated, more so than in most other sectors of the economy. This level of oversight *causes* the public to trust that clinicians’ recommendations are usually correct, or that medicines are usually beneficial more than harmful.

The problem, as software becomes an increasingly central feature of the biomedical ecosystem, is that no commensurate oversight framework exists in the software world. Biomedical IT regulations tend to be ad hoc and narrowly domain-focused. For example, code bases in the United States that manage HL-7 data (a standardized EMR—Electronic Medical Record—format) must meet certain requirements, but there is no comparable framework for software targeting other kinds of healthcare information. This is not only—or not primarily—an issue of lax government oversight. The deeper problem is that we do not have a clear picture, in the framework of computer programming and software development, of what a robust regulatory framework would look like: what kinds of questions it would ask; what steps a company could follow to demonstrate regulatory compliance; and what indicators the public should consult to verify that any software that could affect their medical outcome has been properly vetted.

In the United States, HL-7 (which encompasses multiple clinical, continuity-of-care, and decision-support formats) is a good example of how technology can be backed with robust certification and Quality Assurance layers (the “7” refers to the seventh, or application-specific, layer of a technology stack which collectively enables network-based digital sharing of health records and other bioinformatic content). Government-supported programs such as ONC Health IT Certification (ONC stands for “Office of the National Coordinator for Health Information Technology”), which maintains a network of software-evaluation labs and testing infrastructure to vet Health IT products (called “modules”), confirm in particular that software claiming HL-7 compliance can properly *consume* and *query* information presented to the system via document formats or data structures included in the HL-7 family of standards. This gives developers clear guidelines of what behaviors their applications must emulate in order to participate legitimately in biomedical data sharing, and also allows doctors and patients to garner more information about Health IT vendors’ solutions.

The limitation of HL-7 is, however, that its robust evaluation methodology only applies to the collection of data formats standardized in the HL-7 family. Consequently, any data generated by new treatments or technologies—whether innovations in clinical methods, Cyber-Physical medical devices, pharmaceuticals, computational methodology, or any other research—will remain outside the certification pipeline unless the new kind of generated data can be translated into one of the HL-7 formats. Some new research may be amenable to conventional EMR standards: clinical trials of a new medication, for example, may fit comfortably in existing reporting formats because the desired outcomes—lowering blood pressure, lowering viral load, shrinking tumors, and so forth—can be measured and expressed via standard records. However, in many cases the whole premise of clinical innovations is to rethink analytic methodologies; or clinical technology is introduced which generates new kind of data (reflecting the novel physical design of diagnostic equipment, for example) that needs to be included in medical records according to its own, internal data model.

As a concrete example, consider Eran Bellin's clinical looking glass (CLG) model, developed at Montefiore Medical Center in New York City [10]. Inspired by “patient-centered” care trends, CLG seeks to organize clinical data around “patient-centric time frames” and the structural representation of patients’ trajectory through a medical system. Diagnoses, for example, are then recorded as diagnostic *events* (to use an example from CLG’s documentation, diabetes may be recorded as the event of a lab test reporting HemoglobinA1C above 9.5), and subsequent time-points and time-spans are defined as offsets from such “index events.” The clinical rationale for CLG data models is to facilitate “patient-centered” studies of medical data—analyzing, for instance, the length of time between a patient starting a treatment course and the point at which a desired target outcome is achieved; Bellin argues that conventional clinical software is designed around “business-volume-centric” queries which originate from hospitals’ administrative requirements rather than patient-centered concerns, which makes it more difficult to query normal clinical databases in a patient-centered manner [11, pp. 5–6]. The CLG software counters that trend by developing a “patient-centered” query methodology (and a GUI for doctors to formulate queries graphically), along with a higher-scale “cohort model” where patients are aggregated (while the patient-centered data models are preserved at the cohort level). Internally, this query system depends upon an “object model” whose diagnostic, temporal, and demographic architecture differs from conventional Health IT representations. Moreover, the CLG application can pool databases from multiple sources, so that data sharing and integration is an intrinsic CLG feature.

Currently CLG is embodied in a single proprietary application, but if the CLG query framework is appreciated by doctors as noticeably more effective, it is easy to imagine that the CLG object model will be generalized from its specific application context and repositioned as an object-system which diverse software components could adopt. In this scenario, given that CLG is expressly formulated as an alternative to traditional EMRs, it is reasonable to assume that the CLG object model would engender its own “consume and query” protocols—for data sharing, validation,

and analysis—which, consequently, would be too divergent from HL-7 formats for the HL-7 certification process to be applicable. In that instance, comparable rigor would thus demand either a separate certification initiative specific to CLG or an informal (and therefore unregulated) commitment on the part of developers to comply with behavioral standards appropriate for the CLG object model.

The case of CLG is just one example of clinical innovation; given the (thankfully) fertile ground for research on new treatments and technologies, we can see our healthcare system as targets for many analogous innovations from the biochemical, bioinformatics, and Cyber-Physical domains. The sheer volume of new data models makes it likely that “data-centric” certification as in HL-7—standardizing a predefined set of data models—will continually become outdated. For this reason, Requirements Engineering should look more toward a “software-centric” paradigm, where developers consciously adopt design patterns abstracted from particular data models.^b

Outside the medical arena, similar analyses—of the need for robust but dynamically evolving vetting of new technologies—could be made regarding software in Cyber-Physical settings like transportation, energy (power generation and electrical grids), physical infrastructure, environmental protections, government and civic data, and so forth—settings where software errors threaten personal and/or property damages. The public has a relatively inchoate idea of issues related to cyber safety, security, and privacy: we (collectively) have an informal impression that current technology is failing to meet the public’s desired standards, but there is no clear picture of what IT engineers can or should do to improve the technology going forward.

Regulatory oversight is only effective in proportion to scientific clarity vis-à-vis desired outcomes. Drugs and treatment protocols, for instance, can be evaluated through “gold standard” double-blind clinical trials—alongside statistical models, like “five-sigma” criteria, which measure scientists’ confidence that trial results are truly predictive, rather than results of random chance. This package of scientific methodology provides a framework which can then be adopted in legal or legislative contexts. With respect to medications, policy makers can stipulate that pharmaceuticals should be tested in double-blind trials, with statistically verifiable positive results, before being approved for general-purpose clinical use. Such a well-defined policy approach is *only possible* because there are biomedical paradigms which define how treatments can be tested to maximize the chance that positive test results predict similar results for the general patient population.

Analogously, a general theory of cyber safety should be a software-design issue before it becomes a policy or contractual issue. Software engineering and programming language design need their own evaluative guidelines, their own analogs to double-blind trials and five-sigma confidence. It is at the region of low-level

^bAs a case study, the dataset for this chapter includes an object model with some similarities to CLG; the documentation and protocols associated with query evaluators for that model demonstrate procedural-level Requirements Engineering using techniques that can be applied across many data profiles—shifting the weight of certification, or code evaluation, away from the specific data model and toward procedural-level implementational maxims.

software design—of actual source code in its local implementation and holistic integration—that engineers can develop technical “best practices” which then provide substance to regulative oversight. Stakeholders or governments can recommend (or require) that certain practices adopted, but only if engineers identify coding standards which are believed, on firm theoretical grounds, to effectuate safer, more robust software.

3.1.1 Gatekeeper code

There are several design principles which can help ensure safety in large-scale, native/desktop-style GUI -based applications. These include the following:

1. Identify operational relationships between types. Suppose D is a data structure modeled via type t . This type can then be associated with a type (say, t') of GUI components which visually display values of that type t . A simple data structure may have GUI representation via small “widgets” embedded in other components (consider a thermometer icon to display temperature). Conversely, if D has many component parts, its corresponding GUI type may need to span its own application window, with a collection of nested textual or graphical elements. There may also be a type (say, t'') representing t -values in a format suitable for database persistence. Application code should explicitly indicate these sorts of inter-type relationships.
2. Identify coding assumptions which determine the validity of typed values and/or function calls. For each application-specific data type, consider whether every computationally possible instance of that type is actually meaningful for the real-world domain which the type represents. For instance, a type representing blood pressure has a subset of values which are biologically meaningful—where systolic pressure is greater than diastolic and where both numbers are in a sensible range. Likewise, for every procedure defined on application-specific data types, consider whether the procedure might receive arguments that are computationally feasible but empirically nonsensical. Then, establish a protocol for acting upon erroneous data values or procedure parameters. How should the error be handled, without disrupting the overall application?
3. Identify points in the code base that represent new data being introduced into the application, and code that can materially affect the “outside world.” Most of the code behind GUI software will manage data being transferred between different parts of the system, internally. However, there will be specific implementations for receiving new data from external sources or signals. These are places where data “enters the system.”^c Conversely, other code points localize the software’s capabilities to initiate external effects.^d The functions that leverage such

^cA simple example is, for desktop applications, the preliminary code that runs when users click a mouse button. In the Cyber-Physical context, an example might be code that is activated when motion-detector sensors signal something moving in their vicinity.

^dFor instance, one consequence of users clicking a mouse button might be that the on-screen cursor changes shape. Alternatively, motion detection might trigger lights to be turned on. In these cases, the software is hooked up to external devices that have tangible capabilities, such as activating a light-source or modifying the on-screen cursor.

capabilities reveal data “leaving the system.” Distinguishing where data “enters” and “leaves” the system from where data is transferred *inside* the application helps ensure that incoming data and external effects are properly vetted.^e

The methods I propose in this chapter are applicable to each of these concerns, but for purposes of exposition I will focus on the second issue: testing type instances and procedure parameters for fine-grained specifications (more precise than strong typing alone).

Strongly typed programming languages offer some guarantees on types and procedures: a function that takes an **int** will never be called on a value that is *not* an integer (e.g., the character-string “**46**” instead of the *number* 46). Likewise, a type where one field is an **int** (representing someone’s age, say) will never be instantiated with something *other than* an **int** in that field. Such minimal guarantees, however, are too coarse for safety-conscious programming. Even the smallest **unsigned int** type (8-bit) would permit someone’s age to be 255 years, which is surely an error. So any safety-conscious code dealing with ages needs to check that the numbers fall in a range narrower than built-in types allow on their own, or to ensure that such checks are performed ahead of time.

The central technical challenge of safety-conscious coding is therefore to *extend* or *complement* each programming language’s built-in type system so as to represent more fine-grained assumptions and specifications. While individual tests may seem straightforward on a local level, a consistent data-verification architecture—how this coding dimension integrates with the totality of software features and responsibility—can be much more complicated. Developers need to consider several overarching questions, such as the following:

- Should data validation be included in the same procedures which operate on (validated) data, or should validation be factored into separate procedures?
- Should data validation be implemented at the type level or the procedural level? That is, should specialized data types be implemented that are guaranteed only to hold valid data? Or should procedures work with more generic data types, and perform validations on a case-by-case basis?
- How should incorrect data be handled? In Cyber-Physical software, there may be no obvious way to abort an operation in the presence of corrupt data. Terminating the application may not be an option; silently canceling the desired operation or trying to substitute “correct” or “default” data may be unwise; and presenting technical error messages to human users may be confusing.

^eSeveral mathematical frameworks have been developed to codify the intuition of software components as “systems” with external data sources and effects, extending the model of software as self-contained information spaces: notably, Functional-Reactive Programming [12–14] and (a little more indirectly) the theory of Hypergraph Categories [15–18].

These questions do not have simple answers. As such, we should develop a rigorous theoretical framework so as to codify the various options involved—what architectural decisions can be made, and what the strengths and weaknesses of different solutions are.

I will use the term *gatekeeper code* for any code that checks programming assumptions more fine-grained than strong typing alone allows—for example, that someone’s age is not reported as 255 years, or that systolic pressure is not recorded as less than diastolic. I will use the term *fragile code* for code that *makes* programming assumptions *without itself* verifying that such assumptions are obeyed. Fragile code is especially consequential when incorrect data would cause the code to fail significantly—to crash the application, enter an infinite loop, or any other nonrecoverable scenario.

Note that “fragile” is not a term of criticism—some algorithms simply work on a restricted space of values, and it is inevitable that code implementing such algorithms will only behave properly when provided with values having the requisite properties. It is necessary to ensure that such algorithms are *only* called with correct data. But insofar as testing of the data lies outside the algorithms themselves, the proper validation has to occur *before* the algorithms commence. In short, *fragile* and *gatekeeper* code often has to be paired off: for each segment of fragile code that *makes* assumptions, there should be a corresponding segment of gatekeeper code that *checks* those assumptions.

In that general outline, however, there is room for a variety of coding styles and paradigms. Perhaps these can be broadly classified into three groups:

1. Combine gatekeeper and fragile code in one procedure.
2. Separate gatekeeper and fragile code into different procedures.
3. Implement narrower types so that gatekeeper code is called when types are first instantiated.

Consider a function that calculates the difference between systolic and diastolic blood pressure, returning an unsigned integer. If this code were called with malformed data wherein systolic and diastolic were inverted, the difference would be a negative number, which (under binary conversion to an unsigned integer) would come out as a potentially extremely large positive number (as if the patient had blood pressure in, say, the tens of thousands). This nonsensical outcome indicates that the basic calculation is fragile. We then have three options: test “systolic-greater-than diastolic” *within the procedure*; require that this test be performed prior to the procedure being called; or use a special data structure configured such that systolic-over-diastolic can be confirmed as soon as any blood pressure value is constructed in the system.

There are strengths and weaknesses of each option. Checking parameters at the start of a procedure makes code more complex and harder to maintain, and also makes updating the code more difficult. The blood pressure case is a simple example, but in real situations there may be more complex data-validation requirements, and

separating code that *checks* data from code that *uses* data, into different procedures, may simplify subsequent code maintenance. If the *validation* code needs to be modified—and if it is factored into its own procedure—this can be done without modifying the code that actually works on the data (reducing the risk of new coding errors).

Also, engineers may appreciate the flexibility to upgrade how improper data is handled *throughout the application*. Suppose it is decided to log, and periodically review, all instances of malformed parameters “rejected” by gatekeeper code. Every **if...then...else** block could potentially then need to be paired with logging code, and it would be time-consuming and error-prone to verify that such protocol is obeyed everywhere, throughout a large, complex code base. Isolating gatekeeper procedures, and labeling them as such, would make it easier both to find all gatekeeping logic and to modify the protocol for handling failed validations.

In essence, factoring *gatekeeper* and *fragile* code into separate procedures exemplifies the programming maxim of “separation of concerns”: it makes the overall system more flexible and easier to maintain. However, such separation creates a new problem—ensuring that the gatekeeping procedure is always called. Meanwhile, using special-purpose, narrowed data types add complexity to the overall project if these data types are unique to that one code base, and therefore incommensurate with data provided by external sources. In these situations, the software must transform data between more generic and more specific representations before sharing it (as sender or receiver), which makes the code more complicated.

Because there is no one best “gatekeeping protocol,” these issues should be studied holistically, defining a range of options that can be weighed at the planning and prototyping stages—well before most of the serious production code is implemented. Documentation that such preparatory considerations have been worked through can then help external monitors certify that the eventual code base was developed rigorously and carefully.

In the specific Cyber-Physical context, gatekeeping is especially important when working with device data. Such data are almost always constrained by the physical construction of devices and the kinds of physical quantities they measure (if they are sensors) or their physical capabilities (if they are “actuators,” devices that cause changes in their environments). For sensors, it is an empirical question what range of values can be expected assuming proper functioning (and therefore what validations can check that the instrument is working as intended). For actuators, it should be similarly understood what range of values guarantee safe, correct behavior. For any device, then, we can construct a *profile*—an abstract, mathematical picture of the space of “normal” values associated with proper device performance. Gatekeeping code may thereby ensure that data received from or sent to devices fits within the profile. Defining device profiles, and explicitly notating the corresponding gatekeeping code, should accordingly be an essential pre-implementation planning step for Cyber-Physical software hubs.

3.1.2 Fragile code

Fragile code is code that makes assumptions stronger than the programming language on its own can guarantee. Where safety and quality are priorities, fragile code needs gatekeeping code to ensure that these assumptions are warranted.

Fragile code is not necessarily a harbinger of poor design. Sometimes implementations can be optimized for special circumstances, and optimizations are valuable and should be used wherever possible. Consider an optimized algorithm that works with two lists that must be the same size. Such an algorithm should be preferred over a less efficient one when possible—which is to say, whenever dealing with two lists that are indeed the same size. Suppose this algorithm is included in an open-source library intended to be shared among many different projects. The library’s engineer might, quite reasonably, deliberately choose not to check that the algorithm is invoked on same-sized lists—checks that would complicate the code, and sometimes slow the algorithm unnecessarily. It is then the responsibility of code that *calls* whatever procedure implements the algorithm to ensure that it is being employed correctly—specifically, that this “client” code does *not* try to use the algorithm with *different-sized* lists. Here “fragility” is probably well-motivated: accepting that algorithms are sometimes implemented in fragile code can make the code cleaner and its intentions clearer, and permits the algorithms being optimized for speed.

The opposite of fragile code is sometimes called “robust” code. While robustness is desirable in principle, code that simplistically avoids fragility may be harder to maintain than deliberately fragile but carefully documented code. Robust code often has to check for many conditions to ensure that it is being used properly, which can make the code harder to maintain and understand. The hypothetical algorithm that I contemplated in the previous paragraph could be made robust by *checking* (rather than just *assuming*) that it is invoked with same-sized lists. But if it has other requirements—that the lists are non-empty, and so forth—the implementation can get padded with a chain of preliminary gatekeeper code. In such cases, the gatekeeper code may be better factored into a different procedure, or expressed as a specification that engineers must study before attempting to use the implementation itself.

Such transparent declaration of coding assumptions and specifications can inspire developers using the code to proceed attentively, which can be safer in the long run than trying to avoid fragile code through engineering alone. The takeaway is that while “robust” is contrasted with “fragile” at the smallest scales (such as a single procedure), the overall goal is systems and components that are robust at the largest scale—which often means accepting *locally* fragile code. Architecturally, the ideal design may combine individual, *locally fragile* units with rigorous documentation and gatekeeping so that the *totality* is robust. Defining and declaring specifications is then an intrinsic part of implementing code bases which are both robust and maintainable.

Unfortunately, specifications are often created only as human-readable documents, which might have a semi-formal structure but are not actually machine-readable. There is then a disconnect between features *in the code itself* that promote

robustness, and specifications intended for *human* readers—developers and engineers. The code-level and human-level features promoting robustness will tend to overlap partially but not completely, demanding a complex evaluation of where gatekeeping code is needed and how to double-check via unit tests and other post-implementation examinations. This is the kind of situation—an impasse, or partial but incomplete overlap, between formal and semi-formal specifications—that many programmers hope to avoid via strong/expressive type systems.

Advanced type-theoretic constructs—including Dependent Types, typestate, and effect-systems—model requirements with more precision than can be achieved via conventional type systems alone. Integrating these paradigms into core-language type systems permits data validation to be integrated with general-purpose type checking, without the need for static analyzers or other “third-party” tools (i.e., projects maintained orthogonally to the actual language—that is, to languages’ compilers and runtimes). This means that Requirements Specifications that would otherwise be *human* documents—communiqués between developers, but only at best partially enforced by programming languages internally—get formalized to the point where compilers (for example) evaluate requirements without human intervention. Unfortunately, these advanced type systems are also more complex to implement. If Software Language Engineers aspire to make Dependent Types and similar advanced constructs part of their core language, creating compilers and runtime engines for these languages becomes proportionately more difficult.

Programming languages are, at one level, artificial *languages*—they allow humans to communicate algorithms and procedures to computer processors, and to one another. But programming languages are also themselves engineering artifacts. It is a complex project to transform textual source-code—which is human-readable and looks a little bit like natural language—into binary instructions that computers can execute. For each language, there is a stack of tools—parsers, compilers, and/or runtime libraries—which enable source code to be executed according to the language specifications. Language design is therefore constrained by what is technically feasible for these supporting tools. Practical language design, then, is an interdisciplinary process which needs to consider both the dimension of programming languages as communicative media and as digital artifacts with their own engineering challenges and limitations.

These limitations then produce a split between tools *in the language itself* and those maintained as separate projects *analyzing* code in a given language. They raise the question, which has no simple answer, of what should be guaranteed by *the language* and what should be tested externally. I will now examine this question in greater detail.

3.1.3 Core language vs. external tools

Because of programming languages’ engineering limitations, such as I just outlined, software projects should not necessarily rely on core-language features for responsible, safety-conscious programming. In short, methodologies for safety-conscious

coding can be split between those that depend on core-language features, and those that rely on external, retroactive analysis of sensitive code. On the one hand, some languages and projects prioritize specifications that are intrinsic to the language and integrate seamlessly and operationally into the language’s foundational compile-and-run sequence. Improper code (relative to specifications) should not compile, or, as a last resort, should fail gracefully at run-time. Moreover, in terms of programmers’ thought processes, the description of specifications should be intellectually continuous with other cognitive processes involved in composing code, such as designing types or implementing algorithms. For the sake of discussion, I will call this paradigm “internalism.”

The “internalist” mindset seeks to integrate data validation seamlessly with other language features. Malformed data should be flagged via similar mechanisms as code which fails to type-check, and errors should be detected as early in the development process as possible. Such a mindset is evident in passages like this (describing the Ivory programming language):

Ivory’s type system is shallowly embedded within Haskell’s type system, taking advantage of the extensions provided by [the Glasgow Haskell Compiler]. Thus, well-typed Ivory programs are guaranteed to produce memory safe executables, *all without writing a stand-alone type-checker* [my emphasis]. In contrast, the Ivory syntax is *deeply* embedded within Haskell. This novel combination of shallowly-embedded types and deeply-embedded syntax permits ease of development without sacrificing the ability to develop various back-ends and verification tools [such as] a theorem-prover back-end. All these back-ends share the same AST [Abstract Syntax Tree]: Ivory verifies what it compiles. (Elliott et al. [19, p. 1])

In other words, the creators of Ivory are promoting the fact that their language buttresses via its type system—and via a mathematical precision suitable for proof engines—code guarantees that for most languages require external analysis tools.

Contrary to this “internalist” philosophy, other approaches (perhaps best described as “externalist”) favor a neater separation of specification, declaration, and testing from the core language. In particular—according to the “externalist” mind-set—most of the more important or complex safety-checking does not natively integrate with the underlying language, but instead requires either an external source code analyzer, or regulatory runtime libraries, or some combination of the two. Moreover, it is unrealistic to expect all programming errors to be avoided with enough proactive planning, expressive typing, and safety-focused paradigms: any complex code base requires some retroactive design, some combination of unit-testing and mechanisms (including those third party to both the language and the projects whose code is implemented in the language) for externally analyzing, observing, and higher-scale testing for the code, plus post-deployment monitoring.

As a counterpoint to the features cited as benefits to the Ivory language, which I identified as representing the “internalist” paradigm, consider Santanu Paul’s Source Code Algebra (SCA) system described in [20–22]:

Source code files are processed using tools such as parsers, static analyzers, etc. and the necessary information (according to the SCA data model) is stored in a repository. A user interacts with the system, in principle, through a variety of high-level languages, or by specifying SCA expressions directly. Queries are mapped to SCA expressions, the SCA optimizer tries to simplify the expressions, and finally, the SCA evaluator evaluates the expression and returns the results to the user.

We expect that many source code queries will be expressed using high-level query languages or invoked through graphical user interfaces. High-level queries in the appropriate form (e.g., graphical, command-line, relational, or pattern-based) will be translated into equivalent SCA expressions. An SCA expression can then be evaluated using a standard SCA evaluator, which will serve as a common query processing engine. The analogy from relational database systems is the translation of SQL to expressions based on relational algebra. (Paul and Prakash [20, pp. 25–26])

So the *algebraic* representation of source code is favored here because it makes computer code available as a data structure that can be processed via *external* technologies, like “high-level languages,” query languages, and graphical tools. The vision of an optimal development environment guiding this kind of project is opposite, or at least complementary, to a project like Ivory: the whole point of Source Code Algebra is to pull code verification—the analysis of code to build trust in its safety and robustness—*outside* the language itself and into the surrounding Development Environment ecosystem.

These philosophical differences (what I dub “internalist” vs. “externalist”) are normative as well as descriptive: they influence programming language design, and how languages in turn influence coding practices. One goal of language design is to produce languages that offer rigorous guarantees—fine-tuning their type system and compilation model to maximize the level of detail guaranteed for any code that type-checks and compiles. Another goal of language design is to define syntax and semantics permitting valid source code to be analyzed as a data structure in its own right. Ideally, languages can aspire to both goals. In practice, however, achieving both equally can be technically difficult. The internal representations conducive to strong type and compiler guarantees are not necessarily amenable to convenient source-level analysis, and vice versa.

Language engineers, then, have to work with two rather different constituencies. One community of programmers tends to prefer that specification and validation be integral to/integrated with the language’s type system and compile-run cycle (and standard runtime environment), whereas a different community prefers to treat code evaluation as a distinct part of the development process, something logically, operationally, and cognitively separate from hand-to-screen codewriting (and may chafe at languages restricting certain code constructs because they can theoretically produce coding errors, even when the anomalies involved are trivial enough to be tractable for even barely adequate code review). One challenge for language engineers is accordingly to serve both communities. We can, for example, aspire to implement type systems which are sufficiently expressive to model many specification, validation, and gate-keeping scenarios, while also anticipating that language code should be syntactically and semantic designed to be useful in the context of external tools (like static analyzers) and models (like Source Code Algebras and Source Code Ontologies).

The techniques I discuss here work toward these goals on two levels. First, I propose a general-purpose representation of computer code in terms of Directed Hypergraphs, sufficiently rigorous to codify a theory of functional types as types whose values are (potentially) initialized from formal representations of source code—which is to say, in the present context, code graphs. Next, I analyze different kinds of “lambda abstraction”—the idea of converting closed expressions to open-ended formulae by asserting that some symbols are “input parameters” rather than fixed values, as in Lambda Calculus—from the perspective of axioms regulating how inputs and outputs may be passed to and obtained from computational procedures. I bridge these topics—Hypergraphs and Generalized Lambda Calculi—by taking abstraction as a feature of code graphs wherein some hypernodes are singled out as procedural “inputs” or “outputs.” The basic form of this model—combining what are essentially two otherwise unrelated mathematical formations, Directed Hypergraphs and (typed) Lambda Calculus—is laid out in Sections 3.3 and 3.4. I then engage a more rigorous study of code-graph hypernodes as “carriers” of runtime values, some of which collectively form “channels” concerning values which vary at runtime between different executions of a function body. Carriers and channels piece together to form “Channel Groups” that describe structures with meaning both within source code as an organized system (at “compile time” and during static code analysis) and at runtime. Channel Groups have four different semantic interpretations, varying via the distinctions between runtime and compile-time and between *expressions* and (function) *signatures*. I use the framework of Channel Groups to identify design patterns that achieve many goals of “expressive” type systems while being implementationally feasible given the constraints of mainstream programming languages and compilers.

Prior to launching into that mostly theoretical discussion, however, I will examine real-world Cyber-Physical data with a little more specificity. My goal in the following pages is to describe the sorts of details that need to be expressed in Cyber-Physical data models and therefore verified in Cyber-Physical code. Whereas my subsequent analyses of code representation will address *how* code can demonstrate its underlying data model, my preliminary discussion will motivate *why* code needs to do so in the first place.

3.2 Case studies

To motivate the themes I will emphasize going forward, this section will examine some concrete data models that are used or proposed in various Cyber-Physical contexts. I hope this discussion will lay out parameters on device behavior or shared data to illustrate typical modeling patterns and their corresponding safety or validation requirements. As an initial overview, the following are some examples of data profiles that might be wedged to deployed Cyber-Physical devices (my comments here are also summarized in Table 3.1):

Table 3.1 Example data profiles.

Data type	Dimensions/fields	Requirements
Blood pressure	Beats per minute; resting/active (boolean) or exercise level (scalar)	Systolic more than diastolic; plausible resting/active range
Accelerator	Incline; Movement in two or three dimensions	Biaxial or triaxial
Audio sample	Binary data/file and/or (waveform analysis) feature set	Length (in time) (1D)
Audio feature	Time-interval inside sample and/or spectral numerics	Subinterval of sample (1D)
Image	Matrix of values in a color model: RGB, RGBa, HSV, etc.	Matrix dimensions (2D)
Image segment	Rectangular coordinates; geometric characterization (e.g., ellipse dimensions); scales of resolution where detectable	Subregion of image (2D)
Bioacoustic sample	Audio sample plus species identifier; geospatial and timestamp coordinates	Well-formed spacetime coordinates
Speech sample	Audio sample plus text transcription; spacetime coordinates; identify language/dialect and/or speaker	Valid dialect and/or speaker identifier
Dependency-parsed text	Text transcription plus parse serialization; audio metadata	Valid and accessible metadata
Lexical text component	Index into parse serialization; part of speech tag; lexical/semantic classification	Valid index

Heart-rate monitor A heart-rate sensor generates continuously sampled integer values whose understood Dimension of Measurement is in “beats per minute” and whose maximum sensible range (inclusive of both rest and exercise) corresponds roughly to the [40–200] interval. Interpreting heart-rate data depends on whether the person is resting or exercising. Therefore, a usable data structure might join a beats-per-minute dimension with a field indicating (or measuring) exertion, either a two-valued discrimination between “rest” and “exercise” or a more granular sampling of a person’s movement cotemporous with the heart-rate calculations.

Accelerometers These devices measure objects’ or people’s rate of movement (see [23–26], etc.), and therefore can be paired with heart-rate sensors to quantify how heart rate is affected by exercise (likewise for other biometric instruments, such as those calculating respiration rate). Outside the biomedical context, accelerometers are important for Smart Cities (or factories, and so forth) for modeling the integrity of buildings, bridges, and industrial areas or structures (see, e.g., [27, 28]).

An accelerometer presents data as voltage changes in two or three directional axes, data which may only produce signals when a change occurs (and therefore

is not continuously varying), and which is mathematically converted to yield information about physical objects’ (including a person’s) movement and incline. Mechanically, that is, accelerometers actually measure *voltage*, from which quantitative reports of movement and incline can be derived. Accelerometers are classified as *biaxial* or *triaxial* depending on whether they sample forces in two or three spatial dimensions.

The pairwise combination of heart-rate and acceleration data (common in wearable devices) is then a mixture of these two measurement profiles—partly continuous and partly discrete sampling, with variegated axes and interdimensional relationships.

Remote medical diagnosis An emerging application of Cyber-Physical technology involves medical equipment deployed outside conventional clinical settings—in remote areas with little electricity, refugee camps, temporary ad hoc medical units (established to contain potential epidemics, for instance), and so forth. These settings have limited diagnostic capabilities, so data are often transmitted to distant locations in lieu of on-site laboratories.

A good case study derives from “medical whole slide imaging” (mWSI) [29], where a cell phone attached to an ordinary microscope, by subtle modifications of camera position and microscope resolution, allows many views to be made on one slide.^f Positional data (the configuration of the phone and microscope) then merges with image segmentation computations characteristic of conventional whole slide imaging (see, e.g., [30]), and by extension diagnostic pathology in general, which is concerned with isolating medically significant image features and identifying diagnostically indicative anomalies (such as cell shapes suggesting cancer).

Segmentation, in turn, generates multiple forms of geometric data: in [31], for instance, segments are identified as approximations to ellipse shapes, and features are tracked across scales of resolution, so geometric data merges ellipse dimensions with positional data (in the image) and a metric of feature persistence across scales. (Features that are detectable at many scales of resolution are more likely to be empirically significant rather than visual “noise”; calculating cross-scale “persistence” is an applied methodology within Statistical Topology—see, e.g., [32–34]). Merged with mWSI configuration info and patient data, the whole data package integrates geometric, Cyber-Physical, and health-record aspects.

Speech sampling Audio sensors can be used to isolate different people’s speech episodes (see Chapters 5 and 7). Feature extraction cancels background noise and partitions the foreground audio into different segments, individuated (potentially) by differences between speakers as well as each speaker’s separated conversation turns. Such data can then be employed in several ways. The two chapters just mentioned present methodology for estimating speakers’ emotional states and identifying samples’ spoken language or dialect, respectively, while the chapter

^fThe modified microscopes themselves—rigged so that mounted cameras can capture views on microscope slides and transmit them via Wi-Fi or cell phones—are an invention of AlexaPath, an NYU-incubated company that subsequently developed products in AI and other aspects of remote pathology.

by Teixeira et al. (which I quoted earlier) discusses speech-activated User Interfaces.

The data profile germane to an audio processor will be determined by the system’s overarching goals. For example, Choi and Gutierrez-Osuna [35] describe techniques for measuring emotional stress via heart-rate signals. Combined with speech-derived data, a system might accordingly be designed around emotional profiles, merging linguistic and biometric evidence. For those use-cases, programming would emphasize signs of emotional changes (reinforced by both metrics), and secondarily isolating times and locations, which factor into proper software responses to users’ moods.

On the other hand, a voice-based User Interface might similarly model speakers’ identity and location, but perform Natural Language Processing to translate speech patterns into models of user requests. Alternatively, the use-case in [Chapter 5](#) profiles speech data for language classification (namely, matching voices to the language or dialect spoken) as part of a “smart city” network. The priority here is not necessarily identifying individual speakers, or mapping vocalizations to semantic or syntactic parses (although dialect-identification is obviously a first step for those capabilities); even without further Natural Language Processing, tagged samples can yield a geospatial model of language-use in a given area.

Bioacoustic sampling Similar to speech sampling (at least up to the point where acoustical analysis gives way to syntax and semantics), audio samples can be used to track and identify species ([Chapters 8 and 9](#)). Here again feature extraction foregrounds certain noise patterns, but the main analytic objective is to map audio samples to the species of the animal that produced them. Sensor networks can then build a geospatial/temporal model of species’ distribution in the area covered by the network: which species are identified, their prevalence, their concentration in different smaller areas, and so forth. These measurements can be employed in the study of species populations and behavioral patterns, and can also add data to urban-planning or ecological models. For example, precipitous decline of a species in some location can signal environmental degradation in that vicinity.

Datasets such as those accompanying [36] (the smallest, labeled CLO-43SD, is profiled within this chapter’s dataset) provide a good overview of data generated during species identification: in addition to audio samples themselves (in WAV format), the dataset includes NPY (Numerical Python) files representing different spectral analysis methods applied to the bird songs, as well as a metadata file summarizing species-level data (such as the count of samples identified for each species). Every species also acquires a four-letter identifier then used as part of the WAV and NPY file names, which thereby themselves serve a classifying role, semantically linking each sample to its species. These three levels of information are a good example of the contrast in granularity—and the mechanisms of information acquisition—between raw Cyber-Physical input (the audio files), midstream processing (the spectral representations), and summarial

overviews (species counts and labels; other avian datasets might also recognize geospatial coordinates obtained via noting sensor placement, as a further metadata dimension).

Facial recognition Given a frontal (or, potentially, partial) view, software can match faces rather reliably to a preexisting database or track faces across different locales ([37–41], etc.). The most common methodology depends on normalizing each foreground image segment (corresponding to one face) into a rectangle, whose axes then establish vector components for any features inside the segment. Feature extraction then isolates anatomical features like eyes, nose, mouth, and chin, quantifying their position and distances, yielding a collection of numeric values which can statistically identify a person with relatively small error rates.

Given privacy concerns, enterprise or government use of this data is controversial: should analyses be performed on every person, or only on exceptional circumstances (crime investigation, say)? Can facial-recognition outcomes be anonymized so that faces would be tracked across locations but not tied to specific persons without extra (normally inaccessible) data? When and by whom should face data be obtainable, and under what legal or commercial circumstances? Should stores be allowed to use these methods to prevent shoplifting, for example? What about searching for a missing or kidnapped child, or keeping tabs on an elderly patient? When does surveillance cross the line from benevolent (protecting personal or public safety) to privacy-invasive and authoritarian?

Of course, there are many other examples of Cyber-Physical devices and capabilities that could be enumerated. But these cases illustrate certain noteworthy themes. One observation is that a gap often exists between how devices physically operate and how they are conceptualized—accelerometers, for instance, mechanically measure voltage, not acceleration or incline; but their data exposed to client software is constructed to be applied as vectors indicating persons' or objects' movement. Moreover, multiple processing steps may be needed between raw physical inputs and usable software-facing data structures. Such processing may generate a large amount of intermediate data; feature extraction from audio or image samples, say, can yield numeric aggregates with tens or hundreds of different fields. Further processing usually reduces these structures to narrower summaries: an audio sample might be consolidated to a spatial location and temporal timestamp, along with a mapping to an individual person speaking (perhaps along with a text transcription), or human language spoken, or animal species. Engineers then have to decide what level of detail to expose across a software network. Another issue is integrating data from multiple sources: most of the more futuristic scenarios envision multi-modal Ubiquitous Computing spaces where, for example, speech and biometric inputs are cross-referenced.

Different levels of data resolution also intersect with privacy concerns: simpler data structures are more likely to employ private or sensitive information as an organizing instrument, heightening security and surveillance concerns. For example, a simple facial-recognition system would match faces against known

residents of or visitors to the relevant municipalities. This is less technologically challenging than anonymized systems which would persist more mid-processing data in order to complete the algorithmic cycle—matching faces to concrete individuals—only under exceptional circumstances; of course, though, it is also a greater invasion of privacy.

Analogously, syncing speech technology with personal health data would be simplified by directly matching speaker identifications to biosensor devices wearers. Again, though, using personal identities as an anchor for disparate data points makes the overall system more vulnerable to intrusive or inappropriate use. In total, security concerns might call for more complex data structures wherein shared data excludes the more condensed summaries wherever they may expose private details, and relies more on multipart, mid-level processing structures. Rather than organize face-recognition around a database of persons, for example, the basic units might be numeric profiles paired with probabilistic links notating that a face detected at one time and place matches a face analyzed elsewhere, but without that similarity being anchored in a personal identifier.

Other broad issues raised by these Cyber-Physical case studies include (1) testing and quality assurance and (2) data interoperability. In the case of testing, many of the scenarios outlined earlier require complex computational transformations to convert raw physical data into usable software artifacts. In [42], for example, the authors present technology to measure heart rate from a distance, based on subtle analysis of physical motions associated with blood circulation and breathing. The analytic protocols leverage feature extraction from wireless signal patterns. As with feature extraction in audio and image-analysis (e.g., face recognition) settings, algorithms need to be rigorously tested to guard against false inferences or erroneous generated data. In [42], the ultimate goal is to introduce heart and breathing monitors within a Smart Home environment, with computations performed on embedded Operating Systems. However, testing and prototyping of the technology should be conducted in a desktop Operating System environment so as to generate or leverage test data, document algorithmic revisions, and in general prove the system’s trustworthiness in a controlled setting (including a software environment which transparently windows onto computational processes) before this kind of network is physically deployed.

With respect to data integration, notice how projects mentioned here often anticipate pooled or overlapping information. For instance, Smart Homes are envisioned to embed sensors analyzing speech, biomedical data points like heart rate, atmospheric measurements (temperature, say), and appliance or architectural states (windows, doors, or refrigerator doors being open, ovens or stove burners being turned on, heaters/coolers being active, and so forth). In some cases, this data would be cross-referenced, so that, for example, a voice command would close a window or turn off a stove. Analogously, Soltane et al. [43] (one of whose co-authors is also a coauthor of this volume’s chapter on bird species) describe a combination of face-recognition and speech analysis for “multi-modal biometric authentication”; here again a

component supplying image-processing data and one supplying speech metrics will need to transmit data to a hub where the two inputs can be pooled. Or, as I pointed out in the case of Mobile Whole Slide Imaging, image-segmentation, Cyber-Physical, and personal-health information fields may all be integrated into a holistic diagnostic platform.

Overall, future Cyber-Physical systems may be integrated not only with respect to their empirical domain but in term of the environs where they are deployed—Smart Homes, Smart Cities (or factories or industrial plants), hospitals and medical offices, schools and children’s activities centers, refugee or displaced-persons camps/campuses, and so forth. I will take Smart Homes as a case in point. We can imagine future homes/apartments provisioned with a panoply of devices evincing a broad spectrum of scientific backgrounds, from biology and medicine to ecology and industrial manufacturing.

3.2.1 How “Internet of Things” interoperability affects data modeling priorities

So, let us imagine the following scenario: homeowners have a choice of applications that they may install on their computers, supplied by multiple vendors or institutions, which access the myriad of Smart Home devices they have installed around the property. Cyber-Physical products are engineered to interoperate with such hub applications—and therefore with third-party components—rather than just networking with their own proprietary access points, like phone apps.

In this eventuality, technology inside the home is charged with pooling data from many kinds of devices into a comprehensive Smart Home platform, where users can see a broad overview, access disparate device data from a central location, and where cross-device data will be merged into aggregate models: for example, cross-referencing speech and biometric inputs. Devices must be designed to broadcast data to third-party software platforms, and software hubs must wrangle received data into a common format permitting integration algorithms—for example, syncing speech and biometrics—to operate properly. Received device data must therefore be systematically mapped to appropriate transform procedures and GUI components. This is important because we are no longer considering data models from the viewpoint of devices’ own capabilities (viz., their specific physical measurements and parameters). More technically, the key libraries associated with devices are no longer merely low-level drivers or IoT signal processors. Instead, the technology stack would include an intermediate semantic layer acting between “smart objects” and corresponding hub applications.

The data models at this semantic midlayer, moreover, would no longer be device-centric. Instead, they would be assessed in a software-centric milieu: how do we route device data to proper interpretive procedures? How do we consolidate device

data into GUI presentations? This means that common representational formats for wireless data, or standardized IoT Ontologies—however, much these may technically embody semantic middleware—are insufficient as *high-level, software-centric* semantic layers.

In the case of a Smart Home, once we commit to aggregating devices via a software hub, the key organizing principle is a mesh of procedures implemented in the hub application that can pool all relevant devices into one information space. What needs to be standardized then are not so much *data formats*, but in fact *data-handling procedures*.

To put it differently, device manufacturers would now be dealing with an ecosystem in which hub applications receive and aggregate their data, affording users access points to and overviews of device data and state. Hub applications may be provided by different companies and iterations, their inner workings opaque to devices themselves. What can be standardized, however, are the *procedures* implemented within hub software to receive and properly act upon device data. Software might guarantee, for example, that so long as devices are supplying signals in their documented formats, the software has capabilities to receive the signals, unpack the data, and internally represent the data in a manner suitable for device particulars. Devices can then specify what kind of internal representations are appropriate for their specific data, essentially specifying conditions on software procedures and data types.

In short, the key units of mutual trust and verification among and between Cyber-Physical devices and Cyber-Physical software are not, in theory, data structures themselves, but instead *procedures* for processing relevant data structures. Robust Cyber-Physical ecosystems can be developed by reinforcing *procedural alignment* wherever possible, including by curating substantial collections of reusable software libraries, either for direct application or as prototypes and testing tools. Procedural alignment means that disparate components become integrated by virtue of each components' anticipating the procedures which the others have available for sending, receiving, and manipulating data—an integration strategy I will dub “POSC” (Procedure-Oriented and Software-Centric) by contract to *data format* or even *Ontological* alignment.

Suppose many Cyber-Physical sensors were paired with open-source code libraries which illustrate how to process the data each device broadcasts. Commercial products could use those libraries directly, or, if they want to substitute closed-source alternatives, might be required to document that their data management emulates the open-source prototypes. Test suites and testing technology can then be implemented against the open-source libraries and reused for stress-testing analogous proprietary components. This appears to be the most likely path to ensuring interoperable, high-quality Cyber-Physical technology that serves the ultimate goal of integrated Smart Home (and Smart City, etc.) solutions.

That hypothesis notwithstanding, there are a lot more academic papers on Cyber-Physical Ontologies or common signal/message formats, like CoAP and MQTT ([44–49], etc.) than there are open-source libraries which prototype device data,

its validation, parameters, and proper transformations.^g A good case-in-point can be found in [50, p. 4 ff.] using C structures to model CoAP meta-data: while it is reasonable, even expected, for low-level driver code to be implemented in C, this code should also be the basis of data models implemented in a language like C++ where dimensions and ranges can be made explicit in the data types. Nevertheless, many engineers will instead focus on describing the more nuanced data modeling dimensions through Ontologies and other semantic specifications wholly separate from programming type systems. This has the effect of scattering the data models into different artifacts: low-level implementations with relatively little semantic expressiveness, and expressive Ontologies which, due to a lack of type-level correspondence between modeling and programming paradigms, are siloed from implementations themselves.

Conversely, in lieu of “data-centric” Ontologies whose mission is to standardize how information is mapped to a *representation*, in this chapter I will consider “Procedural” Ontologies: ones which focus on procedural capabilities and requirements that indicate whether software components are properly managing (e.g., Cyber-Physical) data. The idea is that proving procedural conformance should be a central step, and an organizing groundwork, for showing that software intended for production deployment is trustworthy and complies with technical and legal specifications.

In practical terms, the previous discussion mentioned the CLO-43SD (avian) dataset and then Chapter 5 by Vuddagiri et al., which (as noted below) builds off the AP17-OLR “challenge” corpus. I will also be referring to recent CoNLL corpora. So, together, these constitute three representative examples of datasets tangibly applicable to Cyber-Physical and/or NLP Research and Development: one audio-based (for species identification); one audio/speech; and one linguistic. Based on the idea that R&D datasets should germinate deployment data models, we should look to datasets like these to provide semantic and type-theoretic encapsulations of their information spaces and analytic methods (e.g., spectral waveform analysis,

^gThe rationale for emphasizing standard data formats is probably that these formats constrain any procedure which operates on the data, so standardization in data representation indirectly leads to standardization in data-management procedures, or what I am calling “procedural alignment.” This accommodates the fact that shared data may be used in many different software environments—components implemented in different programming languages and coding styles. However, more detailed guarantees can be engineered by grounding standardization on procedures rather than data representations themselves. To accommodate multiple programming languages and paradigms, data models can be supplemented with a “reference implementation” which prototypes proper behavior vis-à-vis conformant data; components in different languages can then emulate the prototype, serving both as an implementation guide and a criterion for other developers to accept a new implementation as trustworthy. There are several examples of a reference implementation used as a standardizing tool, analogous to an Ontology, such as the LIBRETS (Real Estate Transaction Standard) library, servers and clients for FHIR (Fast Healthcare Interoperability Resources), and clients for DICOM (Digital Imaging and Communications in Medicine), for example, for Whole Slide Imaging (see <https://www.orthanc-server.com/static.php?page=wsi>).

or Dependency Grammar parsing). Accompanying materials for this chapter provide profiles of the three aforementioned datasets (perhaps sampled, modified, or expanded somewhat), which consolidate their various files and formats into a streamlined “POSC” representational paradigm. The demo code presents examples of procedures and data types targeting these datasets, as well as an architecture for deploying datasets in a procedure-oriented and software-centric manner, in terms of how files and the information they contain are organized, and in terms of employment of data-publishing standards such as the “Research Object” model [51–55].

At present, to make these issues more concrete with further case studies, in this introductory discussion I will examine in more detail the specific case of speech and language data structures.

3.2.2 Linguistic case study

Establishing data models for deployed technology is certainly part of the Research and Development cycle, which means that data profiles tend to emerge within the scientific process of formulating and refining technical and algorithmic designs. This is particularly true for complex, computationally nuanced challenges such as image segmentation or (to cite one previous example) measuring heartbeats and breathing patterns via subtle waveform analysis. We can assume that every R&D phase will itself leave behind an ecosystem of testing data and code which can be decisive for consolidating data models, directly or indirectly influencing production code for systems (even if their deployment and commercialization is well after the R&D period).

In the case of speech and language technology, a research-oriented data infrastructure has been systematically curated, in several subdisciplines, by academic or industry collaborations. The Conference on Natural Language Learning (CoNLL), for example, invites participants to develop Natural Language Processing techniques targeted at a common “challenge” dataset, updated each year. These datasets, along with the data formats and code libraries which allow software to use that data, thereby serve as a reference-point for Computational Linguistics researchers in general. Similarly, this volume’s chapter on Language Identification describes research targeting a multilingual dataset (labeled AP17-OLR) curated for an annual “Oriental Language Challenge” conference dedicated to language/dialect classification for languages spoken around East Asia (from East Asian language families and also Russian).

Technically, curated and publicly accessible datasets are a different genre of information space than real-time data generated by Cyber-Physical technology (e.g., voices picked up by microphones in a Smart Home). That is to say, software developed to access speech and language datasets like the CoNLL’s or the Oriental Language Challenge has different requirements than software responding to voice requests in real time—or other deployment use-cases, such as medical transcription, or identifying dialects spoken in an urban community. However, data models derived from publicly shared test corpora *do* translate over to real-time data: we can assume that R&D datasets are collections of signals or information granules which are structurally similar to those produced by operating Cyber-Physical devices. As a result,

portions of the software targeting R&D datasets—specifically, the procedures for acquiring, transforming, validating, and interactively displaying individual samples—remain useful as components or prototypes for deployed products. Code libraries employed in R&D cycles should typically be the basis for data models guiding the implementation of production software.

To make this discussion more concrete, I will use the example of CoNLL datasets. This chapter’s demo includes samples from the most recent collection of CoNLL files and conference challenge tasks (at the time of writing) as well as demo code which operates on such data via techniques described in this chapter. The CoNLL format is representative of the kinds of linguistic parsing requisite for using Natural Language content (such as speech input) in Cyber-Physical settings.

This volume’s chapter by Teixeira et al. considers voice-activated Cyber-Physical interfaces; here Natural Language segments become the core elements in translating user queries to actionable software responses. The proposed systems analyze speech patterns to build textual reconstructions of speakers’ communications, then parse the text as Natural Language content, before eventually (if all goes well) interpreting the parsed and analyzed text as an instruction that the software can follow. Text data can then be supplemented with metrics measuring vocal patterns, syntactic and semantic information, speaker’s spatial location, and other information that can help interpret speakers’ wishes insofar as software can respond to them.

The authors discuss, for instance, the possibility of annotating language samples (after speech-to-text translation) with Dependency Grammar parses.^b Textual content can also be annotated with models of intonation, stress patterns, and other acoustic features (because the original inputs are audio-based) that can help an NLP engine to parse sentences properly (e.g., by noting which words or syllables are vocally emphasized). We can assume that audio processing as well as NLP technology would supply intermediary processing somewhere between acoustic devices and the centralized application.

Different kinds of linguistic details require different data models. Dependency parses, for instance, are often notated via some version of a specialized CoNLL format, which textually serializes parse and lexical data, usually one word per line. The most recent standard (dubbed CoNLL-U) recognizes 10 fields for each word, identifying, in particular, Parts of Speech and syntactic connections with other words [68–72]. As a custom format, CoNLL-U requires its own parser, such as the UDPipe library for C++ (a slightly modified version of this library is published with this chapter’s dataset). So, for hub applications, a reasonable assumption is that these programs compile in the UDPipe library or an alternative with similar capabilities, in order for them to handle parsed NLP data.

^bAdequately describing Dependency Grammar is outside the scope of this chapter, but, in a nutshell, Dependency Grammar models syntax in terms of word-to-word relations rather than via phrase hierarchies; as such, Dependency parses yields directed, labeled graphs (node labels are words and edge labels are drawn from an inventory of syntactic interword connections), which are structurally similar to Semantic Web graphs. See also [56–62], etc.; variants include Link Grammar [63, 64] and Extensible Dependency Grammar [65–67].

Suppose, then, that a hub application will periodically receive a data package comprising an audio sample along with text transcriptions and NLP -generated, for example, CoNLL-U data. To make sense of linguistic content, the software would presumably pair the NLP -specific information with extra details, such as the identity of the speaker (if a Smart Home system knows of specific users), where and when each sentence or request was formulated, and perhaps the original audio input (allowing functionality such as users playing back instructions they uttered in the past). A relevant data model might thereby comprise: (1) CoNLL-U data itself; (2) location info, such as spatial position and which room a speaker is found in; (3) timestamps; (4) speaker info, if available; (5) audio files; and maybe (6) extra acoustical or intonation data. (Those extra details could include annotations based on how conversation analysts note speech patterns, or might be waveform features derived from initial processing of speech samples.)

This data model would presumably translate to multiple data types: we can envision (1) a class for UDPipe sentences obtained from CoNLL-U; (2) a class for audio samples; and (3) speaker and time/location information; plus versions of these classes appropriate for GUIs and database persistence. In addition, these data requirements for speech and text samples only considers obtaining a valid parse for the text; to actually react to speech input, an application would need to map lexical data to terms and actions the software itself, in the context of its own capabilities, can recognize. For instance, *close the window* would map to an identifier for which window is intended (inferred perhaps from speaker location) and a *close* operation, which could be available via actuators embedded in the window area. All of the objects that users might semantically reference in voice commands therefore need their own data models, which must be interoperable with linguistic parses. So along with data types specific to linguistic elements we can consider “bridge” types connecting linguistic data (e.g., lexemes) to data types modeling physical objects.

Likewise, we can anticipate the procedures which speech and/or language data types need to implement: correctly decoding CoNLL-U files; mapping time/location data points to a spatial model of the Smart Home (which room is targeted by the location and also if that point is close to a door, window, appliance, and so forth; and perhaps matching the location to a 3D or panoramic-photography graphics model for visualization); audio-playback procedures, along with interactive protocols for this process such as users pausing and restarting playback; and procedures to map identified speakers to user profiles known to the Smart Home system. The audio devise makers and NLP providers—assuming those products are delivered as one or several suites separate and apart from the Smart Home hubⁱ—can mandate that hub applications demonstrate procedural implementations that satisfy these requirements as a precondition for accessing their broadcast data. Conversely, hub

ⁱFor this discussion I assume that hub applications do not perform NLP internally, but rather receive post-NLP input from NLP technologies designed with particular attention to vocal commands for “smart” objects.

applications can stipulate the procedural mandates they are prepared to honor as a guide to how devices and their drivers and middleware components should be configured for an integrated Smart Home ecosystem.

The essential point here is that procedural requirements and validation becomes the essential glue that unifies the diverse Smart Home components, and allows products designed by different companies, with different goals, to become interoperable. Once again, procedural alignment and predictability are more important than standardized data formats.

We can also consider representative criteria for testing procedures—in particular, preconditions that procedures need to recognize. In CoNLL-U, individual words can be extracted from a parse-model, but the numeric index for the word must fall within a fixed range (based on word count for the relevant sentence). In audio playback, time intervals are only meaningful in the context of the length of the audio sample in seconds or milliseconds. Similarly, features extracted from an audio sample (of human speech or, say, a bird song) are localized by time points which have to reside within a sample window; and image features are localized in rectangular coordinates that need to fit inside the surrounding image. Therefore, procedures engaged with these data structures should be checked to ensure that they honor these ranges and properly respond to faulty data outside them. This is an example of the kind of localized procedural testing that, cumulatively, establishes software as trustworthy.

I will discuss similar procedural-validity issues for the remainder of this section before developing more abstract or theoretical models of procedures, as formal constructions, subsequently in the chapter.

3.2.3 Proactive design

I have thus far argued that applications which process Cyber-Physical data should rigorously organize their functionality around specific devices' data profiles. The procedures that directly interact with devices—receiving data from and perhaps sending instructions to each one—will in many instances be “fragile” in the sense I invoke in this chapter. Each of these procedures may make assumptions legislated by the relevant device’s specifications, to the extent that using any one procedure too broadly constitutes a system error. Furthermore, Cyber-Physical devices may exhibit errors due to mechanical malfunction, hostile attacks, or one-off errors in electrical-computing operations, causing performance anomalies which look like software mistakes even if the code is entirely correct [73, 74]. As a consequence, *error classification* is especially important—distinguishing kinds of software errors and even which problems are software errors to begin with.

Summarizing the case studies from earlier in this section, Table 3.1 identifies several details about dimensions, parameters of operation, data fields, and other pieces of information relevant to implementing procedures and data types capturing Cyber-Physical data. These types may derive from Cyber-Physical input directly or may model artifacts constructed from Cyber-Physical input midstream, such as audio or image files, or text transcriptions representing speech input. The summaries are

not rigorous profiles, just suggestive cues about what sort of details engineers should consider when formalizing data models. In general, detailed models should be defined for any input source (including those transformed by middleware components, such as NLP engines), thereby profiling both Cyber-Physical devices and also “midstream” artifacts such as audio or image files—that is, aggregates, derived from Cyber-Physical input, that can be shared between software components (within hub applications and/or between hubs and middleware).

These data profiles need to be integrated with Cyber-Physical code from a perspective that cuts across multiple dimensions of project scale and lifetime. Do we design for biaxial or triaxial accelerometers, or both, and may this change? Is heart rate to be sampled in a context where the range considered normal is based on “resting” rate or is it expanded to factor in subjects who are exercising? These kinds of questions point to the multitude of subtle and project-specific specifications that have to be established when implementing and then deploying software systems in a domain like Ubiquitous Computing. It is unreasonable to expect that all relevant standards will be settled a priori by sufficiently monolithic and comprehensive data models. Instead, developers and end-users need to acquire trust in a development process which is configured to make standardization questions become apparent and capable of being followed up in system-wide ways.

For instance, the hypothetical questions I pondered in the previous paragraph—about biaxial versus triaxial accelerometers and about at-rest versus exercise heart-rate ranges—would not necessarily be evident to software engineers or project architects when the system is first conceived. These are the kind of modeling questions that tend to emerge as individual procedures and datatypes are implemented. For this reason, code development serves a role beyond just concretizing a system’s deployment software. The code at fine-grained scales also reveals questions that need to be asked at larger scales, and then the larger answers reflected back in the fine-grained coding assumptions, plus annotations and documentation. The overall project community needs to recognize software implementation as a crucial source for insights into the specifications that have to be established to make the deployed system correct and resilient.

For these reasons, code-writing—especially at the smallest scales—should proceed via paradigms disposed to maximize the “discovery of questions” effect (see also, as a case study, [75, pp. 6–10]). Systems in operation will be more trustworthy when and insofar as their software bears witness to a project evolution that has been well-poised to unearth questions that could otherwise diminish the system’s trustworthiness.

“Proactiveness,” like transparency and trustworthiness, has been identified as a core USH principle, referring (again in the series intro, as earlier) to “data transmission to healthcare providers …*to enable necessary interventions*” (my emphasis). In other words—or so this language implies, as an unstated axiom—patients need to be confident in deployed USH products to such degree that they are comfortable with clinical/logistical actions—the functional design of medical spaces; decisions about course of treatment—being grounded in part on data generated from a USH

ecosystem. This level of trust, or so I would argue, is only warranted if patients feel that the preconceived notions of a USH project have been vetted against operational reality—which can happen through the interplay between the domain experts who germinally envision a project and the programmers (software and software-language engineers) who, in the end, produce its digital substratum.

I have argued that hub *libraries*—intermediaries between Cyber-Physical devices and hub applications—are the key components where diverse requirements may be exercised. Hub libraries therefore need capabilities for documenting coding assumptions and requirements, such that their corresponding applications garner users' trust and acceptance. Hub applications, in short, would be deemed trustworthy insofar as their hub libraries are properly engineered. These, then, are the practical concerns driving the code-documentation proposals I will develop in the next two sections. Hub libraries are an environment where such techniques may be especially applicable.

3.3 Directed Hypergraphs and generalized lambda calculus

Thus far in this chapter, I have written in general terms about architectural features related to Cyber-Physical software; especially, verifying coding assumptions concerning individual data types and/or procedures. My comments were intended to summarize the relevant territory, so that I can add some theoretical details or suggestions from this point forward. In particular, I will explore how to model software components at different scales so as to facilitate robust, safety-conscious coding practices.

Note that almost all non-trivial software is in some sense “procedural”: the total package of functionality provided by each software component is distributed among many individual, interconnected procedures. Each procedure, in general, implements its functionality by calling *other* procedures in some strategic order. Of course, often inter-procedure calls are *conditional*—a calling procedure will call one (or some sequence of) procedures when some condition holds, but call alternate procedures when some other conditions hold. In any case, computer code can be analyzed as a graph, where connections exist between procedures insofar as one procedure calls, or sometimes calls, the other.

This general picture is only of only limited applicability to actual applications, however, because the basic concept of “procedure” varies somewhat between different programming languages. As a result, it takes some effort to develop a comprehensive model of computer code which accommodates a representative spectrum of coding styles and paradigms.

There are perhaps three different perspectives for such a comprehensive theory. One perspective is to consider source code as a data structure in its own right, employing a Source Code Algebra or Source Code Ontology to assert properties of source code and enable queries against source code, qua information space. A second option derives from type theory: to consider procedures as instances of

functional types, specified by tuples of input and output types. A procedure is then a transform which, in the presence of (zero or more) inputs having the proper types, produces (one or more) outputs with their respective types. (In practice, some procedures do not return values, but they *do* have some kind of side-effect, which can be analyzed as a variety of “output.”) Finally, third, procedures can be studied via mathematical frameworks such as Lambda Calculus, allowing notions of functions on typed parameters, and of functional application—applying functions to concrete values, which is analogous to calling procedures with concrete input arguments—to be made formally rigorous.

I will briefly consider all three of these perspectives—Source Code Ontology, type-theoretic models, and Lambda Calculus—in this section. I will also propose a new model, based on the idea of “channels,” which combines elements of all three.

3.3.1 Generalized lambda calculus

Lambda (or λ -)Calculus emerged in the early 20th century as a formal model of mathematical functions and function-application. There are many mathematical constructions which can be subsumed under the notion of “function-application,” but these have myriad notations and conventions (compare the visual differences between mathematical notations—integrals, square roots, super-scripted and sub-scripted indices, and so forth—to the much simpler alphabets of mainstream programming languages). But the early 20th century was a time of great interest in “mathematical foundations,” seeking to provide philosophical underpinnings for mathematical reasoning in general, unifying disparate mathematical methods and subdisciplines. One consequence of this foundational program was an attempt to capture the formal essence of the concept of “function” and of functions being applied to concrete values.

A related foundational concern is how mathematical formulae can be nested, yielding new formulae. For example, the volume of a sphere (expressed in terms of its radius R) is $\frac{4\pi R^3}{3}$. The symbol R is just a mnemonic which could be replaced with a different symbol, without the formula being different. But it can also be replaced by a more complex expression, to yield a new formula. In this case, substituting the formula for a cube’s half-diagonal— $\sqrt[3]{2V}$ where V is its volume—for R , in the first formula, yields $\frac{4}{3}\sqrt[3]{27\pi V}$: a formula for the sphere’s volume in terms of the volume of the largest cube that can fit inside it (Anderson [76] has similar interesting examples in the context of code optimization). This kind of tinkering with equations is, of course, a bread-and-butter of mathematical discovery. In terms of foundations research, though, observe that the derivation depended on two givens: that the R symbol is “free” in the first formula—it is a place-holder rather than the designation of a concrete value, like π —and that free symbols (like R) can be bound to other formulae, yielding new equations.

From cases like these—relatively simple geometric expressions—mathematicians began to ask foundational questions about mathematical formulae: what are all formulae that can be built up from a set of core equations via repeatedly

substituting nested expressions for free symbols? This question turns out to be related to the issue of finite calculations: in lieu of building complex formulae out of simpler parts, we can proceed in the opposite direction, replacing nested expressions with values. Formulae are constructed in terms of unknown values; when we have concrete measurements to plug in to those formulae, the set of unknowns decreases. If *all* values are known, then a well-constructed formula will converge to a (possibly empty) set of outcomes. This is roughly analogous to a computation which terminates in real time. On the other hand, a *recursive* formula—an expression nested inside itself, such as a continued fraction—is analogous to a computation that loops indefinitely.^j

In the early days of computer programming, it was natural to turn to λ -Calculus as a formal model of computer procedures, which are in some ways analogous to mathematical formulae. As a mathematical subject, λ -Calculus predates digital computers as we know them. While there were no digital computers at the time, there *was* a growing interest in mechanical computing devices, which led to the evolution of cryptographic machines used during the Second World War. So there was indeed a practical interest in “computing machines,” which eventually led to John von Neumann’s formal prototypes for digital computers.

Early on, though, λ -Calculus was less about blueprints for calculating machines and more about *abstract* formulation of calculational processes. Historically, the original purpose of λ -Calculus was largely a mathematical *simulation* of computations, which is not the same as a mathematical *prototype* for computing machines. Mathematicians in the decades before the Second World War investigated logical properties of computations, with particular emphasis on what sort of problems could always be solved in finite time, or what kinds of procedures can be guaranteed to terminate—a “Computable Number,” for example, is a number which can be approximated to any degree of precision by a terminating function. Similarly, a Computable Function is a function from input values to output values that can be associated with an always-terminating procedure which necessarily calculates the desired outputs from a set of inputs. The spaces of Computable Functions and Computable Numbers are mathematical objects whose properties can be studied through mathematical techniques—for instance, Computable Numbers are known to be a countable field within the real numbers. These mathematical properties are proven using a formal description of “any computer whatsoever,” which has no concern for the size and physical design of the “computers” or the time required for its “programs,” so long as they are finite. Computational procedures in this context are not actual implementations but rather mathematical distillations that can stand in for calculations for the purpose of mathematical analysis (interesting and representative contemporary articles continuing these perspectives include, for example, [77–79]).

^jAlthough there are sometimes techniques for converting formulae like Continued Fractions into “closed-form” equations which do “terminate.”

It was only after the emergence of modern digital computers that λ -Calculus became reinterpreted as a model of *concrete* computing machines. In its guise as a Computer Science (and not just Mathematical Foundations) discipline, λ -Calculus has been most influential not in its original form but in a plethora of more complex models which track the evolution of programming languages. Many programming languages have important differences which are not describable on a purely mathematical basis: two languages which are both “Turing complete” are abstractly interchangeable, but it is important to represent the contrast between, say, Object-Oriented and Functional programming. In lieu of a straightforward, mathematical model of formulae as procedures that map inputs to outputs, modern programming languages may add new constructs that determine different mechanisms whereby procedures can read and modify values: objects, exceptions, closures, mutable references, side-effects, signal/slot connections, and so forth. Accordingly, new programming constructions have inspired new variants of λ -Calculus, analyzing different features of modern programming languages—Object Orientation, Exceptions, call-by-name, call-by-reference, side-effects, polymorphic type systems, lazy evaluation—in the hopes of deriving formal proofs of program behavior insofar as computer code uses the relevant constructions. In short, a reasonable history can say that λ -Calculus mutated from being an abstract model for studying Computability as a mathematical concept, to being a paradigm for prototype specifications of concretely realized computing environments.

Modern programming languages have many different ways of handing-off values between procedures. The “inputs” to a function can be “message receivers” as in Object-Oriented programming, or lexically scoped values “captured” in an anonymous function that inherits values from the lexical scope (loosely, the area of source code) where its body is composed. Procedures can also “receive” data indirectly from pipes, streams, sockets, network connections, database connections, or files. All of these are potential “input channels” whereby a function implementation may access a value that it needs. In addition, procedures can “return” values not just by providing a final result but by throwing exceptions, writing to files or pipes, and so forth. To represent these myriad “channels of communication” computer scientists have invented a menagerie of extensions to λ -Calculus—a noteworthy example is the “Sigma” calculus to model Object-Oriented Programming; but parallel extensions represent call-by-need evaluation, exceptions, by-value and by-reference capture, etc.

Rather than study each system in isolation, in this chapter, I propose an integrated strategy for unifying disparate λ -Calculus extensions into an overarching framework. The “channel-based” tactic I endorse here may not be optimal for a *mathematical* calculus that has formal axioms and provable theorems, but I believe it can be useful for the more practical goal of modeling computer code and software components, to establish recommended design patterns and to document coding assumptions.

In this perspective, different extensions or variations to λ -Calculus model different *channels*, or data sources through which procedures receive and/or modify

values. Different channels have their own protocols and semantics for passing values to functions. We can generically discuss “input” and “output” channels, but programming languages have different specifications for different genres of input/output, which we can model via different channels. For a particular channel, we can recognize language-specific limitations on how values passed in to or received from those channels are used, and how the symbols carrying those values interact with other symbols both in function call-sites and in the body of procedure implementations. For example, procedures can output values by throwing exceptions, but exceptions are unusual values which have to be handled in specific ways—languages employ exceptions to signal possible programming errors, and they are engineered to interrupt normal program flow until or unless exceptions are “caught.”

Computer scientists have explored these more complex programming paradigms in part by inventing new variations on λ -calculi. Here I will develop one theory representing code in terms of Directed Hypergraphs, which are subject to multiple kinds of lambda abstraction—in principle, unifying multiple λ -Calculus extensions. The following section will lay out the details of this form of Directed Hypergraph and how λ -calculi can be defined on its foundation, while the last section summarizes an expanded type theory which follows organically from this approach.

Many concepts outlined here are reflected in the accompanying code set (which includes a C++ Directed Hypergraph library). My strategy for unifying multiple λ -calculi depends in turn on hypergraph code representations, which is a theme in the umbrella of graph-based data modeling, to which I now turn.

3.3.2 Directed Hypergraphs and “channel abstractions”

A *hypergraph* is a graph whose edges (a.k.a. “hyperedges”) can span more than two nodes [80, vol. 2, p. 24, 81–87]. A *directed* hypergraph (“DH”) is a hypergraph where each edge has a *head set* and *tail set* (both possibly empty). Both of these are sets of nodes which (when non-empty) are called *hypernodes*. A hypernode can also be thought of as a hyperedge whose tail-set (or head-set) is empty. Note that a typical hyperedge connects two hypernodes (its head- and tail-sets), so if we consider just hypernodes, a hypergraph potentially reduces to a directed ordinary graph.^k While “edge” and “hyperedge” are formally equivalent, I will use the former term when attending more to the edge’s representational role as linking two hypernodes, and use the latter term when focusing more on its tuple of spanned nodes irrespective of their partition into *head* and *tail*.

^kHere when distinguishing “head” and “tail” I will invert the orientation which most mathematical treatments of hypergraphs use; that is, I define hyperedges such that the edge *starts at* the head and *ends at* the tail. My rationale is that hyperedges induce an orientation not only on the head/tail pair, but *within* the head and tail, which become ordered tuples rather than sets. Hyperedges can therefore be seen as paths that “visit” a chain of hyponodes, first those in the head, then those in the tail. My terminology is consistent with software libraries wherein the *beginning* of an ordered list is called its “head.”

I assume that hyperedges always span an *ordered* node-tuple which induces an ordering in the head- and tail-sets: so a hypernode is an *ordered list* of nodes, not just a *set* of nodes. I will say that two hypernodes *overlap* if they share at least one node; they are *identical* if they share exactly the same nodes in the same order; and *disjoint* if they do not overlap at all. I call a Directed Hypergraph “reducible” if all hypernodes are either disjoint or identical. The information in reducible DHs can be factored into two “scales,” one a directed graph whose nodes are the original hypernodes, and then a table of all nodes contained in each hypernode. Reducible DHs allow ordinary graph traversal algorithms when hypernodes are treated as ordinary nodes on the coarser scale (so that their internal information—their list of contained nodes—is ignored).¹

To avoid confusion, I will hereafter use the word “hyponode” in place of “node,” to emphasize the container/contained relation between hyper-nodes and hypo-nodes. I will use “node” as an informal word for comments applicable to both hyper- and hypo-nodes. Some Hypergraph theories and/or implementations allow hypernodes to be nested: that is, a hypernode can contain another hypernode. In these theories, in the general case any node is potentially both a hypernode and a hyponode. For this chapter, I assume the converse: any “node” (as I am hereafter using the term) is *either* hypo- or hyper-. However, multiscale Hypergraphs can be approximated by using hyponodes whose values are proxies to hypernodes.

Here I will focus on a class of DHs which (for reasons to emerge) I will call “Channelizable.” Channelizable Hypergraphs (CHs) have these properties:

1. They have a Type System \mathbb{T} and all hyponodes and hypernodes are assigned exactly one canonical type (they may also be considered instances of super-types or subtypes of that type).
2. All hyponodes can have (or “express”) at most one value, an instance of its canonical type, which I will call a *hypovertex*. Hypernodes, similarly, can have at most one *hypovertex*. Like “node” being an informal designation for hypo-nodes and hyper-nodes, “vertex” will be a general term for both hypo-vertices and hyper-vertices. Nodes that do have a vertex are called *initialized*. The hypovertices “of” a hypernode are those of its hyponodes.
3. Two hyponodes are “equatable” if they express the same value of the same type. Two (possibly non-identical) hypernodes are “equatable” if all of their hyponodes, compared one by one in order, are equitable. I will also say that values are “equatable” (rather than just saying “equal”) to emphasize that they are the respective values of equatable nodes.

¹A weaker restriction on DH nodes is that two non-identical hypernodes *can* overlap, but must preserve node-order: that is, if the first hypernode includes nodes N_1 , and N_2 immediately after, and the second hypernode also includes N_1 , then the second hypernode must also include N_2 immediately thereafter. Overlapping hypernodes cannot “permute” nodes—cannot include them in different orders or in a way that “skips” nodes. Trivially, all reducible DHs meet this condition. Any graphs discussed here are assumed to meet this condition.

4. There may be a stronger relation, defined on equatable non-equivalent hypernodes, whereby two hypernodes are *inferentially equivalent* if any inference justified via edges incident to the first hypernode can be freely combined with inferences justified via edges incident to the second hypernode. Equatable nodes are not necessarily inferentially equivalent.
5. Hypernodes can be assumed to be unique in each graph, but it is unwarranted to assume (without type-level semantics) that two equatable hypernodes in different graphs are or are not inferentially equivalent. Conversely, even if graphs are uniquely labeled—which would appear to enable a formal distinction between hypernodes in one graph from those in another—CH semantics do not permit the assumption that this separation alone justifies inferences presupposing that their hypernodes *are not* inferentially equivalent.
6. All hypo-nodes and hyper-nodes have a “proxy,” meaning there is a type in \mathbb{T} including, for each node, a unique identifier designating that node, that can be expressed in other hyponodes.
7. There are some types (including these proxies) which may only be expressed in hyponodes. There may be other types which may only be expressed in hypernodes. Types can then be classified as “hypotypes” and “hypertypes.” The \mathbb{T} may stipulate that all types are *either* hypo or hyper. In this case, it is reasonable to assume that each hypotype maps to a unique hypertype, similar to “boxing” in a language which recognizes “primitive” types (in Object-Oriented languages, boxing allows non-class-type values to be used as if they were objects).
8. Types may be subject to the restriction that any hypernode which has that type can only be a tail-set, not a head-set; call these *tail-only* types.
9. Hyponodes may not appear in the graph outside of hypernodes. However, a hypernode is permitted to contain only one hyponode.
10. Each edge, separate and apart from the CH’s actual graph structure, is associated with a distinct hypernode, called its *annotation*. This annotation cannot (except via a proxy) be associated with any other hypernode (it cannot be a head- or tail-set in any hypernode). The first hyponode in its annotation I will dub a hyperedge’s *classifier*. The outgoing edge-set of a hypernode can always be represented as an associative array indexed by the classifier’s vertex.
11. A hypernode’s type may be subject to restrictions such that there is a single number of hyponodes shared by all instances. However, other types may be expressed in hypernodes whose size may vary. In this case, the hyponode types cannot be random; there must be some pattern linking the distribution of hyponode types evident in hypernodes (with the same hypernode types) of different sizes. For example, the hypernodes may be dividable into a fixed-size, possibly empty sequence of hyponodes, followed by a chain of hyponode-sequences repeating the same type pattern. The simplest manifestation of this structure is a hypernode all of whose hyponodes are the same type.
12. Call a *product-type transform* of a hypernode to be a different hypernode whose hypovertices are tuples of values equatable to those from the first

hypernode, typed in terms of product types (i.e., tuples). For example, consider two different representations of semi-transparent colors: as a four-vector RGBT, or as an RGB three-vector paired with a transparency magnitude. The second representation is a product-type transform of the first, because the first three values are grouped into a three-valued tuple. We can assert the requirement in most contexts that CHs whose hypernodes are product-type transforms of each other contain “the same information” and as sources of information are interchangeable.

13. The Type System \mathbb{T} is *channelized*, that is, closed under a Channel Algebra, as will be discussed later.

These definitions allude to two strategies for computationally representing CHs. One, already mentioned, is to reduce them to directed graphs by treating hypernodes as integral units (ignoring their internal structure). A second is to model hypernodes as a “table of associations” whose keys are the values of the classifier hyponodes on each of their edges. A CH can also be transformed into an *undirected* hypergraph by collapsing head- and tail-sets into an overarching tuple. All of these transformations may be useful in some analytic/representational contexts, and CHs are flexible in part by morphing naturally into these various forms.

Notice that information present *within* a hypernode can also be expressed as relations *between* hypernodes. For example, consider the information that I (Nathaniel), age 46, live in Brooklyn as a registered Democrat. This may be represented as a hypernode with hyponodes $\langle [\text{Nathaniel}], [46] \rangle$, connected to a hypernode with hyponodes $\langle [\text{Brooklyn}], [\text{Democrat}] \rangle$, via a hyperedge whose classifier encodes the concept “lives in” or “is a resident of.” However, it may also be encoded by “unplugging” the “age” attribute so the first hypernode becomes just $[\text{Nathaniel}]$ and it acquires a new edge, whose tail has a single hyponode $[46]$ and a classifier (encoding the concept) “age” (see the comparison in Fig. 3.1). This construction can work in reverse: information present in a hyperedge can be refactored so that it “plugs in” to a single hypernode.

These alternatives are not redundant. Generally, representing information via hyperedges connecting two hypernodes implies that this information is somehow conceptually apart from the hypernodes themselves, whereas representing

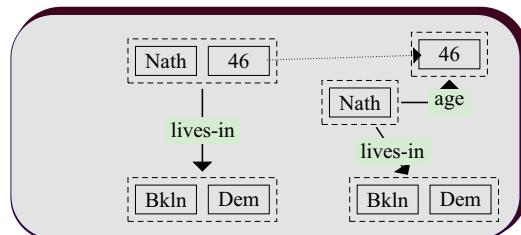


FIG. 3.1

Unplugging a node.

information via hyponodes *inside* hypernodes implies that this information is central and recurring (enforced by types), and that the data thereby aggregated forms a recurring logical unit. In a political survey, people’s names may *always* be joined to their age, and likewise their district of residence *always* joined to their political affiliation. The left-hand side representation of the info (seen as an undirected hyperedge) $\langle [\text{Nathaniel}], [\text{46}], [\text{Brooklyn}], [\text{Democrat}] \rangle$ in Fig. 3.1 captures this semantics better because it describes the name/age and place/party pairings as *types* which require analogous node-tuples when expressed by other hypernodes. For example, any two hypernodes with the same type as $\langle [\text{Nathaniel}], [\text{46}] \rangle$ will necessarily have an “age” hypervertex and so can predictably be compared along this one axis. By contrast, the right-hand (“unplugged”) version in Fig. 3.1 implies no guarantees that the “age” data point is present as part of a recurring pattern.

The two-tiered DH structure is also a factor when integrating serialized or shared data structures with runtime data values. In the demo DH library, for example, it is assumed that each node can be associated with a runtime, binary data allocation (practically speaking, a pointer to user data). Hypernodes’ internal structure can therefore be represented *either* via hyponodes explicit in the graph content *or* by internal structure in the user data (or some combination). Graph deserialization can then be a matter of mapping hyponodes to fields in the “internal” data allocations, before then mapping inter-hypernode relations to the proper hypervertex relations. Code Sample 3.1 demonstrates the pattern of hypervertex construction as C++ objects that get wrapped in new nodes (1 to 2), along with obtaining nodes already registered in a runtime graph (3) and then inserting the new nodes (with stated relationships) alongside prior ones into the runtime graph (4).^m

In general, graph representations like CH and RDF serve two goals: first, they are used to *serialize* data structures (so that they may be shared between different locations, such as via the internet); and, second, they provide formal, machine-readable descriptions of information content, allowing for analyses and transformations, to infer new information or produce new data structures. The design and rationale of representational paradigms is influenced differently by these two goals, as I will review now with an eye in part on drawing comparisons between CH and RDF.

3.3.3 Channelized hypergraphs and RDF

The Resource Description Framework (RDF) models information via directed graphs (Refs. [88–91] are good discussions of Semantic Web technologies from a graph-theoretic perspective), whose edges are labeled with concepts that, in well-structured contexts, are drawn from published Ontologies (these labels play a similar role to “classifiers” in CHs). In principle, all data expressed via RDF graphs is defined by

^mThe code samples in this text are drawn from a working demo at the time of writing; the actual code belonging to a downloadable dataset at the time of publication may be slightly revised. The dataset will include components to help readers cross-reference between the chapter’s samples and working demo code.

Sample 3.1 Initializing hypernodes

```
caon_ptr<RE_Node> RE_Graph_Build::make_new_node(
    caon_ptr<RE_Function_Def_Entry> fdef)
{
    caon_ptr<RE_Node> result = new RE_Node(fdef);
    RELAE_SET_NODE_LABEL(result, "<fdef>");
    return result;
}

...
caon_ptr<RE_Node> RE_Graph_Build::
    new_function_def_entry_node(RE_Node& prior_node,
        RE_Function_Def_Kinds kind,
        caon_ptr<RE_Node> label_node)
{
    caon_ptr<RE_Function_Def_Entry> fdef = new
        RE_Function_Def_Entry(&prior_node,
            kind, label_node);
    caon_ptr<RE_Node> result = make_new_node(fdef);
    fdef->set_node(result);
    return result;
}

...
caon_ptr<RE_Node> RE_Graph_Build::create_tuple(
    RE_Tuple_Info::Tuple_Formations tf,
    RE_Tuple_Info::Tuple_Indicators ti,
    RE_Tuple_Info::Tuple_Formations sf,
    bool increment_id)
{
    int tuple_id = increment_id?++tuple_entry_count_:0;
    caon_ptr<RE_Tuple_Info> tinfo = new RE_Tuple_Info(
        tf, ti, tuple_id);
    caon_ptr<RE_Node> result = new RE_Node(tinfo); 1
    return result;
}

...
caon_ptr<RE_Node> RE_Markup_Position::
    check_implied_lambda_tuple(
        RE_Function_Def_Kinds kind)
{
    ...
    if(caon_ptr<RE_Call_Entry> rce =
        current_node_->re_call_entry())
    {
        ...
        caon_ptr<RE_Node> fdef_node = graph_build_-> 2
            new_function_def_entry_node(
                *last_pre_entry_node_, kind);
        last_pre_entry_node_->delete_relation(
            rq_.Run_Call_Entry, current_node_);
        current_function_def_entry_node_ = fdef_node;
        caon_ptr<RE_Node> tuple_info_node = graph_build_->
            create_tuple_node( 2
                RE_Tuple_Info::Tuple_Formations::Indicates_Input
                , RE_Tuple_Info::Tuple_Indicators::Enter_Array
                , RE_Tuple_Info::Tuple_Formations::N_A );
        caon_ptr<RE_Node> entry_node = 3
            rq_.Run_Call_Entry(current_node_);
        ...
        fdef_node << fr_/rq_.Run_Call_Entry >> 4
            current_node_;
        current_node_ << fr_/rq_.Run_Data_Entry >> 4
            tuple_info_node;
        tuple_info_node << fr_/rq_.Run_Data_Entry >> 4
            entry_node;
    ...}}}
```

unordered sets of labeled edges, also called “triples” (“**SUBJECT, PREDICATE, OBJECT**,” where the “Predicate” is the label). In practice, however, higher-level RDF notations such as TTL (**TURTLE** or “Terse RDF Triple Language”) and Notation3 (**N3**) deal with aggregate groups of data, such as RDF containers and collections.

For example, imagine a representation of the fact “(A/The person named) Nathaniel, 46, has lived in Brooklyn, Buffalo, and Montreal” (shown in Fig. 3.2 as both a CH and in RDF). If we consider TURTLE or N3 as *languages* and not just *notations*, it would appear as if their semantics is built around hyperedges rather than triples. It would seem that these languages encode many-to-many or one-to-many assertions, graphed as edges having more than one subject and/or predicate. Indeed, Tim Berners-Lee himself suggests that “Implementations may treat list as a data type rather than just a ladder of rdf:first and rdf:rest properties” [92, p. 6]. That is, the specification for RDF list-type data structures invites us to consider that they *may* be regarded integral units rather than just aggregates that get pulled apart in semantic interpretation.

Technically, perhaps, this is an illusion. Despite their higher-level expressiveness, RDF expression languages are, perhaps, supposed to be deemed “syntactic sugar” for a more primitive listing of triples: the *semantics* of TURTLE and N3 are conceived to be defined by translating expressions down to the triple sets that they logically imply (see also [93]). This intention accepts the paradigm that providing semantics for a formal language is closely related to defining which propositions are logically entailed by its statements.

There is, however, a divergent tradition in formal semantics that is oriented to type theory more than logic. It is consistent with this alternative approach to see a different semantics for a language like TURTLE, where larger-scale aggregates become “first class” values. So, $\langle [\text{Nathaniel}], [\text{46}] \rangle$ can be seen as a (single, integral) *value*

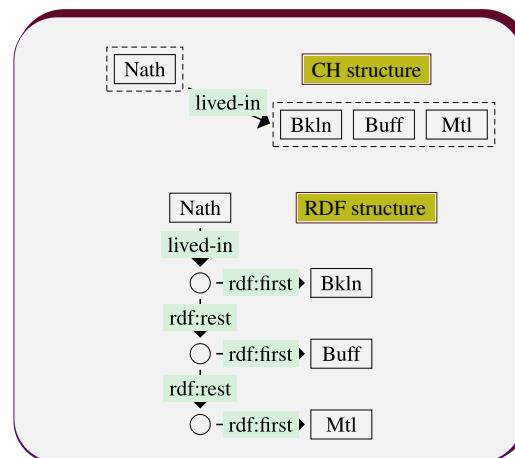


FIG. 3.2

CH versus RDF collections.

whose *type* is a $\langle \text{name}, \text{age} \rangle$ pair. Such a value has an “internal structure” which subsumes multiple data points. The RDF version is organized, instead, around a *blank node* which ties together disparate data points, such as my name and my age. This blank node is also connected to another blank node which ties together place and party. The blank nodes play an organizational role, since nodes are grouped together insofar as they connect to the same blank node. But the implied organization is less strictly entailed; one might assume that the $\langle [\text{Brooklyn}], [\text{Democrat}] \rangle$ nodes could just as readily be attached individually to the “name/age” blank (i.e., I live in Brooklyn, *and* I vote Democratic).

Why, that is, are Brooklyn and Democratic grouped together? What concept does this fusion model? There is a presumptive rationale for the name/age blank (i.e., the fusing name/age by joining them to a blank node rather than allowing them to take edges independently): conceivably there are multiple 46-year olds named Nathaniel, so *that* blank node plays a key semantic role (analogous to the quantifier in “*There is a Nathaniel, age 46...*”); it provides an unambiguous nexus so that further predicates can be attached to *one specific* 46-year-old Nathaniel rather than any old $\langle [\text{Nathaniel}], [46] \rangle$. But there is no similarly suggested semantic role for the “place/party” grouping. The name cannot logically be teased apart from the name/age blank (because there are multiple Nathaniels), but there seems to be no *logical* significance to the place/party grouping. Yet pairing these values *can* be motivated by a modeling convention—reflecting that geographic and party affiliation data are grouped together in a dataset or data model. The logical semantics of RDF make it harder to express these kinds of modeling assumptions that are driven by convention more than logic—an abstracting from data’s modeling environment that can be desirable in some contexts but not in others.

So, why does the Semantic Web community effectively insist on a semantic interpretation of TURTLE and N3 as *just* a notational convenience for N-TRIPLES rather than as higher-level languages with a different higher-level semantics—and despite statements like the earlier Tim Berners-Lee quote insinuating that an alternative interpretation has been contemplated even by those at the heart of Semantic Web specifications? Moreover, defining hierarchies of material composition or structural organization—and so by extension, potentially, distinct scales of modeling resolution—has been identified as an intrinsic part of domain-specific Ontology design (see Refs. [94–101], or Ref. [102]). Semantic Web advocates have not, however, promoted multilayer structure as a feature of Semantic models fundamentally, as opposed to criteriology *within* specific Ontologies. To the degree that this has an explanation, it probably has something to do with reasoning engines: the tools that evaluate SPARQL queries operate on a triplestore basis. So the “reductive” semantic interpretation is arguably justified via a warrant that the definitive criteria for Semantic Web representations are not their conceptual elegance vis-à-vis human judgments but their utility in cross-ontology and cross-context inferences.

As a counter-argument, however, note that many inference engines in Constraint Solving, Computer Vision, and so forth, rely on specialized algorithms and cannot be reduced to a canonical query format. Libraries such as GeCODE and ITK are important

because problem solving in many domains demands fine-tuned application-level engineering. We can think of these libraries as supporting *special* or domain-specific reasoning engines, often built for specific projects, whereas OWL-based reasoners like FACT++ are *general* engines that work on general-purpose RDF data without further qualification. In order to apply “special” reasoners to RDF, a contingent of nodes must be selected that is consistent with reasoners’ runtime requirements.

Of course, special reasoners cannot be expected to run on the domain of the entire Semantic Web, or even on “very large” datasets in general. A typical analysis will subdivide its problem into smaller parts that are each tractable to custom reasoners—in radiology, say, a diagnosis may proceed by first selecting a medical image series and then performing image-by-image segmentation. Applied to RDF, this two-step process can be considered a combination of general and special reasoners: a general language like SPARQL filters many nodes down to a smaller subset, which are then mapped/deserialized to domain-specific representations (including runtime memory). For example, RDF can link a patient to a diagnostic test, ordered on a particular date by a particular doctor, whose results can be obtained as a suite of images—thereby selecting the particular series relevant for a diagnostic task. General reasoners can *find* the images of interest and then pass them to special reasoners (such as segmentation algorithms) to analyze. Insofar as this architecture is in effect, Semantic Web data are a site for many kinds of reasoning engines. Some of these engines need to operate by transforming RDF data and resources to an optimized, internal representation. Moreover, the semantics of these representations will typically be closer to a high-level N3 semantics taken as *sui generis*, rather than as interpreted reductively as a notational convenience for lower-level formats like N-TRIPLE. This appears to undermine the justification for reductive semantics in terms of OWL reasoners.

Perhaps the most accurate paradigm is that Semantic Web data have two different interpretations, differing in being consistent with special and general semantics, respectively. It makes sense to label these the “special semantic interpretation” or “semantic interpretation for special-purpose reasoners” (SSI, maybe) and the “general semantic interpretation” (GSI), respectively. Both these interpretations should be deemed to have a role in the “semantics” of the Semantic Web.

Another order of considerations involve the semantics of RDF nodes and CH hypernodes particularly with respect to uniqueness. Nodes in RDF fall into three classes: blank nodes; nodes with values from a small set of basic types like strings and integers; and nodes with URLs that are understood to be unique across the entire World Wide Web. There are no blank nodes in CH, and intrinsically no URLs either, although one can certainly define a URL *type*. There is nothing in the semantics of URLs which guarantees that each URL designates a distinct internet resource; this is just a convention which essentially fulfills itself *de facto* because it structures a web of commercial and legal practices, not just digital ones; for example, ownership is uniquely granted for each internet domain name. In CH, a data type may be structured to reflect institutional practices that guarantee the uniqueness of values in some context: books have unique ISBN codes; places have distinct GIS locations, etc. These uniqueness requirements, however, are not intrinsically part of CH, and need to be

expressed with additional axioms. In general, a CH hypernode is a tuple of relatively simple values and any additional semantics are determined by type definitions (it may be useful to see CH hypernodes as roughly analogous to C **structs**—which have no *a priori* uniqueness mechanism).

Also, RDF types are less intrinsic to RDF semantics than in CH [103]. The foundational elements of CH are value-tuples (via nodes expressing values, whose tuples in turn are hypernodes). Tuples are indexed by position, not by labels: the tuple ⟨[Nathaniel], [46]⟩ does not in itself draw in the labels “name” or “age,” which instead are defined at the type-level (insofar as type-definitions may stipulate that the label “age” is an alias for the node in its second position, etc.). So there is no way to ascertain the semantic/conceptual intent of hypernodes without considering both hyponode and hypernode types. Conversely, RDF does not have actual tuples (though these can be represented as collections, if desired); and nodes are always joined to other nodes via labeled connectors—there is no direct equivalent to the CH modeling unit of a hyponode being included in a hypernode by position.

At its core, then, RDF semantics are built on the proposition that many nodes can be declared globally unique by fiat. This does not need to be true of all nodes—RDF types like integers and floats are more ethereal; the number 46 in one graph is indistinguishable from 46 in another graph. This can be formalized by saying that some nodes can be *objects* but never *subjects*. If such restrictions were not enforced, then RDF graphs could become in some sense overdetermined, implying relationships by virtue of quantitative magnitudes devoid of semantic content. This would open the door to bizarre judgments like “my age is non-prime” or “I am older than Mohamed Salah’s 2018 goal totals.” One way to block these inferences is to prevent nodes like “the number 46” from being subjects as well as objects. But nodes which are not primitive values—ones, say, designating Mohamed Salah himself rather than his goal totals—are justifiably globally unique, since we have compelling reasons to adopt a model where there is exactly one thing which is *that* Mohamed Salah. So RDF semantics basically marries some primitive types that are objects but never subjects with a web of globally unique but internally unstructured values which can be either subject or object.

In CH, the “primitive” types are effectively hypotypes; hyponodes are (at least indirectly) analogous to object-only RDF nodes insofar as they can only be represented via inclusion inside hypernodes. But CH hypernodes are neither (in themselves) globally unique nor lacking in internal structure. In essence, an RDF semantics based on guaranteed uniqueness for atom-like primitives is replaced by a semantics based on structured building-blocks without guaranteed uniqueness. This alternative may be considered in the context of general versus special reasoners: since general reasoners potentially take the entire Semantic Web as their domain, global uniqueness is a more desired property than internal structure. However, since special reasoners only run on specially selected data, global uniqueness is less important than efficient mapping to domain-specific representations. It is not computationally optimal to deserialize data by running SPARQL queries.

Finally, as a last point in the comparison between RDF and CH semantics, it is worth considering the distinction between “declarative knowledge” and “procedural knowledge” (see, e.g., [80, vol. 2, pp. 182–197]). According to this distinction, canonical RDF data exemplify *declarative* knowledge because they assert apparent facts without explicitly trying to interpret or process them. Declarative knowledge circulates among software in canonical, reusable data formats, allowing individual components to use or make inferences from data according to their own purposes.

Counter to this paradigm, return to hypothetical Cyber-Physical examples, such as the conversion of voltage data to acceleration data, which is a prerequisite to accelerometers’ readings being useful in most contexts. Software possessing capabilities to process accelerometers therefore reveals what can be called *procedural* knowledge, because software so characterized not only receives data but also processes such data in standardized ways.

The declarative/procedural distinction perhaps fails to capture how procedural transformations may be understood as intrinsic to some semantic domains—so that even the information we perceive as “declarative” has a procedural element. For example, the very fact that “accelerometers” are not called “Voltmeters” (which are something else) suggests how the Ubiquitous Computing community perceives voltage-to-acceleration calculations as intrinsic to accelerometers’ data. But strictly speaking, the components that participate in USH networks are not just engaged in data sharing; they are functioning parts of the network because they can perform several widely recognized computations that are understood to be central to the relevant domain—in other words, they have (and share with their peers) a certain “procedural knowledge.”

RDF is structured as if static data sharing were the sole arbiter of semantically informed interactions between different components, which may have a variety of designs and rationales—which is to say, a Semantic Web. But a thorough account of formal communication semantics has to reckon with how semantic models are informed by the implicit, sometimes unconscious assumption that producers and/or consumers of data will have certain operational capacities: the dynamic processes anticipated as part of sharing data are hard to separate conceptually from the static data which is literally transferred. To continue the accelerometer example, designers can think of such instruments as “measuring acceleration” even though *physically* this is not strictly true; their output must be mathematically transformed for it to be interpreted in these terms. Whether represented via RDF graphs or Directed Hypergraphs, the semantics of shared data is incomplete unless the operations which may accompany sending and receiving data are recognized as preconditions for legitimate semantic alignment.

While ontologies are valuable for coordinating and integrating disparate semantic models, the Semantic Web has perhaps influenced engineers to conceive of semantically informed data sharing as mostly a matter of presenting static data conformant to published Ontologies (i.e., alignment of “declarative knowledge”). In reality, robust data sharing also needs an “alignment of *procedural* knowledge”: in an ideal Semantic Network, procedural capabilities are circled among components,

promoting an emergent “collective procedural knowledge” driven by transparency about code and libraries as well as about data and formats. The CH model arguably supports this possibility because it makes type assertions fundamental to semantics. Rigorous typing both lays a foundation for procedural alignment and mandates that procedural capabilities be factored in to assessments of network components, because a type attribution has no meaning without adequate libraries and code to construct and interpret type-specific values.



Despite their differences, the Semantic Web, on the one hand, and Hypergraph-based frameworks, on the other, both belong to the overall space of graph-oriented semantic models. Hypergraphs can be emulated in RDF, and RDF graphs can be organically mapped to a Hypergraph representation (insofar as Directed Hypergraphs with annotations are a proper superspace of Directed Labeled Graphs). Semantic Web Ontologies for computer source code can thus be modeled by suitably typed DHs as well, even while we can also formulate Hypergraph-Based Source Code Ontologies as well. So, we are justified in assuming that a sufficient ontology exists for most or all programming languages. This means that, for any given procedure, we can assume that there is a corresponding DH representation which embodies that procedure’s implementation.

Procedures, of course, depend on *inputs* which are fixed for each call, and produce “outputs” once they terminate. In the context of a graph-representation, this implies that some hypernodes represent and/or express values that are *inputs*, while others represent and/or express its *outputs*. These hypernodes are *abstract* in the sense (as in Lambda Calculus) that they do not have a specific assigned value within the body, *qua* formal structure. Instead, a *runtime manifestation* of a DH (or equivalently a CH, once channelized types are introduced) populates the abstract hypernodes with concrete values, which in turn allows expressions described by the CH to be evaluated.

These points suggest a strategy for unifying Lambda calculi with Source Code Ontologies. The essential construct in λ -calculi is that mathematical formulae include “free symbols” which are *abstracted*: sites where a formula can give rise to a concrete value, by supplying values to unknowns; or give rise to new formulae, via nested expressions. Analogously, nodes in a graph-based source-code representation are effectively λ -abstracted if they model input parameters, which are given concrete values when the procedure runs. Connecting the output of one procedure to the input of another—which can be modeled as a graph operation, linking two nodes—is then a graph-based analog to embedding a complex expression into a formula (via a free symbol in the latter).

Carrying this analogy further, I earlier mentioned different λ -Calculus extensions inspired by programming-language features such as object-orientation, exceptions, and by-reference or by-value captures. These, too, can be incorporated into a Source Code Ontology: for example, the connection between a node holding a value passed

to an input parameter node, in a procedure signature, is semantically distinct from the nodes holding “Objects” which are senders and receivers for “messages,” in Object-Oriented parlance. Variant input/output protocols, including objects, captures, and exceptions, are certainly semantic constructs (in the computer-code domain) which Source Code Ontologies should recognize. So we can see a convergence in the modeling of multifarious input/output protocols via λ -Calculus and via Source Code Ontologies. I will now discuss a corresponding expansion in the realm of applied Type Theory, with the goal of ultimately folding type theory into this convergence as well.

3.3.4 Procedural input/output protocols via type theory

Parallel to the historical evolution where λ -Calculus progressively diversified and re-oriented toward concrete programming languages, there has been an analogous (and to some extent overlapping) history in Type Theory. When there are multiple ways of passing input to a function, there are potentially multiple kinds of function types. For instance, object-orientation inspired expanded λ -calculi that distinguish function inputs which are “method receivers” or “**this** objects” from ordinary (“lambda”) inputs. Simultaneously, object-orientation also distinguishes “class” from “value” types and between function-types which are “methods” versus ordinary functions. So, to take one example, a function telling us the size of a list can exhibit two different types, depending on whether the list itself is passed in as a method-call target (`list.size()` vs. `size(list)`).

One way to systematize the diversity of type systems is to assume that, for any particular type system, there is a category \mathbb{T} of types conformant to that system. This requires modeling important type-related concepts as “morphisms” or maps between types. Another useful concept is an “endofunctor”: an “operator” which maps elements in a category to other (or sometimes the same) elements. In a \mathbb{T} , an endofunctor selects (or constructs) a type t_2 from a type t_1 —note how this is different from a morphism which maps *values of* t_1 to t_2 . Type systems are then built up from a smaller set of “core” types via operations like products, sums, enumerations, and forming “function-like” types.

We may think of the “core” types for practical programming as number-based (booleans, bytes, and larger integer types), with everything else built up by aggregation or encodings (like ASCII and UNICODE, allowing types to include text and alphabets; or pixel-coordinates and colors, allowing for graphical/visual components).ⁿ Ultimately, a type system \mathbb{T} is characterized (1) by which are its core types and (2) by how aggregate types are built from simpler ones (which essentially involves endofunctors and/or products).

ⁿIn other contexts, however, non-mathematical core types may be appropriate: for example, the grammar of natural languages can be modeled in terms of a type system whose core are the two types **Noun** and **Proposition** and which also includes function types (maps) between pairs or tuples of types (verbs, say, map **Nouns**—maybe multiple nouns, for example, direct objects—to **Propositions**).

In Category Theory, a Category \mathbb{C} is called “Cartesian Closed” if for every pair of elements e_1 and e_2 in \mathbb{C} there is an element $e_1 \rightarrow e_2$ representing (for some relevant notion of “function”) all functions from e_1 to e_2 [104]. The stipulation that a type system \mathbb{T} include function-like types is roughly equivalent, then, to the requirement that \mathbb{T} , seen as a Category, is Cartesian-Closed. The historical basis for this concept (suggested by the terminology) is that the construction to form function-types is an “operator,” something that creates new types out of old. A type system \mathbb{T} may then be “closed” under products: if t_1 and t_2 are in \mathbb{T} then $t_1 \times t_2$ must be as well. Analogously, \mathbb{T} supports function-like types if it is closed under a kind of “functionalization” operator—if the $t_1 \times t_2$ product can be mapped onto a function-like type $t_1 \rightarrow t_2$.

In general, more sophisticated type systems \mathbb{T} are described by identifying new kinds of inter-type operators and studying those type systems that are closed under these operators: if t_1 and t_2 are in \mathbb{T} then so is the combination of t_1 and t_2 , where the meaning of “combination” depends on the operator being introduced. Expanded λ -calculi—which define new ways of creating functions—are correlated with new type systems, insofar as “new ways of creating functions” also means “new ways of combining types into function-like types.”

Furthermore, “expanded” λ -calculi generally involve “new kinds of abstraction”: new ways that the building-blocks of functional expressions, whether these be mathematical formulae or bodies of computer code, can be “abstracted,” treated as inputs or outputs rather than as fixed values. In this chapter, I attempt to make the notion of “abstraction” rigorous by analyzing it against the background of DHs that formally model computer code. So, given the correlations I have just described between λ -calculi and type systems—specifically, on \mathbb{T} -closure stipulations—there are parallel correlations between type systems and *kinds of abstraction defined on Channelized Hypergraphs*. I will now discuss this further.

3.3.4.1 Kinds of abstraction

The “abstracted” nodes in a CH are loosely classifiable as “input” and “output,” but in practice there are various paradigms for passing values into and out of functions, each with their own semantics. For example, a “**this**” symbol in C++ is an abstracted, “input” hypernode with special treatment in terms of overload resolution and access controls. Similarly, exiting a function via **return** presents different semantics than exiting via **throw**. As mentioned earlier, some of this variation in semantics has been formally modeled by different extensions to λ -Calculus.

So, different hypernodes in a CH are subject to different kinds of abstraction. Speaking rather informally, hypernodes can be grouped into *channels* based on the semantics of their kind of abstraction. More precisely, channels are defined initially on *symbols*, which are associated with hypernodes: in any “body” (i.e., an “implementation graph”) hypernodes can be grouped together by sharing the same symbol, and correlatively sharing the same value during a “runtime manifestation” of the CH. Therefore, the “channels of abstraction” at work in a procedure can be

identified by providing a name representing the *kind* of channel and a list of symbols affected by that kind of abstraction.

I propose “Channel Algebra” as a tactic for capturing the semantics of channels, so as to model programming languages’ conventions and protocols with respect to calls between procedures. Once we get beyond the basic contrast between “input” and “output” parameters, it becomes necessary to define conditions on channels’ size, and on how channels are associated with different procedures that may share values. Here are several examples:

- In most Object-Oriented languages, any procedure can have at most one **this** (“message receiver”) object. Let **sigma** model a “Sigma” channel, as in “Sigma Calculus” (written as ς -calculus: see, e.g., [105–108], etc.). We then have the requirement than any procedure’s **sigma** channel can carry at most one value.
- In all common languages which have exceptions, procedures can *either* throw an exception *or* return a value. If **return** and **exception** model the channels carrying standard returns and thrown exceptions, respectively, this convention translates to a requirement that the two channels cannot both be non-empty.
- A thrown exception cannot be handled as an ordinary value. The whole point of throwing exceptions is to disrupt ordinary program flow, which means the exception value is only accessible in special constructs, like a **catch** block. One way to model this restriction is to forbid **exception** channels from transferring values to other channels. Instead, exception values are bound (in **catch** blocks) to lexically scoped symbols (I will discuss channel-to-symbol transfers later).
- Suppose a procedure is an Object-Oriented method (it has a non-empty “**sigma**” channel). Any other methods called from that procedure will—at least in the conventional Object-Oriented protocol—automatically receive the enclosing method’s **sigma** channel unless a different object for the called method is supplied expressly.
- In the object-oriented technique known as “method chaining,” one procedure’s **return** channel is transferred to a subsequent procedure’s **sigma** channel. The pairing of **return** and **sigma** thereupon gives rise to one function-composition operator. With suitable restrictions (on channel size), **return** and **lambda** channels engender a different function-composition operator. So channels can be used to define operators between procedures which yield new function-like values (i.e., instances of function-like types). In some cases, function-like values defined via inter-function operators can be used in lieu of those instantiated from implemented procedures (although the specifics of this substitutability—an example of so-called “eta (η) equivalence”—varies by language).

These examples represent possible combinations or interconnections (sharing values) between channels, together with semantic restrictions on when such connections are possible. In this chapter, I assume that notations describing these connections and restrictions can be systematized into a “Channel Algebra,” and then used to model programming language conventions and computer code. A basic example of inter-channel aggregation would be how a **lambda** channel, combined with a **return**

channel, associated with one procedure, yields a conventional input/output pairing. One particular channel formation—**lambda+return**, say—therefore models the basic λ -Calculus and, simultaneously, a minimal definition of function-like types. Notionally, a procedure is, in the simplest conceptualization, the unification of an input channel and an output channel—written, say, $\mathfrak{C}_1 \oplus \mathfrak{C}_2$ (with the \oplus possibly holding extra stipulations, like \mathfrak{C}_1 and \mathfrak{C}_2 cannot both be non-empty). So a “channel sum” creates the basic foundation for a procedure, analogous to how input and output graph elements yield the foundations for morphisms in Hypergraph Categories. More complex channel combinations and protocols can then model more complex variations on λ -calculi and on programming language type systems.

3.3.4.2 Channelized type systems

Collectively, to summarize my discussion to this point, I will say that formulations describing channel kinds, their restrictions, and their interrelationships outline a *Channel Algebra*, which expresses how channels combine to describe possible function signatures—and accordingly to describe functional *types*. The purpose of a Channel Algebra is, among other things, to elucidate how formal languages (like programming languages) formulate functions and procedures, and the rules they put in place for inputs and outputs. If χ is a Channel Algebra, a language adequately described by its formulations (channel kinds, restrictions, and interrelationships) can be called a χ -language. The basic λ -Calculus can be described as a χ -language for the algebra defined by a minimal **lambda+return** combination (with **return** channels restricted to at most one element). Analogously, a type system \mathbb{T} is a “ χ -type-system,” and is “closed” with respect to χ , if valid signatures characterized using channel kinds in χ correspond to types found in \mathbb{T} . Types may be less granular than signatures: as a case in point, functions differing in signature only by whether they throw exceptions may or may not be deemed the same type. But a channel construction on types in \mathbb{T} must also yield a type in \mathbb{T} .

I say that a type system is *channelized* if it is closed with respect to some Channel Algebra. Channelized Hypergraphs are then DHs whose type system is Channelized. We can think of channel constructions as operators which combine groups of types into new types. Once we assert that a CH is Channelized, we know that there is a mechanism for describing some Hypergraphs or subgraphs as “procedure implementations,” some of whose hypernodes are subject to kinds of abstraction present in the relevant Channel Algebra. Channel formulae and signatures describe source-code norms which could also be expressed via more conventional Ontologies. So Channel Algebra can be seen as a generalization of (RDF-environment) Source Code Ontology (of the kinds studied, for example, by Refs. [109–114]). Given the relations between RDF and Directed Hypergraphs (despite differences I have discussed here), Channel Algebras can also be seen as adding to Ontologies governing Directed Hypergraphs. Such is the perspective I will take for the remainder of this chapter.

For a Channel Algebra χ and a χ -closed type system (written, say) \mathbb{T}^χ , χ extends \mathbb{T} because function-signatures conforming to χ become types in \mathbb{T} . At the same time, \mathbb{T} also extends χ , because the elements that populate channels in χ have types within \mathbb{T} . Assume that for any type system, there is a partner “Type Expression Language” (TXL) which governs how type descriptions (especially for aggregate types that do not have a single symbol name) can be composed consistent with the logic of the system. The TXL for a type-system \mathbb{T} can be notated as $\mathcal{L}_{\mathbb{T}}$. If \mathbb{T} is channelized, then its TXL is also channelized—say, $\mathcal{L}_{\mathbb{T}^\chi}$ for some χ .

Similarly, we can then develop for Channel Algebras a *Channel Expression Language*, or CXL, which can indeed be integrated with appropriate TXLs. Formal declarations of channel axioms—for example, restrictions on channel sizes, alone or in combination—are examples of terms that should be representable in a CXL. However, whereas the CXL expressions I have described so far elucidate the overall shape of channels—which channels exist in a given context and their sizes—CXL expressions can also add details concerning the *types* of values that can or do populate channels. CXL expressions with these extra specifications then become function signatures, and as such type-expressions in the relevant TXL. A channelized TXL is then a superset of a CXL, because it adds—to CXL expressions for function-signatures—the stipulation that a particular signature does describe a *type*; so CXL expressions become TXL expressions when supplemented with a proviso that the stated CXL construction describes a function-like type’s signature. With such a proviso, descriptions of channels used by a function qualifies as a type attribution, connecting function symbol-names to expressions recognized in the TXL as describing a type.

Some TXL expressions designate function-like types, but not all, since there are many types (`int`, etc.) which do not have channels at all. While a TXL lies “above” a CXL by adding provisos that yield type-definition semantics from CXL expressions, the TXL simultaneously in a sense lies “beneath” the CXL in that it provides expressions for the non-functional types which in the general case are the basis for CXL expressions of functional types, since most function parameters—the input/output values that populate channels—have nonfunctional types. [Section 3.5](#) will discuss the elements that “populate” channels (which I will call “carriers”) in more detail.

In the following sections, I will sketch a Channel Algebra that codifies the graph-based representation of functions as procedures whose inputs and outputs are related to other functions by variegated semantics (semantics that can be catalogued in a Source Code Ontology). With this foundation, I will argue that Channel-Algebraic type representations can usefully model higher-scale code segments (like statements and code blocks) within a type system, and also how type interpretations can give a rigorous interpretation to modeling constructs such as code specifications and “gatekeeping” code. I will start this discussion, however, by expanding on the idea of employing code-graphs—hypergraphs annotated according to a Source Code Ontology—to represent procedure implementations, and therefore to model procedures as instances of function-like types.

3.4 Modeling procedures via channelized hypergraphs

Assuming we have a suitable Source Code Ontology, software procedures can be seen from two perspectives. On the one hand, they are examples of well-formed code graphs: annotated graph structures convey the lexical symbols, input/output parameters (via different “abstractions,” in the sense of λ -abstraction, subject to relevant channel protocols), and calls to other procedures, through which any given procedure’s functionality is achieved. On the other hand, we can see procedures as instances of function-like types, where the types carried in each channel determine the type of the procedure itself, as a functional value. Although these two perspectives are usually mutually consistent, the notion of functional values is more general than procedures which are expressly implemented in computer code. In particular, as I briefly mentioned earlier, sometimes functional values are denoted via inter-function operators (like the composition $f \circ g$) rather than by giving an explicit implementation. We can say that functions defined via operators (like \circ) lack a “function body.”

Going forward, I will generally use the term *procedure* with reference to function-like type instances that are defined *with* function bodies: that is, they are associated with sections of code that supply the procedure’s implementation, and can be represented via code-graphs. I will use the term *function* more generally for instances of function-like types, irregardless of their provenance. In particular, functions are *values*—instances of types in a relevant type-system \mathbb{T} —whereas I will not usually discuss procedures as “values.” On the other hand, code-graphs capture the implementations through which function-like types are (mostly) populated with concrete values.

To model the general maxim that any coding assumptions made (but not verified) by one procedure—say, \mathcal{P}_1 —should be tested by other procedures which call \mathcal{P}_1 , we need a systematic outline capturing the notion of procedures calling other procedures, in the course of their own implementation. Here I propose to model these details via channels and interrelationships between channels. Moreover, channels can be seen as structures on *graphs*, as well as runtime information flows, so that channels are applicable for both static and dynamic program analysis.

One consequence of my graph-oriented approach is that the technical distinctions between procedures and function-values (in general) have to be duly observed. There are some relevant complications appertaining to the general picture of source-code segments instantiating function-like types. I will briefly review these issues now, before pivoting to more macro-scale themes concerning Requirements Engineering via code models.

3.4.1 Initializing function-typed values

Although in general function-typed values are *initialized* from code-graphs that blueprint their implementation, this glosses over several different mechanisms by which function-typed values may be defined:

1. In the simplest case, there is a one-to-one relationship between a code graph and an implemented function (`f`, say). If `f` is polymorphic, in this case, it must be an example of subtype (or “runtime”) polymorphism where the declared types of `f`’s parameters are actually instantiated, at runtime, by values of their subtypes.
2. A different situation (“compile-time” polymorphism) applies to generic code as in C++ templates. Here, a single code-graph generates multiple function bodies, which differ only by virtue of their expected types. For example, a templated `sort` function will generate multiple function bodies—one for integers, say, one for strings, etc. These functions may be structurally similar, but they have different signatures by virtue of working with different types. This means that symbols used in the function-bodies may refer to different functions even though the symbols themselves do not vary between function-bodies (since, after all, they come from the same node in a single code-graph). That is, the code-graphs rely on symbol-overloading for function names to achieve a kind of polymorphism, where one code-graph yields multiple bodies.

In this compile-time polymorphism, symbols are resolved to the proper overload-implementation at compile-time, whereas in runtime polymorphism this decision is deferred until the runtime-polymorphic function is actually being executed. The key difference is that runtime-polymorphic functions are *one* function-typed value, which can work for diverse types only via subtyping—or via more exotic forms of indirection, like using function-pointers in place of function symbols, whereas compile-time-polymorphic (i.e., templated) functions are *multiple* values, which share the same code-graph representation but are otherwise unrelated.

3. A third possibility for producing function-like values is to define operators on function-like types themselves, which transform functional values to other functional values, by analogy to how arithmetic operations transform numbers to other numbers. As will be discussed later, this may or may not be different from initializing functional values via code-graphs. For instance, given the composition operator `o`, `f o g` may or may not be treated as only a convenient shorthand for a code graph spelling out something like `f(g(x))`.
4. Finally, as a special case of operators on functional values, one function may be obtained from another by “Currying,” that is, fixing the value of one or more of the original function’s arguments. For example, the `inc` (“increment”) function which adds `1` to a value is a special case of addition, where the added value is always `1`. Here again, Currying may or may not be treated as a function-value-initialization process different from ones starting from code-graphs.

The differences between how languages may process the *initialization* of function-type values, which I alluded to in (3) and (4), reflect differences in how function-like values are internally represented. We *might* treat all initializations of these values as via code-graphs (in practice, compiled down via an Abstract Syntax Tree or Graph to some Intermediate Representation or byte-code). Suppose we have an `add` function and want to define an `inc` function, as in `int inc(int x){return add(x,1)}`. Even if a

language has a special Currying notation, that notation could translate behind-the-scenes to an explicit function body, like the code at the end of the last sentence. Alternatively, however, a language engine may also note that **inc** is derived from **add** and can be wholly described by a handle denoting **add** (a pointer, say) along with a designation of the fixed value: in other words, $\langle \&\text{add}, 1 \rangle$. Instead of initializing **inc** from a code-graph, the language can represent it via a two-part data structure like $\langle \&\text{add}, 1 \rangle$ —but only if the language *can* represent function-like types’ instances as compound data structures.

Let us assume that a language can always represent *some* functional values, ones that are obtained from code-graphs, via pointers to (or some other unique identifier for) an internal memory area where at least *some* compiled function bodies are stored. The interesting question is whether *all* function-like values are represented in this manner and, in either case, the consequences for the semantics of functional types—semantic issues such as $f \circ g$ composition operators and Currying (and also, as I will argue, Dependent Types).

3.4.1.1 Addressability and implementation

Talk about polymorphism in a language like C++ covers several distinct language features: achieving code reuse by templating on type symbols is internally very different from using virtual methods calls. The key difference—highlighted by the contrast between runtime- and compile-time polymorphism—is that there are some function implementations which actually compile to *single* functions, meaning in particular that their compiled code has a single place in memory and that they may be invoked through function pointers. Conversely, what appears in written code as one function body may actually be duplicated, somewhere in the compiler workflow, generating multiple function-like values. The most common cases of such duplication are templated code as discussed earlier (though there are more exotic options, for example, via C++ macros and/or repeated file `#includes`). Implementations of the first sort I will call “addressable,” whereas those of the second produce multiple addressable values. These concepts prove to be consequential in the abstract theory of types, although for non-obvious reasons.

To see why, consider first that type systems are intrinsically pluralistic: there are numerous details whereby the type system underlying one computing environment can differ from those employed by other environments. So there is no single, universal “Type Expression Language.” One role of any given TXL is to model what its corresponding language recognizes as a type, or—better—a *potential* type. A TXL expression which designates a (unique) type is well-formed if it unambiguously describes a type that *could* exist. Such an expression does not, however, implement the type on its own, or mandate that the type be implemented; it would merely affirm that the type so designated is implementable within the target language.

As a concrete example, consider a type described in English as: “the type inhabited by functions which take, as one parameter, a Unicode string, and, as the second parameter, an unsigned integer less than the length of the string.” A TXL version of

this specification would only be valid if the requirements thereby described can be satisfied, in the target language, via type-checking alone.

For a more in-depth example, if in C++ I assert “**template** **MyList**,” it would then be consistent with a C++-specific TXL to describe a type as **MyList** (assume this will be implemented as a list of integers). However, the type **MyList** is not, without further code, actually implemented. It is a *possible* type because its description conforms to a relevant TXL, but not an *actual* type. If a programmer supplies a templated implementation for **MyList**, then the compiler can derive a “specialization” of the template for a specific **T**—or the programmer can specialize **MyList** on **int** (or any other chosen type) manually. But in either case the actualization of **MyList** will depend on an implementation (either a templated implementation that works for multiple types or a specialization for a single type); this is separate and apart from **MyList** being a valid *expression* denoting a *possible* type.

Templates and specialization add complexity to discussions about types, because compilers may automatically instantiate concrete types from templated code *unless* programmers supply specializations which deviate from the template. As a result, in a local segment of a source file it may be impossible to know whether or not the code concretizing a templated type is automatically generated from a template. Another complication is that compilers may derive *default implementations* of types’ constructors, unless these are coded explicitly. Taking these two considerations together, it can be difficult in a code base to, given a type, find which code-segments yield that type’s constructors.

As an analytic device, here I assume that every implementable type can be associated with a procedure I will call a *co-constructor*, whose role is to wrap constructor-calls in a readily identifiable code body. Co-constructors are “ordinary” procedures in the sense that they are “addressable.” Specifically, addressable procedures have these properties:

1. You can take their address (assuming we are dealing with a language that supports function pointers in the first place).
2. They have a corresponding (possibly templated) location in source code (and therefore a code-graph). For co-constructors, this location can be marked as such—it should be straightforward to identify all co-constructor implementations in a code base.
3. They can be exposed to scripting engines and runtime reflection; so co-constructors enable type-instances to be created via scripts and other runtime-introspection capabilities.

Operationally, co-constructors are similar to *factory procedures* or *object factories* (see, e.g., [115, pp. 32–35, 116, pp. 34–35, 117, 118]), which similarly delegate to constructors but can be used in contexts where constructors cannot; for example, where it is necessary to address the factory through a pointer (note that in C++ you may not take the address of an actual constructor).

Insofar as co-constructors are *addressable*, they provide an indirect mechanism for designating their corresponding type. I will use the term *preconstructor* to mean a

function-pointer holding the address of a co-constructor, or some similar data structure that uniquely identifies a co-constructor. A preconstructor thereby holds a compact value which is associated with exactly one type. A valid preconstructor, in particular, serves as proof that a given type is implemented—it confirms the existence of at least one fully implemented constructor for that type, indicating that the type is *actual* and not just *potential*.

Suppose certification requires that the function which displays the gas level on a car’s dashboard never attempts to display a value above **100** (intended to mean “One Hundred percent,” or completely full). One way to ensure this specification is to declare the function as taking a *type* which, by design, will only ever include whole numbers in the range **(0,100)**. Thus, a type system may support such a type by including in its TXL notation for “range-delimited” types, types derived from other types by declaring a fixed range of allowed values. A notation might be, say, **int(0,100)**, for integers in the **(0,100)** range—or, more generally expressions like **T(V₁,V₂)**, meaning a *type* derived from **T** but restricted to the range spanned by **V₁** and **V₂** (assumed to be values of **T**—notice that a TXL supporting this notation must consequently support some notation of specific values, like numeric literals).

However, merely describing range-delimited types’ desired space of values does not provide a full implementation specification. What should happen if someone tries to construct an **int(0,100)** value with the number, say, **101**? What about with values taken from an external source, like a web API, where it cannot be proved that the values fall in the proper range? These questions point to implementation choices that transcend formal designations. This is why TXL expressions should be seen as just articulating *potential* types, because bringing types into actuality will usually call for engineering choices that transcend type theory per se. Once types *are* implemented, co-constructors serve as tangible witness to types’ actualization, and preconstructors are convenient proxies referring to those types.^o

Reasoning abstractly about functions and types needs to be differentiated from reasoning about available, implemented types (and functions defined on them). Consider function pointers: what is the address of $f \circ g$ if that expression is interpreted in and of itself as evaluating to a functional value?^p This suggests that a composition operator does not work in function-like types quite like arithmetic operators in numeric types (which is not unexpected insofar as functional values, internally, are more like pointers than numbers-with-arithmetic).^q To put

^oSimilar issues are sometimes addressed by a *modal* type theory (cf., e.g., [119]) where (in one interpretation) a *logical* assertion about a type may be *possible* but not necessary (the modality ranging over “computing environments,” which act like “possible worlds”).

^pIn my perspective here, $f \circ g$ may be a *plausible* value, but it is not an *actual* value without being implemented, whether via a code graph (spelling out the equivalent of $\lambda x.fgx$) or some indirect/behavioral description (analogous to **inc** represented as $\langle \&add, 1 \rangle$).

^qOf course, languages are free to implement functions behind the scenes to expand (say) $f \circ g$, but then $f \circ g$ is just syntactic sugar (even if its purpose is not just to neaten source code, but also to inspire programmers toward thinking of function-composition in quasi-arithmetic ways).

it differently, an **address-of** operator *may* be available for $f \circ g$ if it is available for **f** and **g**, but this depends on language design; it is not an abstract property of type systems.

A similar discussion applies to “Currying”—the proposal that types $t_1 \rightarrow t_2 \rightarrow t_3$ and $t_1 \rightarrow (t_2 \rightarrow t_3)$ are equivalent, in that fixing one value as argument to a binary function yields a new unary function. Again, since the Curried function is not necessarily implemented, there is a *modal* difference between $t_1 \rightarrow t_2 \rightarrow t_3$ and $t_1 \rightarrow (t_2 \rightarrow t_3)$. Languages *may* be engineered to Curry silently any function on demand, but purported $t_1 \rightarrow t_2 \rightarrow t_3$ and $t_1 \rightarrow (t_2 \rightarrow t_3)$ equivalence is not a *necessary* feature of type systems.

To the extent that both mathematical and programming concepts have a place here, we find a certain divergence in how the word “function” is used. If I say that “there exists a function from t_1 to t_2 ,” where t_1 and t_2 are (not necessarily different) types, then this statement has two possible interpretations. One is that, mathematically, I can assume the existence of a $t_1 \Rightarrow t_2$ mapping by appeal to some sort of logic; the other is that a $t_1 \Rightarrow t_2$ function actually exists in code. This is not just a “metalanguage” difference projected from how the discourse of mathematical type theory is used to different ends than discourses about engineered programming languages, which are social as well as digital-technical artifacts. Instead, we can make the difference exact: when a function-value is keyed to a procedure, it is bound to a segment of code subject to analysis and to alternative representations (such as code graphs).

Since co-constructors are *addressable*, they cannot—at least not within the framework I have discussed thus far—be “temporary” function-values analogous to $f \circ g$. This means that *types* cannot be temporary values. More precisely, a type system may be constrained by the proposition that *no type can be created* whose co-constructors would have to be temporary values—or, to put it differently, no type can be created whose co-constructors are not procedures that can be mapped to source-code function-bodies (and thereby to code-graphs).

Notice that co-constructors, then, are not just function-like values; co-constructors have to be in that subspace of function-like values initialized via code-graphs, rather than via some quasi-arithmetic inter-function operator like $f \circ g$. This then limits what we can do with Dependent Types, typestate, and other “expressive” type mechanisms. I will call this the “metaconstructor” problem: insofar as co-constructors are function-like values, they (in principle) need their own constructors—call these “metaconstructors.” We can stipulate that metaconstructors—constructors of co-constructors—have to be derived from code graphs (they cannot be temporary values), but this renders certain advanced type-theoretic features inaccessible to our applied type systems. Conversely, we can accept the idea of constructors being (potentially) temporary values, but this interferes with preconstructors being referential proxies for types themselves (unless types also are, potentially, temporary constructs, which creates a new set of problems). I will now explain this choice in greater depth.

3.4.2 Dependent types and co-constructors

To see why the metaconstructor problem determines how extensively Dependent Types are supported in a type system, consider a variation on the range **(0, 100)** type. In lieu of a fixed range, consider a procedure taking a (variable) **T**-range **(r)** and a number **x** which must be in that range. Here **x** “depends” on **(r)**—its type is **(r)** seen as its own **T(V₁, V₂)** type—so **x** can vary among many range-types, only being fixed at runtime. Defining a *type* for procedures meeting those *specifications* is a classic problem of Dependent Type theory.

Using the **(r)**-type as before, the type of **f**'s second parameter would then be **T** restricted to the **(r)** interval, but here **(r)** is not fixed in **f**'s declaration but rather passed into **f** as a parameter. Unless we know a priori that only a specific set of **(r)s** in the first parameter will ever be encountered, the compiler has to be prepared for **x** being assigned any one of many different range types, depending on the **f**'s first argument. In particular, the compiler cannot know ahead of time which constructor to call for **x**. More precisely, it is impossible for the compiler to have *separate* constructors for millions of possible range types. Instead, the compiler must either “create” a constructor “on the fly” or else have some generic constructor which services many range-types, but then requires extra information to establish which range is desired.

Assuming we use co-constructors to wrap constructors, these two options for compiler writers correspond to the choice of *either* creating ad hoc co-constructors *or* designing co-constructors as a compound data structure. We could certainly write a function that takes a range and a value and ensures that the value fits the range—perhaps by throwing an exception if not, or mapping the value to the range's closest point. Such a function would provide common functionality for a family of constructors each associated with a given range. But a function (**Cf**, say) providing “common functionality” for value constructors is not necessarily itself a value constructor.^r To treat such a function as a *real* value constructor, we would have to add contextual modifiers: **Cf** is a value constructor for range-type **(r)** in the presence of a **T**-pair to specify **(r)** at runtime. The co-constructor for a range type **T(r)** is accordingly the “common functionality” base function *plus* **T**'s passed to it—some sort of $\langle \&Cf, r_1, r_2 \rangle$ compound data structure, again by analogy to **inc** and **$\langle \&add, 1 \rangle$** (see footnote p). Here again, though, the co-constructor is a temporary data structure, created on the fly to model the desired value constructor for an **x** whose type (and therefore whose constructor) is not known until runtime. I contend, on examples like these, that Dependent Typing for a type system **T** is thus logically equivalent to the possibility of **T** co-constructors being temporary values.

But value constructors (and by extension co-constructors) are not just any function-value: they have a privileged status vis-à-vis types, and may be invoked whenever an appropriately typed value is used. Many constructors are called

^rHere I say “value constructor” to clarify that I am not commenting on *type constructors*, which derive specialized types from generic ones.

behind-the-scenes: in C++, the standard function-call mechanism is “pass by value,” wherein values are *copied* when passed between procedures; but any copy can potentially invoke a so-called “copy constructor.” Indeed, programmers use certain constructors as “hooks” to silently insert logic into normal program flow (usually this is to make complex types behave like built-in-types from client code’s point of view). Allowing large type families (like one type for each `int` or each two-number range `(r)`—similar to “inductive families” as discussed by Edwin Brady in the context of the Idris language [120, p. 12])—could easily conflict with user-defined constructor overrides: users (meaning, in this context, library developers) would need not only to write their own (e.g., `copy`) constructors, but also to hook into a complex run-time mechanism for creating constructors ad hoc as temporary values. Conversely, forcing co-constructors to be addressable prohibits “large” type families—like types indexed over other (non-enumerative) types (see, e.g., [121, p. 4])—at least as *actual* types. This apparently precludes full-fledged Dependent Types, since dependent-typed values invariably require in general some extra contextual data—not just a function-pointer—to designate the desired value constructor at the point where a value, attributed to the relevant dependent type, is needed. It may be infeasible to add the requisite contextual information at every point where a dependent-typed value has to be constructed—unless, perhaps, a description of the context can be packaged and carried around with the value, sharing the value’s lifetime.

As I will now review, this analysis in the realm of Dependent Types carries over into *typestate*, which is another mechanism intended to model coding requirements via type-checkable specifications.

3.4.2.1 Dependent types and typestate

Typestates are finer-grained classifications than types. A canonical example of type-state is restricting how functions are called which operate on files. A single “file” type actually covers several cases, including files that are open or closed, and even files that are nonexistent—they may be described by a path on a file system which does not actually point to a file (perhaps in preparation for creating such a file). Instead of *one* type covering each of these cases, we can envision *different* types for nonexistent, closed, or open files. With these more detailed types, constraints like “don’t try to create an already-existing file” or “don’t try to modify a closed or nonexistent file” are enforced by type-checking.

While this kind of gatekeeping is valuable in theory, it raises questions in practice. Reifying “cases”—that is, *typestates* like open, closed, or nonexistent—to distinct *types* implies that a “file” value can go through different types between construction and destruction. If this is literally true, it violates the convention that types are an intrinsic and fixed aspect of typed values. It is true that, as part of a type cast, values can be reinterpreted (like treating an `int` as a `float`), but this typically assumes a mathematical overlap where one type can be considered as subsumed by a different type for some calculation, *without this changing anything*: any integer is equally a ratio with unit denominator, say. “Casting” a closed file to an open one is the opposite effect, using disjunctions between types to capture the fact that state *has*

changed; to capture a trajectory of states for one value—which must then have different types at different times, since this is the whole point of modeling successive states via alternations in type-attribution.

An alternative interpretation is that the “trajectory” is not a single mutated value but a chain of interrelated values, wherein each successive value is obtained via a state-change from its predecessor. But a weakness of this chain-of-values model is that it assumes only one value in the chain is currently correct: a file cannot be both open and closed, so if one value with type “closed file” is succeeded by a different value with type “opened file,” the latter value will be correct only if the file was in fact opened, and the former otherwise—but a compiler cannot know which is which, a priori. Or, instead of a “chain” of differently typed values we can employ a single general “file” type and then “cast” the value to an “open file” type when a function needs specifically an *open* file, and so forth. The effect in that case is to insert the cast operator as a “gatekeeper” function preventing the function receiving the casted value from getting nonconformant input. Again, though, the compiler cannot make any assumptions about whether the “casts” will work (e.g., whether the attempt to open a file will succeed).

In short, typestate forces us to modify some basic assumptions about the relationship between types and values: either values can change types mid-stream, or a lexical scope can subsume a sequence of value “holders” which share the same symbol-name (and maybe the same type) but differ in state (some holding values unrelated to actual program state). Both options upend normal programming expectations. This situation can be juxtaposed with the “metaconstructor problem,” that is, how Dependent Types force a rethink on basic value-constructor theory.

A good real-world example of the overlap between Dependent Types and type-state (also grounded on file input/output) comes from the “Dependent Effects” tutorial from the Idris (programming language) documentation [122]:

A practical use for dependent effects is in specifying resource usage protocols and verifying that they are executed correctly. For example, file management follows a resource usage protocol with ... requirements [that] can be expressed formally in [Idris] by creating a **FILE_IO** effect parameterised over a file handle state, which is either empty, open for reading, or open for writing. In particular, consider the type of [a function to open files]: This returns a **Bool** which indicates whether opening the file was successful. The resulting state depends on whether the operation was successful; if so, we have a file handle open for the stated purpose, and if not, we have no file handle. By case analysis on the result, we continue the protocol accordingly. ...If we fail to follow the protocol correctly (perhaps by forgetting to close the file, failing to check that open succeeded, or opening the file for writing [when given a read-only file handle]) then we will get a compile-time error.

So how does Idris mitigate the type-vs.-typestate conundrum? Apparently the key notion is that there is one single **file** type, but a more fine-grained type-*state*, and, moreover, an *effect system “parameterized over” these typestates*. In other words, the *effect* of **file** operations is to modify *typestates* (not types) of a **file** value.

Moreover, Dependent Typing ensures that functions cannot be called sequentially in ways that “violate the protocol,” because functions are prohibited from having effects that are incompatible with the potentially affected values’ current states. This elegant syntheses of Dependent Types, typestate, and Effectual Typing brings together three of the key features of “fine-grained” or “very expressive” type systems.

But the synthesis achieved by Idris relies on Dependent Typing: typestate can be enforced because Idris functions may support restrictions which *depend* on values’ current typestate to satisfy effect-requirements in a type-checking way. In effect, Idris requires that all possible variations in values’ unfolding typestate are handled by calling code, because otherwise the handlers will not type-check. An analogous tactic in C++ would be to provide an “open file” function only with a signature that takes two callbacks, one for when the `open` succeeds and a second for when it fails (to mimic the Idris tutorial’s “case analysis”). But that C++ version still requires convention to enforce that the two callbacks behave differently: via Dependent Types, Idris can confirm that the “open file” callback, for example, is only actually supplied as a callback for files that have indeed been opened. A better C++ approximation to this design would be to cast files to separate types—not only typestates—after all, but only when passing these values to the callback functions (or, as I will discuss later, using a “passkey” to vouch that a callback’s file argument *can* be thus cast).

In the case of Idris, Dependent Types are feasible because the final “reduction” of expressions to evaluable representations occurs at runtime. In the language of the Idris tutorial:

In Idris, types are first class, meaning that they can be computed and manipulated (and passed to functions) just like any other language construct. For example, we could write a function which computes a type [and] use this function to calculate a type anywhere that a type can be used. For example, it can be used to calculate a return type [or] to have varying input types.

More technically, Tejíšcák and Brady [123, p. 1] elaborate that

Full-spectrum dependent types ...treat types as first-class language constructs. This means that types can be built by computation just like any other value, leading to powerful techniques for generic programming. Furthermore, it means that types can be parameterised on values, meaning that strong, explicit, checkable relationships can be stated between values and used to verify properties of programs at compile-time. This expressive power brings new challenges, however, when compiling programs. ...The challenge, in short, is to identify a phase distinction between compile-time and run-time objects. Traditionally, this is simple: types are compile-time only, values are run-time, and erasure consists simply of erasing types. In a dependently typed language, however, erasing types alone is not enough.

To summarize, Idris works by “erasing” some, but not all, of the extra contextual detail needed to ensure that dependent-typed functions are used (i.e., called) correctly (see also [124, 125, p. 195]). This means that a lot of contextual detail is *not* erased; Idris provides machinery to join executable code and user specifications onto *types* so that they take effect whenever affected types’ values are constructed or passed to functions.

Despite a divergent technical background, the net result is arguably not vastly different from an Aspect-Oriented approach wherein constructors and function calls are “pointcuts” setting anchors upon source locations or logical run-points, where extra code can be added to program flow (see, e.g., [126–128]). Recall my contrast of “internalist” and “externalist” paradigms, sketched at the start of this chapter: Aspect-Oriented Programming involves extra code added by external tools (that “modify” code by “weaving” extra code providing extra features or gatekeeping). Implementations like Idris pursue what often are in effect similar ends from a more “internalist” angle, using the type system to host added code and specifications without resorting to some external tool that introduces this code in a manner orthogonal to the language proper. But Idris relies on Haskell to provide its operational environment; it is not clear how Idris’s strategies (or those of other Haskell and ML-style Dependent Type languages) for attaching runtime expressions to type constructs would work in an imperative or Object-Oriented environment, like C++ as a host language.

3.4.2.2 Simulating dependent types with preconstructors

Because Dependent Types and typestate recognize fine-grained requirements on which values may be passed to which functions, it might seem as if they are a logical continuation of the telos toward granular data modeling. If our goal is to provide the most expressive data models possible within the bounds of computational tractability (I will return to this in the conclusion), we should certainly allow for Dependent Types and any other constructions which logically imply them (essentially, any formulation wherein types or their constructors are ad hoc temporary values).

However, Dependent Types have the technical consequence of leaving pre-runtime values (or whatever construct we recognize as “holding” or “carrying” values) either *without* types, or with *different* types than they have at runtime. In this case it becomes difficult to query code *outside* runtime, which arguably *subtracts* expressiveness from the framework. In short, we are free to explore some foundation for emulating Dependent Types *without* giving up on static reflection; the resulting system would not necessarily be expressively lesser than a \mathbb{T} with full-fledged Dependent Type support.

An elegant compromise might seem to be allowing each value to have two potential type attributions—one “static” and one at runtime, potentially changing with each call to the surrounding function-body. After all, it is often consequential that a value of type t_1 may be coerced to, or reinterpreted as an instance of, t_2 . This means that a t_1 ’s specific value is consistent with also being a t_2 : it falls into the “space” where t_1 and t_2 overlap; or there is an available conversion that creates a t_2 from

the t_1 . However, this convertibility is usually a factor when (using this chapter's terms) the t_1 is transferred to a channel in a place for a t_2 . Bear in mind that types are not "sets"; it is not as if we can regard two types as indistinguishable within the collection of values where they can be interconverted. In my treatment, types are manifest first and foremost as recipes for values to be "handed off" between channels.

So, a "second" or "dynamic" type for some value would only be operational if it corresponds to the "static" type in some receiving channel (this is how subtyping works). But then we are no closer to dealing with "temporary" types, because the "metaconstructor" problem simply reappears within the new channel. This is not to rule out the *receiving* context having its own duality of static and dynamic types, where the hand-off has to match requirements on both static-to-static and dynamic-to-dynamic type-pairs. In that case, however, the dynamic types are not really "second" types to which the initial values are cast; they are more like logical preconditions which must be satisfied at both ends of a channel-transfer.

Indeed, type-attributions do two different things: first they establish that some value is suitable for transfer between procedures, but, second, they affirm certain predicates vis-à-vis that value. With dynamically recognized, temporarily "constructed" types we are actually dealing with the second salience: proof that values *can* be attributed to some (maybe temporary and context-specific) type establishes facts about that value. But in this case we are not interested in using that second type as the infrastructure for a carrier-transfer; we are instead trying to employ the type-attribution *logically* as a transfer precondition. Perhaps a credible analogy is the post office only accepting boxes within a certain size and weight (manifesting logistical constraints in how the boxes can be handled) versus only accepting boxes which their sender certifies not to contain dangerous contents (establishing contractual rules that transcend the raw logistics of transporting packages). Channel packages-to-complexes (I will explain this terminology in the next section) is an analogous "binary" transporting which can be factored into an underlying digital logistics and a more nuanced accounting of package/complex compatibility, wherein we desire to reject certain package/complex pairs that ordinary type systems would allow (e.g., an index parameter on the package side incompatible with the size of a list parameter). The problem is how to achieve semantics modeling Dependent Types within a framework that situates type theory itself in a channel-transfer (and graph-oriented) context: types only "exist" insofar as they regulate inter-channel handoffs. A given type therefore only exists if there are capabilities in the system to test package/complex matches against the proposed type's logical posits.

The solution I suggest to effectuate this compromise involves using preconstructors as witnesses that a given value construction *could* be performed—so that a given value (or values) is/are *logically consistent* with being construed as instance of some (perhaps ad hoc) type, but are not *literally* assigned to that type. The preconstructor can then be passed in to functions as an extra parameter, which when valid (e.g., when not a null function pointer) vouches for the co-construction

it references being permissible. That is, the preconstructor become a “passkey” parameter returned from a gatekeeper procedure and then passed on as evidence of the gatekeeping validation.

As a concrete example, suppose a procedure requires two numbers where the second is greater than the first (the inverse of the systolic-over-diastolic mandate): **f(x, y)** where **x < y**. How can we express the **x < y** condition within **f**'s signature, assuming the signature can only express semantics pertinent to **f**'s type attribution? On the face of it, we know that the desired “increasing” condition is equivalent to **y** having a type like **range_gt**—a range bounded (only) from below—where this “**x**” is the **x** preceding **y** in **f**'s signature. But using such directly as **y**'s type-attribution means that from the perspective of **f**'s own type-attribution, **y** does not have a single, fixed type; its type varies according to the value of **x**. Here again we encounter a “metaconstructor” problem: in order for the **x < y** condition to be modeled *directly* by **y**'s type-attribution we would need the constructor for **y**'s value-constructor to be some operation that produces a temporary function-value—not simply the compilation of a code-graph to an addressable, non-temporary implementation.

These issues go away if, instead of working with a function taking *two* integers, we instead consider a function taking *one* value which is a monotone-increasing pair (something like **int f(mi_pair pr)**). A type like **mi_pair**, based on ordered pairs **x, y** of **ints**, solves the metaconstructor problem for **y** because **x** and **y** are no longer distinct **f** parameters with distinct value-constructors; they are subsumed into one pair, whose own value-constructor can check the **x < y** condition. The *requirements* for the original (two-valued) **f** may then be *described* as **x** and **y** being convertible into a pair **pr** which is an instance of **mi_pair** (so that **x < y**). This *description* is not a *type*, but elevating the description to type level can be at least approximated with a wrapper like **int f(int x, int y){return f(x, y, mi_pair(x, y));}**, so **f** when used as **f(x, y)** will silently call the **mi_pair** constructor. This is only approximate because it allows anomalies like **f(x, y, mi_pair(0, 1))**—taking **mi_pairs** on anything *but* **x** and **y** defeats the purpose—but at least we can approximate a Dependent Type signature with a passkey protocol that is not difficult to enforce via calling conventions (client code should never call the three-argument form directly, which could be sequestered to a file-scoped or private member function).

Now, notice that we do not actually need the third argument; we just want to know that the **mi_pair** constructor *would* accept the **x, y** pair. So we can replace the *actual* **mi_pair** constructor with a *preconstructor* that *could* be used as a factory for **mi_pair** instances if needed, but can *also* serve to certify that a certain set of arguments (here a pair of numbers) meets the logical preconditions which an actual constructor would check off.

For this to work, the **mi_pair** type would need to be implemented with a static procedure that returns a valid preconstructor for valid inputs—plus, assuming the preconstructor/co-constructor pattern I am advocating, a co-constructor whose address would be the basis of the preconstructor value. These are obviously not features of C++ (or any other language I know of) but could readily be an implementation norm for data types used in a safety-conscious project. In effect, consistent use

of preconstructors for fine-grained types is one strategy for siting gatekeeping code broadly throughout a code base.

Further discussion of preconstructors is outside the scope of this chapter, but concrete examples of range-checking via preconstructor passkeys can be found in the demo. Here I will make the further point that—if we accept a Channel Algebra that expands beyond present programming languages—we can move preconstructor passkeys to a separate channel, thereby approximating Dependent Types more eloquently. Co-constructors may be identified via a dedicated co-constructing channel—**coconstruct**—which signals that a return value is not *any* procedure returning the associated type, but a constructing procedure which is part of the type’s interface and helps to demarcate its space of values. A **coconstruct** channel paired with a special **preconstruct** channel, for preconstructor passkeys, provides a meta-model wherein Dependent Types, typestate, and many effect-systems can be reasonably encoded.

This last case also points to how a theory of channels adds semantic expressiveness to code models: we can achieve via descriptions of inter-procedure information flows—including distinguishing distinct roles such as passkeys versus ordinary parameters, and constructing returns versus ordinary procedures happening to return a given type—a semantic exactitude that is implementationally harder (from a language engineering perspective) to achieve directly within a type system. Channel Algebras are not limited to channels actually recognized by existing languages—they could be the basis for new languages, and/or new analytic tools isolating patterns in existing code. With this flexibly channels can be lifted into a construct recognized within data and code modeling paradigms—as well as an added structural layer within hypergraphs—in general. These possibilities may become clearer as I present a theory of channels in more detail in the next section.

3.5 Channels and carriers

Suppose one procedure calls a second. From a high-level perspective, this has several consequences that can be semantic-graph represented—among others, that the calling procedure depends on an implementation of the callee being available—but at the source code level the key consequence is that a node representing source tokens which designate functional values enters into different semantic relations (modeled by different kinds of edge-annotations) than nodes marking other types of values and literals. Suppose we have an edge-annotation that x is a value passed to f ; this graph is only semantically well-formed if f ’s representatum has functional type (by analogy to the well-formedness criteria of $\lambda x.fx$).

This motivates the following: suppose we have a Directed Hypergraph, where the nodes for each hyperedge represent source-code tokens (specifically, symbols and literals). Via the relevant Source Code Ontology, we can assert that certain edge-annotations are only possible if a token (in subject or object position) designates

a value passed to a function. From the various edge-annotation kinds which meet this criteria, we can define a set of “channel kinds.”

This implicitly assumes that symbols “hold” values; to make the notion explicit, I will say that symbols are *carriers* for values. Carriers do not necessarily hold a value at every point in the execution of a program; they may be “preinitialized,” and also “retired” (the latter meaning they no longer hold a meaningful value; consider deleted pointers or references to out-of-scope symbols). A carrier may pass through a “career” from preinitialized to initialized, maybe then changing to hold “different” values, and may then be retired.^s I assume each carrier is associated with a single type throughout its career, and can only hold values appropriate for its type.^t

Via carrier *careers* we then have a ready account of the transformations effectuated by a procedure: in particular, **return** or **exception** (or other output channels’) carriers transition from, before the procedure begins, being preinitialized, to, after it ends, being initialized (depending on how the procedure exits). Input carriers may also be modified (i.e., undergo a state-change). As such, procedures result in (*potential*) changes to the state of those carriers within their signature channel-complex (and other carriers sharing values with them). *Morphisms* in carrier-states thereby play a logical role akin to *reduction* in λ -calculi.

In short, *carriers* embody the contrast between abstract or mathematical type theory and practical languages’ type systems. Instead of the (still rather abstract and circular) notion of a *typed value*—an instance of a type—we can focus on carriers which are tangible elements of source code and also (during runtimes) binary resources. Carriers evince different states; in those states where they hold a concrete value, carriers play a conceptual role analogous to type-instances in formal type theory. On the other hand, carriers can have other states which are orthogonal to type systems: carriers holding *no value*, for example, are different than carriers holding values of a “null” type.

With this being the basic idea, I will now consider carrier operations in more detail, before expanding on the theory of carriers by considering how carriers group into channels.

^sBecause “uninitialized” carriers and “dangling pointers” are coding errors, within “correct” code, carriers and values are bound tightly enough that the whole carrier/value distinction might be considered an artifact of programming practice, out of place in a rigorous discussion of programming languages (as logicomathematical systems, in some sense). But even if the “career” of symbols is unremarkable, we cannot avoid in some contexts—within a debugger and/or an **IDE** (Integrated Development Environment, software for writing programs), for example—needing to distinguish formally the carrier from the value which it holds, or recognize that carriers can potentially be in a “state” where, at some point in which they are relevant for code analysis or evaluation, they do not yet (or do not any longer) hold meaningful values. Consequently, the “trajectory” of carrier “lifetime”—from being declared, to being initialized, to falling “out of scope” or otherwise “retired”—should be integrated into our formal inventory of programming constructs, not relegated to an informal “metalanguage” suitable for discussing computer code as practical documents but not as formal systems.

^tThe variety of possible careers for carriers is not directly tied to its type: a carrier which cannot change values (be reinitialized) is not necessarily holding a **CONST**-typed value.

3.5.1 Carrier transfers

In this theory, carriers are the basic means by which values are represented within computer code, including during communications between different parts of code source (such as calling a procedure). The “information flow” modeled by a function-call includes values held by carriers at the function-call site being transferred to carriers at the function-implementation site. This motivates the idea of a “transfer” of values between carriers, a kind of primitive operation on carriers, linking disparate pieces of code. It also illustrates that the symbols used to name function parameters, as part of function signatures, should be considered “carriers” analogous to lexically scoped symbols.

Taking this further, we can define a *channel* as a list of carriers which, by inter-carrier transfers, signify (or orchestrate) the passage of data into and out of function bodies.^u I will use the notation \leftrightarrow to represent intercarrier transfer: let c_1 and c_2 be carriers, then $c_1 \leftrightarrow c_2$ is a transfer “operator” (note that \leftrightarrow is non-commutative; the “transfer” happens in a fixed direction), marking the logical moment when a value is moved from code-point to code-point. The \leftrightarrow is intended to model several scenarios, including “forced coercions” where the associated value is modified. Meanwhile, without further details a “transfer” can be generalized to *channels* in multiple ways. If c_1 and c_2 are carriers which belong to two channels (χ_1, χ_2) , then $c_1 \leftrightarrow c_2$ elevates to a transfer between the channels—but this needs two indices to be concrete: the notation has to specify which carrier in χ_1 transfers to which carrier in χ_2 . For example, consider the basic function-composition $f \circ g$: $(f \circ g)(x) = f(g(x))$. The analogous “transfer” notation would be, say, $g:\text{return}_1 \leftrightarrow f:\text{lambda}_1$: here the first carrier in the **return** channel of **g** transfers to the first carrier in the **lambda** channel of **f** (the subscripts indicate the respective positions).

Most symbols in procedure code (so corresponding nodes in a code graph) accordingly represent carriers, which are either passed in to a function or lexically declared in a function body. Assume each function body corresponds with one lexical scope which can have subscopes (the nature of these scopes and how they fit in graph representation will be addressed later in this section). The *declared* carriers are initialized with values returned from other functions (perhaps the current function called recursively), which can include constructors that work on literals (so, the carrier-binding in source code can look like a simple assignment to a literal, as in `int i = 0`). In sum, whether they are passed *to* a function or declared *in* a function, carriers are only initialized—and only participate in the overall semantics of a program—insofar as they are passed to other functions or bound to their return values.

Furthermore, both of these cases introduce associations between different carriers in different areas of source code. When a carrier is passed *to* a function, there is a corresponding carrier (declared in the callee’s signature) that receives the former’s

^uNote that this usage varies somewhat from process calculi, where a channel would correspond roughly to what is here called a single carrier; I assume channels in the general case are composed of multiple carriers (see, e.g., Restall [129], Luo [130], or Toninho et al. [131]).

value: “calling a function” means transferring values between carriers present at the site of the function call to those present in the function’s implementation. Sometimes this works in reverse: a function’s return may cause the value of one of its carriers to be transferred to a carrier in the caller (whatever carrier is bound to the caller’s return value).

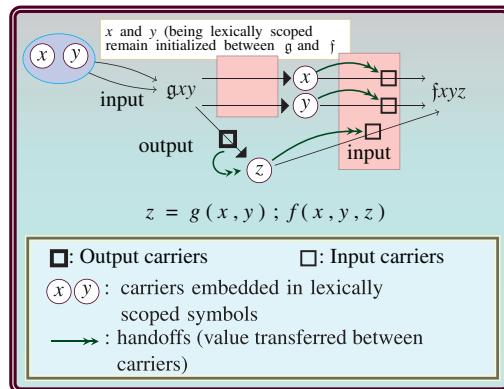
Let c_1 and c_2 be two carriers. The \rightsquigarrow operator (representing a value passed from c_1 to c_2) encompasses several cases. These include the following:

1. Values transfer directly between two carriers in one scope, like $a = b$ or $a := b$.
2. A value transferred between one carrier in one function body when the return value of that function is assigned to a carrier at the call site, as in $y = f(x)$ when f exits with **return 5**, so the value **5** is transferred to y .
3. A value transferred between a carrier at a call-site and a carrier in the called function’s body. Given $y = f(x)$ and f declared as, say, **int f(int i)**, then the value in carrier x at the call-site is transferred to the carrier i in the function body. In particular, every node in the called function’s code-graph whose vertex represents a source-code token representing symbol i then becomes a carrier whose value is that transferred from x .
4. A value transferred between a **return** channel and either a **lambda** or **sigma** channel, as part of a nested expression or a “chain of method calls.” So in $h(f(x))$, the value held by the carrier in f ’s **return** channel is transferred to the first carrier in h ’s **lambda**. An analogous **return** \rightsquigarrow **sigma** transfer is seen in code like $f(x).h()$: the value in f ’s **return** channel becomes the value in h ’s **sigma**, that is, its “**this**” (we can use \rightsquigarrow as a notation between *channels* in this case because we understand the Channel Algebra in force to restrict the size of both **return** and **sigma** to be at most one carrier).

Let $c_1 \rightsquigarrow c_2$ be the special case of \rightsquigarrow corresponding to item (3): a transfer effectuated by a function call, where c_1 is at the call site and c_2 is part of a function’s signature. If f_1 calls f_2 then c_1 is in f_1 ’s context, c_2 is in f_2 ’s context, and c_2 is initialized with a copy of c_1 ’s value prior to f_2 executing. A *channel* then becomes a collection of carriers which are found in the scope of one function and can be on the right-hand side of an $c_1 \rightsquigarrow c_2$ operator.

To flesh out Channels’ “transfer semantics” further, I will refer back to the model of function-implementations as represented in code graphs. If we assume that all code in a computer program is found in some function-body, then we can assume that any function-call operates in the context of some other function-body. In particular, any carrier-transfer caused by a function call involves a link between nodes in two different code graphs (I set aside the case of recursive functions—those which call themselves—for this discussion).

Analysis of value-transfers is particularly significant in the context of Source Code Ontologies and RDF or Directed Hypergraph representations of computer code. This is because code-graphs give us a rigorous foundation for modeling computer programs as sets of function-implementations which call one another. Instead of abstractly talking about “procedures” as conceptual primitives, we can

**FIG. 3.3**

Visualizing carrier transfers (“Handoffs”)

see procedures as embodied in code-graphs (and function-values as constructed from them). Fig. 3.3, for example, shows how graph constructions can track the flow of carriers and values between procedures: this graph-modeling use-case (with additional illustrations) is discussed in greater detail within the demo’s documentation. “Passing values between” procedures is then explicitly a family of relationships between nodes (or hypernodes) in disparate code-graphs, and the various semantic nuances associated with some such transfers (e.g., type casts) can be directly modeled by edge-annotations. Given these possibilities, I will now explore further how the framework of *carriers* and *channels* fits into a code-graph context.

3.5.1.1 Channel groups and code graphs

For this discussion, assume that f_1 and f_2 are implemented functions with code graphs Γ_1 and Γ_2 , respectively. Assume furthermore that some statement or expression in f_1 involves a call to f_2 . There are several specific cases that can obtain: the expression calling f_2 may be nested in a larger expression; f_2 may be called for its side-effects alone, with no concern to its return value (if any); or the result of f_2 may be bound to a symbol in f_1 ’s scope, as in $y = f(x)$. I will take this third case as canonical; my discussion here extends to the other cases in a relatively straightforward manner.

A statement like $y = f(x)$ has two parts: the expression $f(x)$ and the symbol y to which the results of f are assigned. Assume that this statement occurs in the body of function f_1 ; x and y are then symbols in f_1 ’s scope and the symbol f designates (or resolves to) a function which corresponds to what I refer to here as f_2 . Assume f_2 has a signature like `int f(int i)`. As such, the expression $f(x)$, where x is a carrier in the context of f_1 , describes a *carrier transfer* according to which the value of x gets transferred to the carrier i in f_2 ’s context.

I will say that f_2 ’s signature represents a channel “complex”—which, in the current example, has a **lambda** channel of size one (with one carrier of type `int`) and a

return channel of size one (f_2 returns one **int**). Considered in the context of carrier-transfers between code graphs, a channel complex may be regarded as a description of how two distinct code-graphs are to be connected via carrier transfers. When a function is called, there is a channel group which I will call a *package* that supplies values to the channel *complex*. In the concrete example, the statement **y = f(x)** is a *call site* describing a channel *package*, which becomes connected to a function implementation whose signature represents a channel *complex*: a collection of transfers $c_1 \leftrightarrow c_2$ together describe an overall transfer between a *package* and a *complex*.

More precisely, the **f(x)** example represents a carrier transfer whose target is part of f_2 's **lambda** channel, which we can notate $c_1 \leftrightarrow^{\text{lambda}} c_2$. Furthermore, the full statement **y = f(x)** shows a transfer in the opposite direction: the value in f_2 's **return** channel is transferred to the carrier **y** in the *package*. This relation, involving a return channel, can be expressed with notation like $c_2 \leftrightarrow^{\text{return}} c_1$. The syntax of a programming language governs how code at a call site supplies values for carrier transfers to and from a function body: in the current example, binding a call-result to a symbol always involves a transfer from a **return** channel, whereas handling an exception via code like **catch(Exception e)** transfers a value from a called function's **exception** channel. The syntactic difference between code that takes values from **return** and **exception** channels, respectively, helps reinforce the *semantic* difference between exceptions and “ordinary” returns. Similarly, method-call syntax like **obj.f(x)** visually separates the values that get transferred to a “**sigma**” channel (**obj** in this case) from the “ordinary” (**lambda**) inputs, reinforcing Object semantics.

To consolidate the terms I am using: we can interpret both function *signatures* and *calls* in terms of channels. Both involve “carrier transfers” in which values are transferred *to* or *from* the channels described by a function signature. I will say that channels are *convoluted* if there is a potential carrier-transfer between them. The distinction between procedures’ “inputs” and “outputs” can be more rigorously stated, with this further background, as the distinction between channels in function signatures which receive values *from* carriers at a call site (inputs), and those *from which* values are obtained as a procedure has completed (outputs).

A Channel Expression Language (CXL) can describe channels both in signatures and at call-sites. The aggregation of channels generically described by CXL expressions I am calling a *Channel Group*. A Channel Group representing a function *signature* I am calling a *channel complex*, whereas groups representing a function *call* I am calling a *channel package*. Input channels are then those whose carrier transfers occur in the package-to-complex direction, whereas output channels are the converse.

Alongside the package/complex distinction, we can also understand Channel Groups at two further levels. On the one hand, we can treat Channel Groups as found *in source code*, where they describe the general pattern of package/complex transfers. On the other hand, we can represent Channel Groups *at runtime* in terms of

the actual values and types held by carriers as transfers are effectuated prior to, and then after, execution of the called function. Accordingly, each Channel Group may be classified as a *compile-time* package or complex, or a *runtime* package or complex, respectively. The code accompanying this chapter includes a “Channel Group library”—for creating and analyzing Channel Groups via a special Intermediate Representation—that represents groups of each variety, so it can be used both for static analysis and for enhanced runtimes and scripting.

The channel/group/complex/package/carrier vocabulary, I believe, codifies a descriptive framework integrating the *semantic* and *syntactic* dimensions of source code and program execution. Specifically, on the *semantic* side, computer programs can be understood in terms of λ -calculi combined with models of computation (call-by-value or by-reference, eager and lazy evaluation, and so forth). These semantic analyses focus on how values change and are passed between functions during the course of a running program. From this perspective, source code is analyzed in terms of the semantics of the program it describes: where and when are values moved around?

Conversely, source code can also be approached *syntactically*, as well-formed expressions of a computer language. From this perspective, correct source code is matched against language grammars, and individual code elements (like tokens, code blocks, expressions, and statements)—plus their inter-relationships—are established against this background.

The theory of Channel Groups straddles both the semantic and syntactic dimensions of computer code. Semantically, carrier-transfers capture the fundamental building blocks of program semantics: the overall evolving runtime state of a program can be modeled as a succession of carrier-transfers, marking code-points bridged via a transfer. Meanwhile, syntactically, how carriers belong to channels—the carrier-to-channel map fixing carriers’ semantics—structures and motivates languages’ grammars and rules. In particular, carrier-transfers induce relationships between code-graph nodes. As a result, language grammars can be studied through code-graphs’ profiles insofar as they satisfy RDF and/or DH Ontologies.

In sum, a DH and/or Semantic Web representation of computer code can be a foundation for both semantic and syntactic analyses, and this may be considered a benefit of Channel Group representations even if they only restate what are established semantic patterns in mainstream programming language—for example, even if they are restricted to a **sigma-lambda-return-exception** Channel Algebra modeled around, say, C++ semantics prior to C++11 (more recent C++ standards also call for a “**capture**” channel for inline anonymous functions).

At the same time, one of my claims in this chapter is that more complex Channel Algebras can lead to new tactics for introducing more expressive type-theoretic semantics in mainstream programming environments. As such, most of the rest of this section will explore additional Channel Kinds and associated Channel Groups which extend, in addition to merely codifying, mainstream languages’ syntax and semantics.

3.5.2 Channelized-type interpretations of larger-scale source code elements

By intent, Channel Algebras provide a machinery for modeling function-call semantics more complex than “pure” functions which have only one sort of input parameter (as in lambda abstraction)—note that this is unrelated to parameters’ *types*—and one sort of (single-value) return. Examples of a more complex paradigm come from Object-Oriented code, where there are two varieties of input parameters (“**lambda**” and “**sigma**”); the “**sigma**” (**this**) carrier is privileged, because its type establishes the class to which function belongs—influencing when the function may be called and how polymorphism is resolved.^v

Another case study is offered by exceptions. A function throws an exception instead of returning a value. As a result, **return** and **exception** channels typically evince a semantic requirement (which earlier—see page 128—I sketched as an algebra stipulation): when functions have both kinds of channels, only one may have an initialized carrier after the function returns. Usually, thrown-exception values can only be bound to carriers in **catch(...)** formations—once held in a carrier they can be used normally, but carriers in **exception** channels themselves can only transfer values to other carriers in narrow circumstances (this in turn depends on delineating code blocks, which will be reviewed later). So **exception** channels are not a sugared form of ordinary returns, any more than objects are sugar for functions’ first parameter; there are axiomatic criteria defining possible formations of **exception** and **return** channels and carriers, criteria which are more transparently rendered by recognizing **exception** and **return** as distinct channels of communication available within function bodies.

In general, extensions to λ -Calculus are meaningful because they model semantics other than ordinary lambda abstraction. For example, method-calls (usually) have different syntax than non-method-calls, but ς -calculi are not trivial extensions or syntactic sugar for **lambda**s; the more significant difference is that sigma-abstracted symbols and types have different consequences for overload resolution and function composition than **lambda**-abstractions. Similarly, exceptions interact with calling code differently than return values. Instead of scattered λ -extensions, Channel Algebra unifies multiple expansions by endowing functions (their signatures, in the abstract, and function-calls, in the concrete) with multiple channels, each of which can be independently modeled by some λ -extension (objects, exceptions, captures, and so forth).

Specific examples of unorthodox λ s (objects, exceptions, captures) suggest a general case: relations or operators between procedures can be modeled as relations between their respective channels, subject to channel-specific semantic restrictions. A *method* can be described as a function with several different channels: “**lambda**”

^vAlso, as I discussed earlier (page 128), “chaining” method calls means that the result of one method becomes an object that may then receive another method (the following one in the chain). Such chaining allows for an unambiguous function-composition operator.

with ordinary arguments (as in λ -calculus); “**sigma**” channel with a distinguished **this** carrier (formally studied via “ ζ -calculus”); and a **return** channel representing the return value. Because the contrast between these channels is first and foremost *semantic*—they have different meanings in the semantics of the programs where they appear—channels may therefore have restrictions governed by programs’ semantics. For example, as I mentioned in the context of “method chaining,” it may be stipulated that both **sigma** and **return** channels can have at most one carrier; as a result, a special channel-to-channel operator can be defined which is specific to passing values between the carriers of **return** and **sigma** channels. This operator is available because of the intended semantics of the channel system. Each channel kind has its own semantic interpretation, with commensurate axioms and restrictions. Subject to these semantics, carrier-to-carrier operators translate to channel-to-channel operators. A Channel Algebra in this sense is not a single fixed system, but an outline for modeling function-call semantics in the context of different programming languages and environments.

As the preceding paragraphs have presupposed, different functions may have different kinds of channels, which may or may not be reflected in functions’ types (consider the question, can two functions have the same type, if only one may throw an exception)? This may vary between type systems; but in any case the contrast between channel “structures” is *available* as a criteria for modeling type descriptions. On this basis, as I will now argue, we can provide type-system interpretations to source code structures beyond just values and symbols.

3.5.2.1 Statements, blocks, and control flow

The previous paragraphs discussed expanded channel structures—with, for example, objects and exceptions—that model call semantics more complex than the basic **lambda+return** (of classical λ -Calculus). A variation on this theme, in the opposite direction, is to *simplify* call structures: procedures which lack a **return** channel have to communicate solely through side-effects, whose rigorous analysis demands a “type-and-effect” system. Even further, consider functions with neither **lambda** nor **return** (nor **sigma** nor, maybe, **exception**). As an alternate channel of communication, suppose function bodies are nested in overarching bodies, and can “capture” carriers scoped to the enclosing function. “Capture semantics” specifications in C++ are a useful example, because C++ (unlike most languages that support anonymous or “intra-expression” function-definitions) mandates that symbols are explicitly captured (in a “capture clause”), rather than allowing functions to access surrounding lexically scoped with no further notation: This helps visualize the idea that captured symbols are a kind of “input channel” analogous to **lambda** and **sigma**.

I contend this works just as well for code blocks. Any language which has blocks can treat them as unnamed function bodies, with a “**capture**” channel (but not **lambda** or **return**). When (by language design) blocks *can* throw exceptions, it is reasonable to give them “**exception**” channels (further work, that I put off for now, is needed for loop-blocks, with **break** and **continue**). Blocks can then be typed as function-like

values, under the convention that function-types can be expressed through descriptions of their channels (or lack thereof).

Consider ordinary source-code expressions to represent a transfer of values between graph structures: let Γ_1 and Γ_2 be code-graphs compiled from source at a call site and at the callee's implementation, respectively. The function call transfers values from carriers marked by Γ_1 nodes to Γ_2 carriers; with the further detail of "channel complexes" we can additionally say that the recipient Γ_2 carriers are situated in a graph structure which translates to a channel description. So the morphology of Γ_1 has to be consistent with the channel structure of Γ_2 . For regular ("value") expressions, we can introduce a new kind of channel (which in the demo I call "**fground**") acknowledging that the function called by an expression may itself be evaluated by its own expression, rather than named as a single symbol (as in a pointer-to-function call like $(*f)(x)$ in C). A segment of source code represents a value-expression insofar as an equivalent graph representation comprises a Γ semantically and morphologically consistent with the provision of values to channels required by a function call—including the **fground** channel on the basis of which the proper implementation (for overloaded functions) is selected. How the graph-structure maps to the appropriate channels varies by channel kind: for instance, the **return** channel is not passed to the callee, but rather bound to a carrier as the right-hand side of an assignment (an *rvalue*)—or else passed to a different function (thus an example of channel-to-channel connection without an intervening carrier). A well-formed Γ represents part of a procedure's code graph, specifically that describing how a channel complex is concretely provisioned with values (i.e., a package).

I will use the term *call-clause* to designate the portion of a code graph, and the associated collection of source code elements, describing a channel package. Term a call-clause *anchored* if its resulting value is held in a carrier (as in $y = f(x)$), and *transient* if this value is instead passed on (immediately) to another function (as in $h(f(x))$); moreover a call-clause can be *standalone* if it has no result value or this value is not used; and *multiply anchored* if it has several anchored result values—that is, a multi-carrier **return** channel, assuming the type system allows as much. Anchored and standalone call-clauses can, in turn, model *statements*: specifically, "assignment" and "standalone" statements, respectively.

This vocabulary can be useful for interpreting program flow. Assignment statements with no other side-effects can—in principle—be delayed until their grounding carrier is "convoluted" with some other carrier.^w When modeling eager-evaluation languages, particular edge-types can be designated as forcing a temporal order or else

^wOf course, the default choice of "eager" or "lazy" evaluation is programming-language-specific, but for abstract discussion of source code graphs, we have no a priori idea of temporality, of a program executing in time. This is not a matter of concurrency—we have no a priori idea of procedures running at the *same* time any more than of them running sequentially. Any temporal direction through a graph is an interpretation of the graph, and as such it is useful to assume that graphs in and of themselves assert no temporal ordering among their nodes or edges.

edges can be annotated with additional temporalizing details. Without this extra documentation, however, execution order among graph elements can be evaluated based on other criteria.

In the case of statements, an assignment without side-effects has temporalizing relations only with other statements using its anchoring carrier. In particular, the order of statements' runtime need not replicate the order in which they are written in source code. A Channel Algebra may make this the default case, modeling "lazy" evaluation languages, in the absence of any temporalizing factors. The actual runtime order among sibling statements—those in the same block—then depends, in the absence of further information, on how their anchoring carriers are used; this in turn works backward from a function's return channel (in the absence of exceptions or effectual calls). That is, runtime order works backward from statements that initialize carriers in the return channel, then carriers used in those statements, etc.

This order needs to be broken, of course, for statements with side-effects. A case in point is the expansion of "do-notation" in Haskell: without an a priori temporality, Haskell source code relies on the asymmetric order of values passed into lambda abstractions to enforce requirements that effectual expressions evaluate before other expressions (Haskell does not have "statements" per se). Haskell's **do** "blocks" can be modeled (in the techniques used here) as a series of assignment statements where the anchoring carrier of each statement becomes (i.e., transfers its value to) the sole occupant of a **lambda** channel marking a new function body, which includes all the following statements (and so on recursively). There are two concepts in play here: interpreting any sequence of statements (plus one terminating expression, which becomes a statement initializing a **return** carrier) as a function body (not just those covering the extent of a "block"); and interpreting assignment statements as passing values into "hidden" lambda channels. What *looks* like *one* block in Haskell source code internally maps to a string of blocks interspersed with hidden **lambda** transfers. Operationally, Haskell backs this syntactic convention with monad semantics—**lambda** values passed are not the actual value of the monad-typed carrier but its "contained" value (see Giorgolo and Asudeh [132] or Shan [133] for a review of monads^x). For the sake of discussion, let us call this a *monad-subblock* formation.

The temporalizing elements in this formation are the "hidden lambdas." In a multi-channel paradigm, we can therefore consider "monad-subblocks" with respect to other channels. Consider how individual statements can be typed: like blocks, statements may select from symbols in scope and can potentially result in thrown expressions, so their channel structure is something like **capture+exception**. Even without hidden lambdas, observe that the runtime order of statements can be fixed in situations where an earlier statement may affect the value (via non-constant capture) of a carrier whose value is then used by a later statement. So for languages with

^xI cite these articles among many because, written by linguists, they bring an extra multi-disciplinary interest; see also Chatzikyriakidis and Luo [134] and Luo and Soloviev [135]; Kiselyov [136] and Kiselyov and Shan [137]; or Brady [138].

a more liberal treatment of side-effects than Haskell, we can interpret chains of statements *in fixed order* as successively capturing (and maybe modifying) symbols that occur in multiple statements. Having discussed convoluted *carriers*, extend this to channels: in particular, say two **capture** channels are convoluted if there is a modifiable carrier in the first which is convoluted with a carrier in the second (this is an ordered relation). One statement must run before a second if their **capture** channels are convoluted, in that order.

This is approaching toward a “monad-subblock” formation using **capture** in place of **lambda**. To be sure, the Haskell monad-subblock does have the added gatekeeping dimension that the symbol occurring after its appearance as anchoring an assignment statement is no longer the symbol with a monad type, but a different (albeit visually identical) symbol taken from the monad. Between two statements, if the prior is anchored by a monad, the implementation of its **bind** function is silently called, with the subsequent (and all further) statements grouped into a block passed in to **bind**, which in turn (by design) both extracts its wrapped value and (if appropriate) calls the passed function. But this architecture can certainly be emulated on non-**lambda** channels—a transform that would belong to the larger topic of treating blocks as function-values passed to other functions, to which I now turn.

3.5.2.2 Code blocks as typed values

Insofar as blocks can be typed as procedures, they may readily be passed around: so loops, **if...then...else**, and other control flow structures can plausibly be modeled as ordinary function calls. This requires some extra semantic devices: consider the case of **if...then...else** (I will use this also to designate code sequences with potential “**else if**’s”), which has to become an associative array of expressions and functions with “block” type (e.g., with only **capture** and **exception** channels). We need, however, a mechanism to suppress expression evaluation. Recall that expressions are concretized channel-structures which include an **fground** channel providing the actual implementation to call. All we need then is to decorate **fground** with a flag marking whether eager or lazy evaluation is desired. Assume also that carriers can be declared that hold (or somehow point to) *expressions* that evaluate to typed values, in lieu of holding these values directly (note that this is by intent orthogonal to a type system: the point is not that carriers can hold values whose type is designed to encapsulate potential computations yielding another type, like **std::future** in C++). Consider again the nested-expression variant of $c_1 \bowtie c_2$: when the result of one function call becomes a parameter to another function, the value in the former’s **return** carrier (assume there is just one) gets transferred to a carrier in the latter’s **lambda** channel (or **sigma**, say). This handoff can be described before being effectuated: a language runtime is free to vary the order of expression-evaluation no less than of statements. The semantics of a carrier-transfer between f_2 ’s return and f_1 ’s lambda does not stipulate that f_2 has to *run* before f_1 ; language engines can provide semantics for $c_1 \bowtie c_2$ allowing c_1 to hold a delayed capability to evaluate the f_2 expression. Insofar as this is an option, functions can be given a signature—this would be included in

Sample 3.2 Implementing If/Then/Else blocks

```

void test_if_then_else(quint64 args_ptr)
{
    QVector<quint64>& args = *(QVector<quint64>*)
        (args_ptr);

    int i = 0;
    bool test = false;
    for(quint64 qui: args)
    {
        if(i % 2)
        {
            if(test)
            {
                PHR_Callable_Value** pcv =
                    (PHR_Callable_Value**) qui;
                (*pcv)->run();
                return;
            }
        }
        else 1
        {
            PHR_Expression_Object** pxo =
                (PHR_Expression_Object**) qui;

            PHR_Channel_Group_Evaluator* ev = (*pxo)->run();
            qint32 i1 = ev->get_result_value_as<qint32>();
            test = (bool) i1;
        }
        ++i;
    }
    ...
void init_basic_functions(PhaonIR& phr,
    PHR_Code_Model& pcm,
    PHR_Channel_Group_Table& table,
    PHR_Symbol_Scope& pss)
{
    init_test_functions(phr, pcm, table, pss);

    PHR_Type_System* type_system = pcm.type_system();
    PHR_Channel_System& pcs = *phr.channel_system();
    PHR_Channel_Semantic_Protocol* lambda =
        pcs["lambda"];
    ...

    PHR_Channel_Group g1;
    ...
    2
    {
        PHR_Type* ty = type_system->get_type_by_name(
            "argvec");
        PHR_Carrier* phc = new PHR_Carrier;
        phc->set_phr_type(ty);
        g1.init_channel(lambda, 1);
        (*g1[lambda])[0] = phc;
    }
    ...
    table.init_phaon_function(g1, pss, "if-t-e", 700,
        &if_t_e);
    table.init_phaon_function(g1, pss,
        "test-if-then-else", 700, &test_if_then_else);

    g1.clear_all();
}
...
}

```

the relevant TXL—where some carriers are of this “delayed” kind. Functions like **if...then...else** can then be declared in terms of these carriers.

Given a runtime engine based on Channel Algebra, deferred evaluation is relatively straightforward: any delayed expression would be saved according to its channel-package data structure, which can be passed to functions as an encapsulation of the (not-yet-evaluated) expression itself. Code Sample 3.2 shows an implementation from the demo, on the runtime side. At ❶, a pointer to the relevant unevaluated expression is extracted, and a **run** method is called which completes the hitherto-delayed evaluation. The `demo test_if_then_else` procedure takes an “**argvec**” parameter ❷, which allows a variant number of blocks and expressions to be passed as inputs (analogous to C++ **var_arg** lists). The code around ❸ shows a channel complex being constructed which is then used to register the signature for the **if...then...else** kernel function in a lookup table.

Meanwhile, the hook into this C++ runtime code is demonstrated in Sample 3.3, in the “channelized” Intermediate Representation used for the demo. One pertinent point in this code is the instruction at ❹, where the name **temp_anchor_channel_group_by_need** suggests how the compiled channel package is being stored on a “by need” basis (internally, it gets a flag suppressing evaluation before being used in runtime procedures). At ❺, a block itself is assigned to a “callable value” type (cf. the “cv” initials). The arguments identified for the kernel procedure ❻, then, alternate between encapsulations of by-need expressions and compiled blocks deemed internally as opaque “callable values.”

A thorough treatment of blocks-as-functions also needs to consider standard procedural affordances like **break** and **continue** statements. Since blocks can be nested, some languages allow inner blocks to express the codewriter’s intention to “break out of” an outer block from an inner block. One way to model this via Channel Algebra is to introduce a special kind of return channel for blocks (called a “**break**,” perhaps) which, when it has an initialized carrier, uses this channel to hold a value that the enclosing block interprets in turn: by examining the inner **break** the immediately outer block can decide whether it itself needs to “break” and, if so, whether its own **break** channel needs to have an initialized carrier. The presence of such a **break** can type-theoretically distinguish loop blocks from blocks in (say) **if...then...else** contexts.

Further discussion of code models via Channel Groups and Channel Algebras is outside the scope of this chapter, but is demonstrated in greater detail in the accompanying code-set. Hopefully, the best way to present Channel Semantics outside the basic **lambda/sigma/return/exception** quartet is via demonstrations in live code. In that spirit, the demo code focuses on practical engineering and problem-solving where channel models can be useful, and I will briefly review its structure and its organizing rationales in the Conclusion.

Sample 3.3 Channelized intermediate representation, with deferred evaluation

```
.; generate_from_fn_node ;.
push_carrier_stack $ fgroud ;.
hold_type_by_name $ fbase ;.
push_carrier_symbol $ &test_if_then_else ;.
.; args ;.
push_carrier_stack $ lambda ;.

push_unwind_scope $ 1 result ;.

.; unwind_scope: 1 ;.

.; generate_from_fn_node ;.
push_carrier_stack $ fgroud ;.
hold_type_by_name $ fbase ;.
push_carrier_raw_value $ #=? ;.
.; args ;.
push_carrier_stack $ lambda ;.
hold_type_by_name $ u4 ;.
push_carrier_raw_value $ 4 ;.
hold_type_by_name $ u4 ;.
push_carrier_raw_value $ 5 ;.
push_carrier_stack $ result ;.
index_channel_group ;.
coalesce_channel_group ;.
.; pop ;.
pop_unwind_scope ;.
temp_anchor_channel_group_by_need ;.
.; end fgroud entry ;.
hold_type_by_name $ u4 ;.
push_carrier_expression ;.

.; block ... ;.
@fnp ;.

.; generate_from_fn_node ;.
push_carrier_stack $ fgroud ;.
hold_type_by_name $ fbase ;.
push_carrier_symbol $ &prn ;.
.; args ;.
push_carrier_stack $ lambda ;.
hold_type_by_name $ u4 ;.
push_carrier_raw_value $ 78 ;.
coalesce_channel_group ;.
evaluate_channel_group ;.
delete_temps ;.
delete_retired ;.
clear_temps ;.
reset_program_stack ;.
.; end of statement ;.
@fne ;.

.; end block ... ;.
hold_type_by_name $ pcv ;.
push_carrier_anon_fn @ last_source_fn_name ;.
```

3.6 Conclusion

To regard data modeling as just a practical, behind-the-scenes endeavor is to underestimate the scientific richness and importance of data modeling paradigms as theoretical constructs. In science, data models delineate the structure of information generated during scientific investigations (e.g., experiments and field work), and so their structure concretizes scientific theories and/or experimental protocols. Meanwhile, data modeling has to balance the complexity of human concepts with a predictability conducive to software and computational treatments; data modeling thereby helps expose the boundary, in human cognition, between what is mechanical and what is not—between the “mind as computer” metaphor and the philosophies of “situational” and “embodied” cognition that push against it.

With its resonance against these larger themes, data modeling is not just an operational prerequisite for scientific or technology research—and then the conversion of new discoveries into new technologies with practical benefits—but also an interdisciplinary nexus informed by and relevant to Computer Science, mathematics, Philosophy of Science, Sociology of Knowledge, formal semantics, and so forth.

One key interdisciplinary question is: how can data models be expressive enough to represent cultural and scientific ideas and artifacts—without any sense of conceptual mismatch or simplification—but also serve a software ecosystem? To be employed, that is, in a context where data structures require sufficient stability and classifiability that they are amenable to algorithms and mutations to accommodate different software roles, such as database and GUI presentations? Data models should be *systematic* so that they engender safe, reliable code. On the other hand, digital resources should be expressive enough to represent complex concepts without “dehumanizing” their structure—failing to recognize connections or distinctions which are part of human conceptualization, even if these are technically challenging to model computationally.

Achieving all of these goals involves a certain balancing act, where data repositories are modeled via expressive, fine-grained prototypes without becoming too unstructured, or too heterogeneous, for rigorous software implementations. The technical terrain of Ontology-based or type-theoretic modeling can therefore be seen as a drive to expand models’ expressiveness as far as possible, but without losing their underlying formal rigor and tractability. In terms of data models, this can be reflected in the evolution from fixed-field structures (like spreadsheets and relational databases) to labeled-graph Ontologies to Hypergraphs and other multiscale graph paradigms. Parallel to the emergence of Semantic Web technology, there is also a body of research in Scientific Computing, where expressiveness translates to modeling strategies which encapsulate scientific theories and workflows—cf. Object-Oriented simulations (Refs. [139, 140] being a good case study) and such formats or approaches as Conceptual Space theory (in science and linguistics) and Conceptual Space Markup Language [141–148]. Meanwhile, in type theory, a similar impetus leads from the simple type systems of Typed Lambda Calculi through to Dependent

Types, typestate, effect systems, Object-Orientation, and other properties of modern programming environments.

Whatever their features, data models are ultimately only as usable as the software that receives them. Applications may be importing Cyber-Physical measurements “in real time” or affording access to archived research datasets, but in each case the structured formats of shared and/or persisted information must be transformed into interactive (usually GUI-based) presentations if applications are to qualify as productive viewers onto the relevant information space. This is how we should understand the criterion of expressiveness: expressiveness at the modeling level is a means to an end; the ultimate goal is “expressive” software, that is, software whose layout, visual presentations, and interactive features/responsiveness render applications effective vehicles for interfacing with complex, nuanced digital content. Ultimately, then, data models are effective to the extent that they promote effective software engineering for the applications that transform modeled data into user-facing digital content.

On the other hand, this leaves room for differences in what is prioritized: data models can be targeted at a narrow, specialized set of software end-points, or can be designed flexibly to work with a diversity of software products, in the present and going forward. Broader application-scope is desirable in theory, but practically speaking a data model that is open-ended enough to work with a range of software components is potentially too provisional, or insufficiently detailed, to promote the highest-quality software.

Information Technology in the last one or two decades evidently has favored general-purpose data models—or at least serialization techniques—that exist in isolation from applications that work with them. Canonical examples would be JSON, XML, and RDF. Conceptually, however, data models’ most important manifestation are in the software components where they are shared—sent (perhaps indirectly via a generated archive) and received. To the degree that multi-purpose formats like XML are beneficial, there merits are in part that developers can anticipate the code that generated and/or will receive the data: while programmers do not necessarily just write code off of an XML sample (or corresponding Document Type Declaration), any XML document or DTD gives us a rough idea of what its client code would look like.

Nevertheless, for robust software engineering we should aspire to something more rigorous than that. In effect, we should consider documentation of components which send and/or receive data structures to be an intrinsic aspect of rigorous data modeling itself: description of the procedures that construct, serialize/deserialize, validate, and transform data structures, particularly those procedures supplying functionality determinative of their components’ ability to be part of a conformant data-sharing network. In this sense data and code modeling coincide. In particular, characterization of individual procedures—their types, assumptions, and requirements—is an essential building-block of data models generally. Data structures can be indirectly systematized in terms of the procedures which act upon them.

With this background, the code archive supplementing this chapter operationalizes the notion of “Procedural Hypergraph Ontologies,” combining features of

Procedural Ontologies and of Directed Hypergraphs that I have presented in this chapter. Procedural Hypergraph Ontologies extend (or diverge from) conventional Semantic Web Ontology partly by orienting toward Hypergraphs, but more substantially by centering on this procedural dimension: the role of an Ontology being to describe components' procedural interface as well as their targeted data structures. Using a phrase I introduced above (page 72), this technology aspires to promote a *procedure-oriented, software-centric* ("POSC") understanding of data-sharing pipelines.

In particular, the demo presents both a hypergraph serialization format and a methodology for generating interface descriptions, based on channel complexes. The demo code shows a compilation process that works with channel groups, branching off into a runtime engine which actually evaluates channel packages and, separately, algorithms to compile information about procedure signatures and function calls. This last capability can be a point for embedding more detailed Interface Definition metadata, including via the non-standard channel protocols I have discussed in this chapter. Both static data structures and compiled channel groups translate to a Hypergraph format, which thereby serves as a common denominator between code and data.

Architecturally, then, the demo includes several datasets repackaged in a hypergraph-serialization format, and, simultaneously, application-level code that bridges the datasets to GUI components. The code base parses serialized hypergraphs to in-memory hypergraphs and then traverses them, using a kind of visitor pattern, to build in-memory C++ objects, which in turn are mapped to GUI objects. So this chain of processing steps models hypergraph-based data representations and the logistics of incorporating them at the application level. At the same time, C++ objects reconstituted from hypergraphs—as well as the GUI components which receive them—are documented with an Interface Description Language that employs channels for articulating procedural signatures, an IDL which in turn compiles to hypergraph structures leveraged for various code-analysis tasks (e.g., generating a testing mechanism integrated with the application code).

The techniques thereby demonstrated can be practically adopted in several ways. On the one hand, concepts like channels and preconstructors can be applied to mainstream programming languages such as C++, becoming new design patterns or new coding practices that, over a large code base, can help produce components which are statically analyzable and (by systematically documenting and validating coding assumptions) prioritize safety at runtime. On the other hand (as profiled via special languages and Intermediate Representations in the demo), the techniques I have outlined can be used for new data and code models that guide, test, and/or retroactively analyze software components (e.g., using Channel Algebra in a fine-grained Interface Definition Language).

Aside from these practical applications, moreover, I contend that channels and Channelized Hypergraphs can be of interest in topics like linguistics and the Philosophy of Science as well—insofar as data models and representations of information flow encapsulate the structure of scientific theories, and the conceptual networks that lie behind Natural Language as well as formal semantics.

References

- [1] C. Becker, et al., Sustainability design and software: the Karlskrona Manifesto, 2015, Available from: <http://www.cs.toronto.edu/sme/papers/2015/Beckeretal-ICSE2015.pdf>.
- [2] R. Ashri, et al., Towards a Semantic Web Security Infrastructure, American Association for Artificial Intelligence, 2004, Available from: <https://www.aaai.org/Papers/Symposia/Spring/2004/SS-04-06/SS04-06-012.pdf>.
- [3] A. Dwivedi, et al., Cancellable biometrics for security and privacy enforcement on semantic web, 2011. Available from: <https://pdfs.semanticscholar.org/7c7c/957edf8dd1deb2c5baf315021d6fc387d030.pdf>.
- [4] L. Kagal, et al., A policy-based approached to security for the semantic web, 2003, Available from: https://ebiquity.umbc.edu/_file_directory_/papers/60.pdf.
- [5] T. Takahashi, et al., Ontological approach toward cybersecurity in cloud computing, in: 3rd International Conference on Security of Information and Networks (SIN 2010), September 7–11, Taganrog, Rostov Oblast, Russia, 2010, Available from: <https://arxiv.org/pdf/1405.6169.pdf>.
- [6] M. Tavakolifard, On Some Challenges for Online Trust and Reputation Systems (Dissertation), Norwegian University of Science and Technology, 2012, Available from: <https://pdfs.semanticscholar.org/fc60/d309984eddd4f4229aa56de2c47f23f7b65e.pdf>.
- [7] B. Thuraisingham, Security standards for the semantic web, Comput. Stand. Inter. 27 (3) (2005) 257–268, Available from: <https://pdfs.semanticscholar.org/f49c/6558265fcfb0cbb3221af089d5deb06aa35.pdf>.
- [8] M.V. Malko, The chernobyl reactor: design features and reasons for accident, Available from: <http://www.rri.kyoto-u.ac.jp/NSRG/reports/kr79/kr79pdf/Malko1.pdf>, 2002.
- [9] J.-E. Yang, Fukushima Dai-Ichi accident: lessons learned and future actions from the risk perspectives, Nucl. Eng. Technol. 46 (1) (2014) 27–38, Available from: <https://www.sciencedirect.com/science/article/pii/S1738573315300875>.
- [10] E. Bellin, et al., Democratizing information creation from health care data for quality improvement, research, and education—The Montefiore Medical Center Experience, Acad. Med. 85 (8) (2010), Available from: <https://pdfs.semanticscholar.org/ad02/adebdfe8d51c6defb120aac9f6f102e16596.pdf>.
- [11] E. Bellin, The cohort paradigm, 2015, Available from: <http://exploreclg.montefiore.org/upload/training-materials/The%20Cohort%20ParadigmV30.pdf>.
- [12] W. Jeltsch, Categorical semantics for functional reactive programming with temporal recursion and corecursion, in: R. Krishnaswami, P.B. Levy (Eds.), Mathematically Structured Functional Programming 2014 (MSFP 2014), 2014, pp. 127–142, Available from: <https://arxiv.org/pdf/1406.2062.pdf>.
- [13] J. Paykin, et al., Curry-Howard for GUIs: or, user interfaces via linear temporal, classical linear logic, 2014, Available from: <https://www.cl.cam.ac.uk/nk480/obt.pdf>.
- [14] J. Paykin, et al., The essence of event-driven programming, 2016, Available from: https://jpaykin.github.io/papers/pkz_CONCUR_2016.pdf.
- [15] J. Bolt, et al., Interacting conceptual spaces I: grammatical composition of concepts, 2017, Available from: <https://arxiv.org/pdf/1703.08314.pdf>.
- [16] B. Fong, Decorated cospans, Theor. Appl. Categ. 30 (33) (2015) 1096–1120, Available from: <https://arxiv.org/abs/1502.00872>.
- [17] B. Fong, The Algebra of Open and Interconnected Systems (Dissertation), Oxford University, 2016, Available from: <https://arxiv.org/pdf/1609.05382.pdf>.

- [18] A. Kissinger, Finite matrices are complete for (dagger-)hypergraph categories, 2014, Available from: <https://arxiv.org/abs/1406.5942>.
- [19] T. Elliott, et al., Guilt free ivory, in: Haskell Symposium, 2015, Available from: <https://github.com/GaloisInc/ivory/blob/master/ivorypaper/ivory.pdf?raw=true>.
- [20] S. Paul, A. Prakash, Supporting queries on source code: a formal framework, in: Int. J. Softw. Eng. Knowl. Eng. (Special Issue on Reverse Engineering) 4 (3) (1994) 325–348, Available from: <http://web.eecs.umich.edu/~aprakash/papers/ijseke94.pdf>.
- [21] G. Mishne, M. de Rijke, Source code retrieval using conceptual similarity, in: Coupling Approaches, Coupling Media and Coupling Languages for Information Retrieval, Proc. RIAO, Vaucluse, France, 2004, pp. 539–554, Available from: <https://staff.fnwi.uva.nl/m.derijke/Publications/Files/riao2004.pdf>.
- [22] S.R. Tilley, et al., Towards a framework for program understanding, in: Proc. WPC '96, 4th Workshop on Program Comprehension, Berlin, Germany, 1996, pp. 19–28, Available from: <https://pdfs.semanticscholar.org/71d0/4492be3c2abf9e1a88b9b263193a5c51eff1.pdf>.
- [23] M. Altini, Combining wearable accelerometer and physiological data for activity and energy expenditure estimation, in: Proc. Wireless Health 13, November 1–3, 2013, Baltimore, Available from: <https://www.marcoaltini.com/uploads/1/3/2/3/13234002/1569766907-altini.pdf>.
- [24] V. Benevičius, et al., Finite element model of MEMS accelerometer for accurate prediction of dynamic characteristics in biomechanical applications, J. Vibroeng. 13 (4) (2011) 803–809, Available from: <https://www.jvejournals.com/article/10526/pdf>.
- [25] J. Knight, et al., Uses of accelerometer data collected from a wearable system, Pers. Ubiquitous Comput. 11 (2) (2007) 117–132, Available from: http://pureoai.bham.ac.uk/ws/files/4856321/PUC_Accel.pdf.
- [26] H. Lee, et al., An enhanced method to estimate heart rate from seismocardiography via ensemble averaging of body movements at six degrees of freedom, Sensors 18 (1) (2018), pii: E238, Available from: <https://www.ncbi.nlm.nih.gov/pubmed/29342958>.
- [27] G.R. Wetherington Jr., et al., Two-year operational evaluation of a consumer electronics-based data acquisition system for equipment monitoring, in: E. Wee Sit, C. Walber, P. Walter, S. Seidlitz (Eds.), Sensors and Instrumentation, Conference Proceedings of the Society for Experimental Mechanics Series, vol. 5, 2017, Available from: <https://www.osti.gov/servlets/purl/1393863>.
- [28] L. Zhu, et al., Development of a high-sensitivity wireless accelerometer for structural health monitoring, Sensors 18 (1) (2018), pii: E262, Available from: <https://www.ncbi.nlm.nih.gov/pubmed/29342102>.
- [29] L. Auguste, D. Palsana, Mobile whole slide imaging (mWSI): a low resource acquisition and transport technique for microscopic pathological specimens, BMJ Innov. 1 (3) (2015), Available from: https://www.researchgate.net/publication/279276605_Mobile_Whole_Slide_Imaging_mWSI_A_low_resource_acquisition_and_transport_technique_for_microscopic_pathological_specimens.
- [30] N. Farahani, et al., Whole slide imaging in pathology: advantages, limitations, and emerging perspectives, 2015, Available from: <https://www.dovepress.com/whole-slide-imaging-in-pathologyadvantages-limitations-and-emerging-p-peer-reviewed-article-PLMI>.
- [31] A. Kale, S. Aksoy, Segmentation of cervical cell images, in: Proc. 20th International Conference on Pattern Recognition, Istanbul, 2010, pp. 2399–2402, Available from: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.619.3398&rep=rep1&type=pdf>.

- [32] H. Edelsbrunner, J. Harer, Persistent homology—a survey, *Discrete Comput. Geom.* 453 (2008), Available from: <https://www.maths.ed.ac.uk/v1ranick/papers/edelhare.pdf>.
- [33] H. Maxwell, Persistent homology of finite topological spaces, 2010, Available from: <http://www.math.uchicago.edu/may/VIGRE/VIGRE2010/REUPapers/Maxwell.pdf>.
- [34] H. Strange, et al., Modeling mammographic microcalcification clusters using persistent mereotopology, *Pattern Recognit. Lett.* 47 (1) (2014) 157–163, Available from: <https://www.sciencedirect.com/science/article/pii/S0167865514001263>.
- [35] J. Choi, R. Gutierrez-Osuna, Using heart rate monitors to detect mental stress, in: Proc. 6th International Workshop on Wearable and Implantable Body Sensor Networks, 2009, pp. 219–223, Available from: http://research.cse.tamu.edu/prism/publications/bsn09_choi.pdf.
- [36] J. Salamon, et al., Towards the automatic classification of avian flight calls for bioacoustic monitoring, *PLoS ONE* 11 (11) (2016) 1–26, Available from: http://www.justinsalamon.com/uploads/4/3/9/4/4394963/salomon_flightcalls_plosone_2016.pdf.
- [37] W.-L. Chao, Face recognition, Available from: <http://disp.ee.ntu.edu.tw/pujols/Face%20Recognition-survey.pdf>.
- [38] Y. Duan, et al., Topology preserving graph matching for partial face recognition, in: Proceedings of the IEEE International Conference on Multimedia and Expo (ICME), 2017 Available from: http://ivg.au.tsinghua.edu.cn/people/Yueqi_Duan/ICME17_Topo%20Preserving%20Graph%20Matching%20for%20Partial%20Face%20Recognition.pdf.
- [39] G.B. Huang, et al., Towards unconstrained face recognition, in: Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, Anchorage, Alaska, 2008, pp. 1–8, Available from: http://vis-www.cs.umass.edu/papers/unconstrained_face_workshop.pdf.
- [40] P. Kalaiselvi, S. Nithya, Face recognition system under varying lighting conditions, *IOSR J. Comput. Eng.* 14 (3) (2013) 79–88, Available from: <http://www.iosrjournals.org/iosr-jce/papers/Vol14-issue3/M01437988.pdf>.
- [41] F. Lu, et al., Adaptive linear regression for appearance-based gaze estimation, *IEEE Trans. Pattern Anal. Mach. Intell.* 36 (10) (2014) 2033–2046, Available from: <https://www.ncbi.nlm.nih.gov/pubmed/26352633>.
- [42] F. Adib, et al., Smart homes that monitor breathing and heart rate, in: Proc. 33rd Annual ACM Conference on Human Factors in Computing Systems, Seoul, Republic of Korea, April 18–23, 2015, pp. 837–846, Available from: <http://wittrack.csail.mit.edu/vitalradio/content/vitalradio-paper.pdf>.
- [43] M. Soltane, et al., Face and speech based multi-modal biometric authentication, *Int. J. Adv. Sci. Technol.* 21 (2010), Available from: https://www.researchgate.net/publication/228463467_Face_and_Speech_Based_Multi-Modal_Biometric_Authentication.
- [44] F. Albalas, Security-aware CoAP application layer protocol for the internet of things using elliptic-curve cryptography, *Int. Arab J. Inf. Technol.* 15 (3A) (2018), Available from: https://www.researchgate.net/publication/325987571_Security-aware_CoAP_Application_Layer_Protocol_for_the_Internet_of_Things_using_Elliptic-Curve_Cryptography.
- [45] B. Djamaa, et al., Hybrid CoAP-based resource discovery for the internet of things, *J. Amb. Intel. Hum. Comp.* 8 (3) (2017) 357–372, Available from: https://dspace.lib.cranfield.ac.uk/bitstream/handle/1826/11602/Hybrid_CoAP-based_resource_discovery-Internet_of_Things-2017.pdf?sequence=3&isAllowed=y.

- [46] R. Giambona, et al., MQTT+: enhanced syntax and broker functionalities for data filtering, processing and aggregation, in: Proc. Association for Computing Machinery Conference'17, July 2017, Washington, DC, 2018, Available from: <https://arxiv.org/pdf/1810.00773.pdf>.
- [47] C. Gündoğann, et al., NDN, CoAP, and MQTT: a comparative measurement study in the IoT, in: Proc. ACM Conference on Information-Centric Networking, 2018, Available from: <https://arxiv.org/pdf/1806.01444.pdf>.
- [48] J. Haikara, Publish-subscribe communication for CoAP, 2017, Available from: <http://kth.diva-portal.org/smash/get/diva2:1111621/FULLTEXT01.pdf>.
- [49] A. Rodriguez, et al., On modelling and validation of the MQTT IoT protocol for M2M communication, in: Proc. PSNE@Petri Nets/ACSD, 2018, Available from: <http://ceurws.org/Vol-2138/paper5.pdf>.
- [50] A. Patro, S. Banerjee, COAP: a software-defined approach for managing residential wireless gateways, Available from: https://research.cs.wisc.edu/wings/projects/coap/papers/coap_spec.pdf.
- [51] K. Belhajjame, Using a suite of ontologies for preserving workflow-centric research objects, Web Semant. 2015, Available from: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3199184.
- [52] K. Belhajjame, et al., Workflow-centric research objects: first class citizens in scholarly discourse, in: Proc. 2nd Workshop on Semantic Publishing, vol. 903, 2012, Available from: <http://ceurws.org/Vol-903/paper-01.pdf>.
- [53] P.E. Bourne, et al., Improving the future of research communications and e-scholarship, in: Manifesto From Dagstuhl Perspectives Workshop 11331, Available from: http://drops.dagstuhl.de/opus/volltexte/2012/3445/pdf/dagman_v001_i001_p041_11331.pdf.
- [54] K. Fenlon, Modeling digital humanities collections as research objects, 2019, Available from: https://drum.lib.umd.edu/bitstream/handle/1903/21860/fenlon_jcdl2019_researchObjects_final.pdf?sequence=1&isAllowed=y.
- [55] M.D. Wilkinson, et al., The FAIR guiding principles for scientific data management and stewardship, Sci. Data 3 (2016), Available from: <http://nrs.harvard.edu/urn-3:HUL.InstRepos:26860037>.
- [56] F. Abromeit, C. Chiarcos, Automatic detection of language and annotation model information in CoNLL corpora, in: Proc. 2nd Conference on Language, Data and Knowledge (LDK 2019), 2019, Available from: <http://drops.dagstuhl.de/opus/volltexte/2019/10387/>.
- [57] L. Kong, A.M. Rush, N.A. Smith, Transforming dependencies into phrase structures, in: Proc. 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, May–June, 2015, Available from: <https://www.semanticscholar.org/paper/Transforming-Dependencies-into-Phrase-Structures-Kong-Rush/fed1a48da3d35694da6a75952829fbce5ea232c9>.
- [58] J. Nivre, Dependency grammar and dependency parsing, 2005, Available from: <http://stp.lingfil.uu.se/nivre/docs/05133.pdf>.
- [59] T. Osborne, D. Maxwell, A historical overview of the status of function words in dependency grammar, in: Proc. Third International Conference on Dependency Linguistics (Depling 2015), Uppsala, Sweden, August 24–26, 2015, pp. 241–250, Available from: <https://www.aclweb.org/anthology/W15-2127>.
- [60] S. Reddy, et al., Transforming dependency structures to logical forms for semantic parsing, *Trans. Assoc. Comput. Linguist.* 4, 127–140, Available from: <https://aclweb.org/anthology/Q16-1010>.

- [61] G. Schneider, A Linguistic Comparison of Constituency, Dependency and Link Grammar, Zurich University 2008 (Diploma), Available from: https://files_ifi_uzh_ch/cl_gschneid/papers/FINALSgeroldschneider-latl.pdf.
- [62] F. Xia, M. Palmer, Converting dependency structures to phrase structures, 2001, Available from: <http://www.aclweb.org/anthology/H01-1014>.
- [63] B. Goertzel, Probabilistic language networks: integrating word grammar and link grammar in the framework of probabilistic logic, 2008, Available from: <http://goertzel.org/ProwlGrammar.pdf>.
- [64] E. Moreau, From link grammars to categorial grammars, in: Proc. Categorial Grammars 2004, Montpellier, France, Jun, 2004, pp. 31–45, Available from: <https://hal.archives-ouvertes.fr/hal-00487053/document>.
- [65] R. Debusmann, Extensible Dependency Grammar: A Modular Grammar Formalism Based on Multigraph Description (Dissertation), Universität des Saarlandes, 2006.
- [66] R. Debusmann, D. Duchier, A. Rossberg, Modular grammar design with typed parametric principles, in: J. Rogers (Ed.), Formal Grammar/Mathematics of Language, CSLI Publications, 2005, Available from: <http://web.stanford.edu/group/cslipublications/cslipublications/FG/2005/debusmann.pdf>.
- [67] M. Gasser, Toward synchronous extensible dependency grammar, 2011, Available from: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/5643/3/Gasser_Freerbm_t11_Toward.pdf.
- [68] C. Chiarcos, N. Schenk, The ACoLi CoNLL libraries: beyond tab-separated values, In: Proc. 11th International Conference on Language Resources and Evaluation (LREC 2018), Available from: <https://aclweb.org/anthology/L18-1090>.
- [69] J. Graen, et al., Modelling large parallel corpora: the Zurich parallel corpus collection, 2019, Available from: http://corpora.ids-mannheim.de/CMLC7-final/CMLC-7_2019-Graen_et_al.pdf.
- [70] D. Hershcovitch, et al., Universal dependency parsing with a general transition-based DAG, 2018, Available from: http://www.cs.huji.ac.il/oabend/papers/daniel_ud_parser.pdf.
- [71] A. More, CoNLL-UL: universal morphological lattices for universal dependency parsing, In: Proc. 11th International Conference on Language Resources and Evaluation (LREC 2018), Available from: <http://coltekin.net/cagri/papers/more2018.pdf>.
- [72] M. Straka, et al., UDPipe: trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, POS tagging and parsing, In: Proc. 10th International Conference on Language Resources and Evaluation (LREC'16), Available from: http://www.lrec-conf.org/proceedings/lrec2016/pdf/873_Paper.pdf.
- [73] M. Engel, et al., Unreliable yet useful-reliability annotations for data in cyber-physical systems, in: Proc. 2011 Workshop on Software Language Engineering for Cyber-physical Systems (WS4C), 2011, Available from: <https://pdfs.semanticscholar.org/d6ca/ecb4cd59e79090f3ebbf24b0e78b3d66820c.pdf>.
- [74] L. Ramapantulu, et al., A conceptual framework to federate testbeds for cybersecurity, in: Proceedings of the Winter Simulation Conference, 2017, Available from: <http://simulation.su/uploads/files/default/2017-ramapantulu-teo-chang.pdf>.
- [75] F.L. Arantes, Requirements engineering of a web portal using organizational semiotics artifacts, Int. J. Comput. Eng. Inf. Technol. 5 (2013), Available from: <https://arxiv.org/pdf/1305.3255.pdf>.
- [76] K.R. Anderson, Freeing the essence of a computation, 1995, Available from: <https://citeseerx.ist.psu.edu/viewdoc/download>.

- [77] M. Escardó, W.K. Ho, Operational domain theory and topology of sequential programming languages, *Inf. Comput.* 207 (2009) 411–437, Available from: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.104.6465&rep=rep1&type=pdf>.
- [78] M. Hasegawa, Decomposing typed lambda calculus into a couple of categorical programming languages, in: Proceedings of the 6th International Conference on Category Theory and Computer Science, 1995, Available from: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.53.715&rep=rep1&type=pdf>.
- [79] J. Tucker, J. Zucker, Computation by ‘while’ programs on topological partial algebras, *Theor. Comput. Sci.* 219 (1999) 379–420, Available from: <https://core.ac.uk/download/pdf/82201923.pdf>.
- [80] B. Goetzel, et al., Engineering General Intelligence, Parts 1 & 2, Atlantis Press, 2014, Available from: http://wiki.opencog.org/w/Background_Publications.
- [81] H. Liu, et al., Mining biomedical data using RDF hypergraphs, in: 12th International Conference on Machine Learning and Applications, 2013, Available from: http://ix.cs.oregon.edu/dou/research/papers/icmla13_hypergraph.pdf.
- [82] M. Minas, Concepts and realization of a diagram editor generator based on hypergraph transformation, *Sci. Comput. Program.* 44 (2) (2002) 157–180, Available from: <https://www.sciencedirect.com/science/article/pii/S0167642302000370>.
- [83] M. Minas, H.J. Schneider, Graph transformation by computational category theory, 2010, Available from: <https://www2.informatik.uni-erlangen.de/staff/schneider/gtbook/fmn-final.pdf>.
- [84] B. Molnár, Applications of hypergraphs in informatics: a survey and opportunities for research, *Ann. Univ. Sci. Budapest. Sect. Comput.* 42, 261–282, Available from: http://ac.inf.elte.hu/Vol_042_2014/261_42.pdf.
- [85] A. Poulovassilis, M. Levene, A nested-graph model for the representation and manipulation of complex objects, *Data Knowl. Eng.* 6 (3) (1991) 205–224, Available from: <http://www.dcs.bbk.ac.uk/mark/download/tois.pdf>.
- [86] J.G. Stell, Granulation for graphs, *Int. J. Signs Semiot. Syst.* 2 (1) (2012) 32–71, Available from: <https://pdfs.semanticscholar.org/9e0f/a93a899e36dc3df62feabc004a0ecef4365d.pdf>.
- [87] J.G. Stell, Formal concepts analysis over graphs and hypergraphs, in: M. Croitoru, S. Rudolph, S. Woltran, C. Gonzales (Eds.), *Graph Structures for Knowledge Representation and Reasoning*, Springer, 2013, pp. 165–179, Available from: http://eprints.whiterose.ac.uk/78795/7/GKRLNCS_with_coversheet.pdf.
- [88] M. Croitoru, E. Compatangelo, Ontology constraint satisfaction problems using conceptual graphs, in: Proc. SGAI Conference (Specialist Group on Artificial Intelligence), 2006, Available from: <https://pdfs.semanticscholar.org/d05e/eb82298201d6fae0129c6d53fe16db6d4803.pdf>.
- [89] E. Damiani, et al., Modeling semistructured data by using graph-based constraints, 2003, Available from: <http://home.deib.polimi.it/schreibe/TeSI/Materials/Tanca/PDFTanca/csse.pdf>.
- [90] R. Angles, C. Gutierrez, Querying RDF data from a graph database perspective, in: A. Gómez-Pérez, J. Euzenat (Eds.), *The Semantic Web: Research and Applications*, in: *Second European Semantic Web Conference, ESWC 2005, Heraklion, Crete, Greece, May 29–June 1, 2005*.
- [91] M.A. Rodriguez, J.H. Watkins, Grammar-based geodeics for semantic networks, *Knowl.-Based Syst.* 23 (8) (2010) 844–855, Available from: <http://arxiv.org/pdf/1009.0670.pdf>.

- [92] T. Berners-Lee, N3Logic: a logical framework for the world wide web, 2007, Available from: <https://arxiv.org/pdf/0711.1533.pdf>.
- [93] Y. Wilks, The semantic web as the apotheosis of annotation, but what are its semantics?, IEEE Intel. Syst. 23 (2008) 41–49, Available from: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.131.4958&rep=rep1&type=pdf>.
- [94] G.A. Aranda-Corral, J. Borrego-Díaz, Mereotopological analysis of formal concepts in security ontologies, in: A. Herrero, E. Corchado, C. Redondo, A. Alonso (Eds.), Computational Intelligence in Security for Information Systems, Advances in Intelligent and Soft Computing, vol. 85, Springer, Berlin, Heidelberg, 2010 Available from: <https://core.ac.uk/download/pdf/158966553.pdf>.
- [95] T. Bittner, B. Smith, M. Donnelly, The logic of systems of granular partitions, 2002, Available from: <http://ontology.buffalo.edu/smith/articles/BittnerSmithDonnelly.pdf>.
- [96] T. Bittner, B. Smith, A taxonomy of granular partitions, in: MontelloD.R. (Ed.), Spatial Information Theory, COSIT, vol. 2205, Lecture Notes in Computer Science, 2001, Available from: http://qrg.northwestern.edu/papers/Files/Bittner_Smith_Taxonomy_granular_partitions.pdf.
- [97] M. Donnelly, et al., A formal theory for spatial representation and reasoning in biomedical ontologies, Artif. Intell. Med. 36 (2006) 1–27, Available from: <http://www.acsu.buffalo.edu/md63/DonnellyAIMed05.pdf>.
- [98] S.I. Fabrikant, Visualizing region and scale in information spaces, in: Proc. 20th International Cartographic Conference, ICC 2001, Beijing, China, 2001, pp. 2522–2529, Available from: <https://www.semanticscholar.org/paper/VISUALIZING-REGION-AND-SCALE-IN-INFORMATION-SPACES-Fabrikant/526a09e4767ff634c4cfbc51e6f7f4ebb700096a>.
- [99] J. Petitot, B. Smith, New foundations for qualitative physics, in: J.E. Tiles, G.T. McKee, C.G. Dean (Eds.), Evolving Knowledge in Natural Science and Artificial Intelligence, Pitman Publishing, London, 1990, pp. 231–249, Available from: http://ontology.buffalo.edu/smith/articles/qualitative_physics.pdf.
- [100] A. Segev, A. Gal, Putting things in context: a topological approach to mapping contexts and ontologies, in: SpaccapietraS. et al., (Ed.), Journal on Data Semantics IX, Lecture Notes in Computer Science, vol. 4601, 2007, Available from: <https://www.aaai.org/Papers/Workshops/2005/WS-05-01/WS05-01-003.pdf>.
- [101] B. Smith, A. Kumar, The ontology of blood pressure: a case study in creating ontological partitions in biomedicine, 2003, Available from: https://www.researchgate.net/publication/228961604_The_Ontology_of_Blood_Pressure_A_Case_Stud...
- [102] P. Ramellini, Boundary questions between ontology and biology, in: R. Poli, J. Seibt (Eds.), Theory and Applications of Ontology: Philosophical Perspectives, Springer, 2010, pp. 153–175, Available from: http://mirror.thelifeofkenneth.com/lib/electronics_archive/Theory_and_Applications_of_Ontology_Philosophical_Perspectives.pdf.
- [103] H. Paulheim, C. Bizer, Type inference in noisy RDF data, in: AlaniH. et al., (Ed.), The Semantic Web—ISWC 2013, ISWC 2013, Lecture Notes in Computer Science, vol. 8218, 2013, Available from: <http://www.heikopaulheim.com/docs/iswc2013.pdf>.
- [104] R. Brown, et al., Graphs of morphisms of graphs, Electron. J. Comb. 15 (2008), Available from: https://www.emis.de/journals/EJC/Volume_15/PDF/v15i1a1.pdf.
- [105] M. Abadi, L. Cardelli, A semantics of object types, in: Proceedings of the IEEE Symposium on Logic in Computer Science, Paris, 1994, Available from: <http://lucacardelli.name/Papers/PrimObjSemLICS.A4.pdf>.

- [106] J. Campos, V.T. Vasconcelos, Channels as objects in concurrent object-oriented programming, in: Proc. Programming Language Approaches to Concurrency and Communication-Centric Software 2010 (PLACES'10), Available from: <https://arxiv.org/pdf/1110.4157.pdf>.
- [107] K. Fisher, et al., A lambda calculus of objects and method specialization, Nordic J. Comput. 1 (1994) 3–37, Available from: <https://pdfs.semanticscholar.org/5cf7/1e3120c48c23f9cecdbe5f904b884e0e1a2d.pdf>.
- [108] E.N. Zalta, The modal object calculus and its interpretation, in: de RijkeM. (Ed.), Advances in Intensional Logic, 1996, pp. 245–276, Available from: <https://mally.stanford.edu/Papers/calculus.pdf>.
- [109] I. Keivanloo, et al., Semantic web-based source code search, 2010, Available from: https://www.researchgate.net/publication/228942934_SemanticWeb.
- [110] W. Klieber, et al., Using ontologies for software documentation, 2019, Available from: http://www.know-center.tugraz.at/download_extern/papers/MJCAI2009%20software%20ontology.pdf.
- [111] J. Lee, et al., Task-based conceptual graphs as a basis for automating software development, Int. J. Intel. Syst. 15, 1177–1207, Available from: <https://www.csie.ntu.edu.tw/jlee/publication/tbcg99.pdf>.
- [112] R. Turner, A.H. Eden, Towards a programming language ontology, 2007, Available from: https://www.researchgate.net/publication/242381616_Towards_a_Programming_Language_Ontology.
- [113] R. Witte, et al., Ontological text mining of software documents, 2007, Available from: <https://pdfs.semanticscholar.org/7034/95109535e510f81b9891681f99bae1e704fc.pdf>.
- [114] P. Wongthongtham, et al., Development of a software engineering ontology for multisite software development, IEEE Trans. Knowl. Data Eng. 21 (8) (2009) 1205–1217, Available from: https://www.researchgate.net/publication/220072121_Development_of_a_Software_Engineering_Ontology_for_Multisite_Software_Development.
- [115] M. Chochlik, A. Naumann, Static reflection: rationale, design and evolution, 2016, Available from: <http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2016/p0385r0.pdf>.
- [116] J.Y. Dangler, Categorization of Security Design Patterns (Electronic Theses and Dissertations), East Tennessee State University, Paper 1119, 2013, Available from: <https://dc.etsu.edu/cgi/viewcontent.cgi?article=2303&context=etd>.
- [117] D.R. Ireno, Dynamic factory: new possibilities for factory design pattern, in: F. Squazzoni (Ed.), Proceedings 28th European Conference on Modelling and Simulation, 2014, Available from: http://www.scs-europe.net/dlib/2014/ecms14papers/dis_ECMS2014_0114.pdf.
- [118] W.B. McNatt, J.M. Bieman, Coupling of design patterns: common practices and their benefits, in: Proceedings of Computer Software and Applications Conference (COMPSAC 2001), October 2001, Available from: <http://www.cs.colostate.edu/pubserv/pubs/McNatt-bieman-Pubs-McnattBieman01.pdf>.
- [119] M.J. Gabbay, A. Nanevski, Denotation of contextual modal type theory (CMTT): syntax and metaprogramming, J. Appl. Log. 11 (1) (2013) 1–29, Available from: <https://software.imdea.org/aleks/papers/cmtt/cmtt-semantics.pdf>.
- [120] E. Brady, Idris, a general purpose dependently typed programming language: design and implementation, 2013, Available from: <https://pdfs.semanticscholar.org/1407/220ca09070233dca256433430d29e5321dc2.pdf>.
- [121] J.-P. Bernardy, et al., Parametricity and dependent types, in: Proc. ICFP'10, September 27–29, 2010, Available from: <http://www.staff.city.ac.uk/ross/papers/pts.pdf>.

- [122] Idris Development Wiki, The effects tutorial, Available from: <http://docs.idris-lang.org/en/latest/effects/index.html>.
- [123] M. Tejíščák, E. Brady, Practical erasure in dependently typed languages, 2015, Available from: <https://eb.host.cs.st-andrews.ac.uk/drafts/dtp-erasure-draft.pdf>.
- [124] D.R. Christiansen, Practical Reflection and Metaprogramming for Dependent Types (Dissertation), IT University of Copenhagen, 2015, Available from: <http://davidchristiansen.dk/david-christiansen-phd.pdf>.
- [125] R.A. Eisenberg, Dependent Types in Haskell: Theory and Practice (Dissertation), University of Pennsylvania, 2017, Available from: <http://www.cis.upenn.edu/sweirich/papers/eisenberg-thesis.pdf>.
- [126] D. Gopalani, et al., A type system and type soundness for the calculus of aspect-oriented programming languages, in: Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS), vol. 1, 2012 March 14–16, Hong Kong. Available from: http://www.iaeng.org/publication/IMECS2012/IMECS2012_pp263-268.pdf.
- [127] K. Mehner, et al., Analysis of aspect-oriented model weaving, in: RashidA., OssherH. (Eds.), Transactions on Aspect-Oriented Software Development V, Lecture Notes in Computer Science, vol. 5490, 2009, Available from: http://www.mathematik.uni-marburg.de/swt/Publikationen_Taentzer/MMT09.pdf.
- [128] C. Zhang, H.-A. Jacobsen, Refactoring middleware with aspects, IEEE Trans. Parallel Distrib. Syst. 14 (12) (2003), Available from: https://pdfs.semanticscholar.org/0304/5c5cc518c7d44c3f7b117ea11dfaef4932a89.pdf?_ga=2.207853466.903112516.1533046888-196394048.1525384494.
- [129] G. Restall, Logics, situations and channels, 2006, Available from: <https://pdfs.semanticscholar.org/58a5/c8938d8e1500d4ae060cb40137e4dc011520.pdf>.
- [130] Z. Luo, Using signatures in type theory to represent situations, in: T. Murata, K. Mineshima, D. Bekki (Eds.), New Frontiers in Artificial Intelligence, JSAI-isAI 2014, Lecture Notes in Computer Science, vol. 9067, 2014, Available from: https://www.researchgate.net/publication/268079019_Using_Signatures_in_Type_Theory_to_Represent_Situations.
- [131] B. Toninho, et al., Functions as session-typed processes, in: BirkedalL. (Ed.), Foundations of Software Science and Computational Structures, FoSSaCS 2012, Lecture Notes in Computer Science, vol. 7213, 2012, Available from: <http://www.cs.cmu.edu/~fp/papers/fossacs12.pdf>.
- [132] G. Giorgolo, A. Asudeh, Monads as a solution for generalized opacity, in: Proceedings of the EACL 2014 Workshop on Type Theory and Natural Language Semantics (TTNLS), 2014, pp. 19–27, Gothenburg, Sweden, April 26–30, Available from: <http://www.aclweb.org/anthology/W14-1403>.
- [133] C.-C. Shan, Monads for natural language semantics, Available from: <http://arxiv.org/pdf/cs/0205026.pdf> (Archived 17 May 2002).
- [134] S. Chatzikyriakidis, Z. Luo, Individuation criteria, dot-types and copredication: a view from modern type theories, in: Association for Computational Linguistics. Proceedings of the 14th Meeting on the Mathematics of Language (MoL 14), 2015, pp. 39–50, Available from: <http://www.aclweb.org/anthology/W15-2304>.
- [135] Z. Luo, S. Soloviev, Dependent coercions, Electron. Notes Theor. Comput. Sci. 29 (1999) 152–168, Available from: <https://www.sciencedirect.com/science/article/pii/S1571066105803147>.

- [136] O. Kiselyov, Applicative abstract categorial grammars, Available from: <http://okmij.org/ftp/gengo/applicative-semantics/AACG.pdf>.
- [137] O. Kiselyov, C.-C. Shan, Lambda: the ultimate syntax-semantics interface, 2010, Available from: <https://pdfs.semanticscholar.org/fb41/793ae7098c40fcdb6706b905c48a270b0b48.pdf>.
- [138] E. Brady, State machines all the way down: an architecture for dependently typed languages, 2016, Available from: <https://www.idris-lang.org/drafts/sms.pdf>.
- [139] A. Telea, Visualisation and Simulation With Object-Oriented Networks (Dissertation), Eindhoven, 1999, Available from: <http://papers.cumincad.org/data/works/att/83cb/content.pdf>.
- [140] A. Telea, J.J. van Wijk, VISION: an object oriented dataflow system for simulation and visualization, 1999, Available from: <https://www.rug.nl/research/portal/files/3178139/1999ProcVisSymTelea.pdf>.
- [141] B. Adams, M. Raubal, A metric conceptual space algebra, 2009, Available from: <https://pdfs.semanticscholar.org/521a/cbab9658df27acd9f40bba2b9445f75d681c.pdf>.
- [142] B. Adams, M. Raubal, Conceptual space markup language (CSML): towards the cognitive semantic web, in: Proc. 2009 IEEE International Conference on Semantic Computing, Berkeley, CA, 2009, pp. 253–260, Available from: http://idwebhost-202-147.ethz.ch/Publications/RefConferences/ICSC_2009_AdamsRaubal_Camera-FINAL.pdf.
- [143] B. Adams, M. Raubal, The semantic web needs more cognition, 2010, Available from: http://www.semantic-web-journal.net/sites/default/files/swj37_0.pdf.
- [144] I. Douven, et al., Vagueness: a conceptual spaces approach, *J. Philos. Log.* 42 (1) (2010) 1–24, Available from: https://www.researchgate.net/publication/225689962_Vagueness_A_Conceptual_Spaces_Approach.
- [145] P. Gärdenfors, F. Zenker, Theory change as dimensional change: conceptual spaces applied to the dynamics of empirical theories, *Synthese* 190 (6) (2013) 1039–1058, Available from: <http://lup.lub.lu.se/record/1775234>.
- [146] K. Holmqvist, Dimensions of cognition, in: J. Allwood, P. Gärdenfors (Eds.), *Cognitive Semantics*, John Benjamins, Amsterdam, Philadelphia, 1999, pp. 153–171, Available from: <https://www.lucs.lu.se/spinning/categories/cognitive/Holmqvist/kenneth.pdf>.
- [147] M. Raubal, Formalizing conceptual spaces, 2004, Available from: http://www.raubal.ethz.ch/Courses/288MR_Spring08_Papers/Raubal_FormalizingConceptualSpaces_FOI_S04.pdf.
- [148] G. Strle, Semantics Within: The Representation of Meaning Through Conceptual Spaces (Dissertation), University of Novi Gorici, 2012.

A new method of power efficient speech transmission over 5G networks using new signaling techniques

4

Javaid A. Sheikh^a, Sakeena Akhtar^a, Arshid Iqbal Khan^a,
Shabir A. Parah^a, G.M. Bhat^b, Amy Neustein^c

^aPG Department of Electronics and IT, University of Kashmir, Srinagar, India

^bCollege of Engineering, University of Kashmir, Srinagar, India ^cFounder and CEO,
Linguistic Technology Systems, Fort Lee, NJ, United States

4.1 Introduction

Among all the multicarrier modulation techniques, the most widely used technique is Orthogonal Frequency Division Multiplexing (OFDM). OFDM has many advantages such as robustness to channel delays, single tap frequency domain equalization, and efficient implementation, and has been commonly used by 4G Communication Systems such as LTE and Wi-Fi [1]. However, unfortunately the spectral efficiency restricts its use in 5G communication systems because of the loss in spectral efficiency due to higher side lobes and the strict synchronization requirements. Thus new modulation techniques are being considered for 5G networks to overcome some of these factors [2]. The fifth-generation (5G) mobile communication system will emerge to meet new and unparalleled demands beyond the proficiency of previous generations of systems. Spectrum utilization, energy consumption, and cost are three vital factors which must be addressed in sustainable communication networks. In order to achieve sustainability, 5G networks need to improve the network efficiency deployment and operations. The requirement of mobile data traffic growth by a factor of thousands will be sustainably achieved by 5G. Users will have a fiber-like access data rate and very low latency user experience. 5G will be capable of connecting 100 billion devices together. It will be able to provide a reliable experience through a variety of scenarios including cases of ultra-high traffic volume density, ultra-high connection density, and ultra-high mobility. Based on services and awareness, 5G will also be able to provide intelligent optimization [3–5]. The spectrum is one of the most valuable resource for mobile communication. As an example, consider an LTE system at 20 MHz channel bandwidth using 100 resource blocks of 12 subcarriers each at an individual subcarrier spacing of 15 KHz. This utilizes

only 18 MHz of the available spectrum, resulting in 10% spectral loss. Additionally in CP-OFDM, the CP of 144 or 160 samples per OFDM symbol causes another 7% of spectral loss, thus resulting in an overall 17% loss in possible spectrum efficiency. In contrast to 4G, 5G networks must have 3–5 times improved spectrum efficiency and >100 times improved energy and cost efficiency.

Diversity is one of the key techniques that enhance the BER performance of an AWGN channel by increasing the signal-to-noise ratio. Diversity involves multiple communication links between the transmitter and the receiver, and each link carries the replica of the original message symbols. If any of the communication links are in deep fade, there is a probability that the message symbols reach the intended user, despite some of the links being in deep fade. The diversity technique can be used at the transmitter, i.e., transmitting diversity, or at the receiver, i.e., receiving diversity. Transmitting diversity involves use of multiple antennas at the transmitter and a single antenna at the receiver. Receiving diversity involves multiple receive antennas and a single transmit antenna.

4.2 Related work

Eeckhaute et al. presented a paper on performance of emerging multicarrier waveforms for 5G communications. The most promising waveform candidates for 5G air interface—filter-bank multicarrier (FBMC), universal-filtered multicarrier (UFMC), generalized frequency-division multiplexing (GFDM), and resource-block filtered orthogonal frequency-division multiplexing (RB-F-OFDM)—are compared to OFDM used in 4G in terms of spectral efficiency and numerical complexity. The existing technique, however, lacks the Bit Error Rate Analysis using various multicarrier modulation techniques. In our proposed technique, BER analysis using various modulation techniques for a real-time application is performed. In addition to spectral efficiency, Peak-to-Average Power Ratio (PAPR) for OFDM and UFMC is also calculated and compared for 4G and 5G systems [6].

Kishore et al. presented comprehensive analysis of Orthogonal Frequency Division Multiplexing (OFDM), Filter Bank Multicarrier (FBMC), and Universal Filtered MultiCarrier (UFMC). The PAPR of these techniques is analyzed by applying different subcarriers and modulation techniques, and varies according to the modulation techniques used. The existing technique focuses on the PAPR of various multicarrier modulation schemes for OFDM, FBMC, and UFMC systems. However, no comparison of spectral efficiencies of these multicarrier modulation techniques is carried out. In addition, the paper lacks the BER analysis using various modulation techniques for 5G networks which has been the main motive in our proposed technique [7].

Bohdan R. Tomiuk presented in [8] the concept that perfect maximal ratio diversity combining cannot be attained in a practical system due to channel gain estimation errors. The exactness of channel gain estimation has dependencies such as signal-to-noise ratio. A thorough analysis reveals that the old theoretical performance of Maximal Ratio Combining in sluggish fading cannot be realized when

the channel gain estimates are dependent on fixed channel gain estimation signal-to-noise ratio. It is proposed that the benefits of maximal ratio coherent reception over square-law/noncoherent reception in sluggish fading may be much less than previously believed. However, the work did not present the effects of the different modulation schemes on the BER analysis; it also lacks the spatial combining techniques at the receiver for optimum BER performance.

In [9], Prachi Tripathi discusses the fading problem as a significant issue in wireless communication, which can be minimized with assistance of diversity techniques. In this paper, focus has been placed on the Maximal Ratio Combining (MRC) diversity approach for mitigation of the fading problem in iterative interleave-division multiple access (IDMA) receivers. The approach explains the MRC diversity technique on the receiver side, with one transmit and two receive antennas and one transmit and four receive antennas in a low rate coded environment. This paper could not explain the applications of the MRC technique for BPSK and QAM systems, to optimally encode and decode the speech signals for the transmission over the channel contaminated with Additive White Gaussian Noise (AWGN).

4.3 Ultra-filter bank multicarrier modulation

As is well known, in the existence of multipath channels, plain orthogonal multicarrier modulation techniques are not capable of maintaining orthogonality due to Inter Symbol Interference (ISI) among adjacent multicarrier symbols. The traditional method in OFDM to overcome this problem is to introduce a Cyclic Prefix (CP) greater than the time spread introduced by the channel. This allows the operations of traditional transceiver implementations by IFFT and FFT blocks, but introduces a time overhead in the system, resulting in a loss of spectral efficiency. In Filter Bank Multicarrier Modulation (FBMC), the symbol duration is kept unaltered, which avoids the introduction of any time overhead. The overlap among adjacent symbols is removed by using additional filtering called a prototype filter bank at the transmitter and receiver side, in addition to using IFFT/FFT operations [10–12].

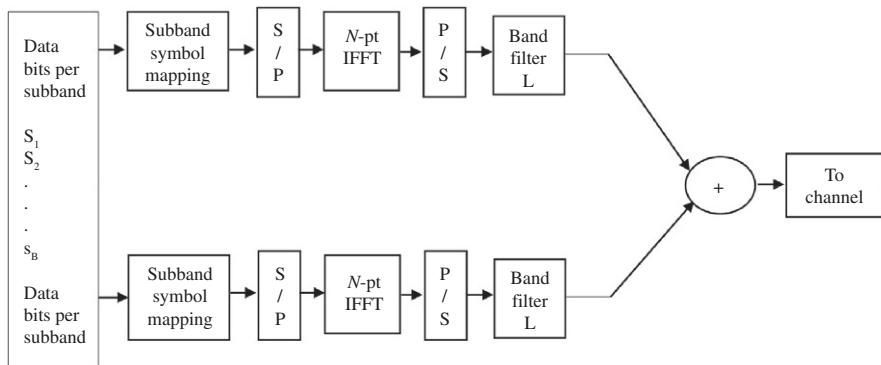
The Universal Filtered Multicarrier (UFMC) is a multicarrier modulation scheme that has been proposed by the EU-funded research project 5GNOW [13, 14]. Indeed, in filtered OFDM the whole set of subcarriers is filtered to limit side lobe effects, and in FBMC, modulation filtering is applied separately to each subcarrier, while in UFMC subcarriers are filtered in groups. UFMC is seen as a generalization of filtered OFDM and OFDM. Let N denote the overall number of subcarriers and B denote number of groups (subcarriers divided in separate groups). For simplicity, we assume that each group is composed of P subcarriers such that

$$N = BP \quad (4.1)$$

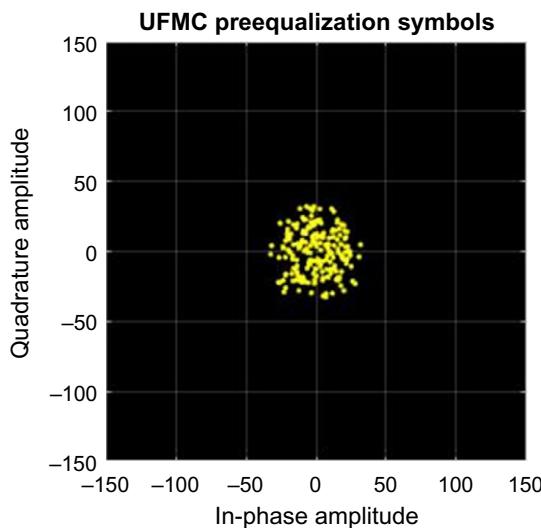
Let $s_1, s_2 \dots s_B$ be the P -dimensional QAM data bits per subband to be transmitted and V be the $(N \times N)$ IFFT matrix with $(N \times P)$ dimensions such that

$$V = [V_1 \ V_2 \ \dots \ V_B] \quad (4.2)$$

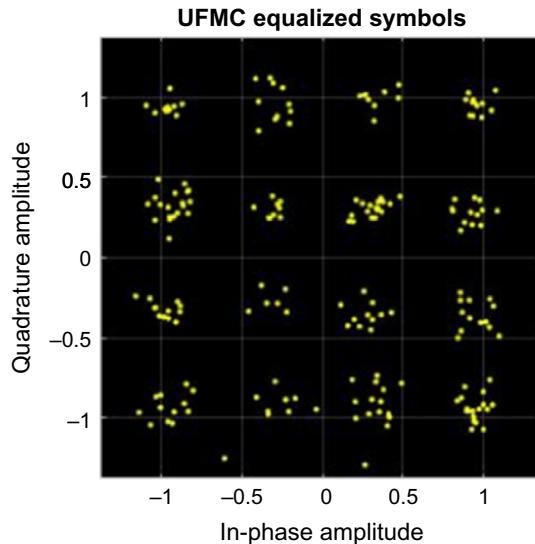
The full band of subcarriers (N) is divided into subbands. Each subband has a fixed number of subcarriers, and not all subbands need to be employed for a given transmission. The B data vectors s_1, s_2, \dots, s_B are processed with the IDFT submatrices V_1, V_2, \dots, V_B , respectively. An N -pt IFFT for each subband is computed, inserting zeros for the unallocated carriers. Each subband is filtered by a filter of length L , and the responses from the different subbands are summed. The filtering is done to reduce the out-of-band spectral emissions. Different filters per subband can be applied; however, in the proposed technique, the same filter is used for each subband [15]. The transmit-end processing is shown in Figs. 4.1–4.3.

**FIG. 4.1**

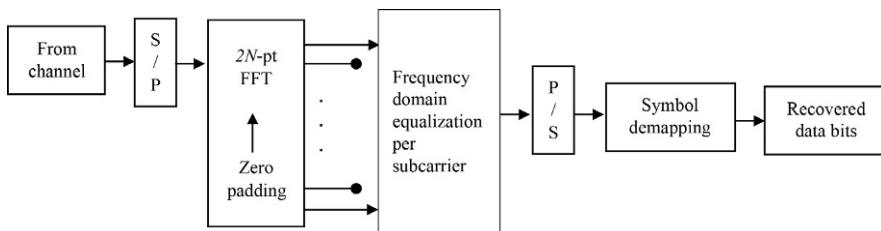
UFMC transmitter.

**FIG. 4.2**

UFMC preequalization symbols.

**FIG. 4.3**

UFMC equalized symbols.

**FIG. 4.4**

UFMC receiver.

Fig. 4.4 depicts the basic UFMC receive processing, which like OFDM is FFT-based. The subband filtering extends the receive time window to the next power-of-two length for the FFT operation. Every alternate frequency value corresponds to a subcarrier main lobe. In typical scenarios, per-subcarrier equalization is used for equalizing the joint effect of the channel and the subband filtering [16, 17]. In our proposed technique, only the subband filter is equalized because no channel effects are modeled. Noise is added to the received signal to achieve the desired SNR.

4.4 Space-time codes

Space-time coding is a pioneering transmit diversity technique that depends on coding over space, i.e., transmit antennas and time to remove diversity. Space-time

coding has received significant consideration attention in the design of high reliability and robust wireless communication systems due to its many advantages. To begin with, it enhances the downlink execution without the requirement for numerous receive antennas at the receive terminals. Second, it can be effectively joined with channel coding, understanding a coding gain along with the spatial diversity gain. Third, it does not require Channel State Information (CSI) at the transmitter, i.e., it works in open-loop mode. Finally, they have appeared to be adaptable against nonperfect working conditions, for example, antenna correlation, channel estimation errors, and Doppler effects. There are different methodologies in coding structures, including Orthogonal Space-Time Block Codes (OSTBC), Alamouti Codes, Space-Time Trellis Codes (STTC), Maximal Ratio Combining (MRC), Space-Time Turbo Trellis Codes (STTTC), and Layered Space-Time (LST) codes. A focal issue in every one of these plans is the misuse of multipath effects with a specific end goal to accomplish high spectral efficiencies and high performance execution gains [18]. In this section, we study the Alamouti Codes, the OSTBC Codes, and the MRC scheme for the thorough analysis of probability of error, i.e., Bit Error Rates (BER) in Binary Phase Shift Keying (BPSK) and Quadrature Amplitude Modulation (QAM) modulated systems. In the subsequent sections, we briefly introduce the important coding schemes like Alamouti, OSTBC, and MRC, and their applications in the measurement of BER for BPSK and QAM modulated systems are discussed below.

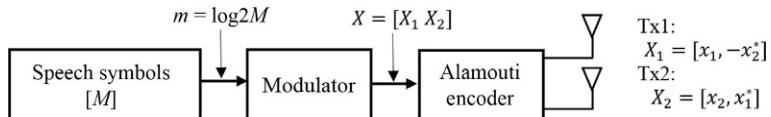
4.4.1 Bit error rate analysis using Alamouti coding

Alamouti Code is a space-time block code used for reliable signal transmission and processing in Multiple Input Single Output (MISO) systems. This code requires neither the Channel State Information (CSI) nor the knowledge of the channel coefficients at the transmitter; this makes it considerably more useful from a practical viewpoint [19]. In this segment, we briefly explain the Alamouti transmit diversity technique, comprising of encoding and decoding procedures, and its robust applications in BER calculations for the BPSK and QAM systems. Prior to BER analysis, we first discuss the speech symbol encoding and decoding process using Alamouti coding. To evaluate the scheme performance, we consider a 1×2 MISO system with two transmitters and one receivers.

4.4.1.1 Encoding of speech symbols

Fig. 4.5 represents the block diagram of the Alamouti space-time encoder.

The speech symbols are grouped in chunks of m information bits, $m = \log 2M$, where M is the total number of speech symbols. Each group of information bits is then modulated using various digital modulation schemes such as BPSK, QAM, and MSK. The Alamouti encoder accepts a block of two modulated speech symbols x_1 and x_2 in each encoding process. Each speech symbol is then mapped to the transmitter antennas in accordance with the code matrix given in Eq. (4.3).

**FIG. 4.5**

Alamouti space-time encoder.

$$X = \begin{bmatrix} x_1 & -x_2^* \\ x_2 & x_1^* \end{bmatrix} \quad (4.3)$$

The encoded speech symbols are then transmitted in two transmission intervals from the two transmit antennas. In the first transmission interval the speech symbols x_1 and x_2 are transmitted concurrently from antenna 1 and antenna 2, respectively. In the second transmission interval, the speech symbols $-x_2^*$ is transmitted from antenna 1 and speech symbol x_1^* from the transmit antenna 2. * represents the complex conjugate of the symbols. The transmitted speech symbol sequences from the two antennas are represented as in Eq. (4.4).

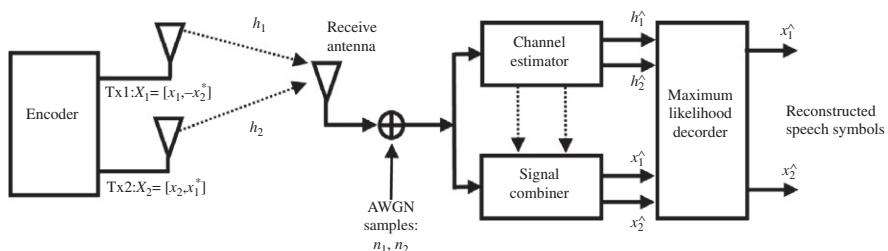
$$\begin{aligned} X_1 &= [x_1, -x_2^*] \\ X_2 &= [x_2 \ x_1^*] \end{aligned} \quad (4.4)$$

It is evident that with the Alamouti implementation, the transmitted speech symbol sequences x_1 and x_2 are orthogonal, because the inner product of the symbol sequences is zero.

4.4.1.2 Decoding of speech symbols

The speech symbol reconstruction process at the receiver is done in accordance with the block diagram shown in Fig. 4.6.

On the receiver side, the received speech symbols over adjacent channels and consecutive intervals are denoted as r_1 arriving in time t and r_2 arriving in time $t+T$, where T is the symbol period.

**FIG. 4.6**

Alamouti space-time decoder.

$$\begin{aligned} r_1 &= h_1 x_1 + h_2 x_2 + n_1 \\ r_2 &= -h_1 x_2^* + h_2 x_1^* + n_2 \end{aligned} \quad (4.5)$$

h_1, h_2 are the complex fading coefficients from antenna 1 and antenna 2 to the receive antenna at t . n_1, n_2 are the Additive White Gaussian Noise (AWGN) samples at t and $t+T$ intervals, respectively. Since the Alamouti coding scheme does not require the CSI for demodulation process this means that all the received speech samples are equi-probable.

A maximum likelihood decoder employs a pair of symbols x_1^*, x_2^* to minimize the distance metric $d^2(r_1, h_1 x_1^* + h_2 x_2^*) + d^2(r_2, -h_1 x_2^* + h_2 x_1^*) = |r_1 - h_1 x_1^* - h_2 x_2^*|^2 + |r_2 + h_1 x_2^* - h_2 x_1^*|^2$ over x_1 and x_2 . The receiver thus reconstructs two disjoint decision statistics based on the linear combination of the received signal.

4.4.2 Orthogonal space-time block code

The Orthogonal Space-Time Block Code (OSTBC) scheme is a promising coding technique for the baseband processing of the speech signal transmission and reconstruction. This scheme employs full spatial diversity order and accepts symbol-wise Maximum Likelihood (ML) decoding algorithms. This coding scheme proved very useful for the robust transmission and detection of speech symbols. The encoding and decoding processes are discussed as follows.

4.4.2.1 Encoding of speech symbols

Unlike the Alamouti encoder, the OSTBC encoder design is relatively complex in terms of base-band processing. We proceed with the OSTBC speech symbol encoding with the assumption that there are 2^m symbol points in the input speech sequence to be transmitted. The OSTBC encoder maps a block of km information bits into the sequence constellation to invoke k modulated speech symbols x_1, x_2, \dots, x_k . The k modulated speech symbols are encoded by the OSTBC encoder to produce n_T concurrent symbol sequences of length p in accordance with code matrix X . The k modulated parallel sequences are transmitted via n_T transmit antennas concurrently in p time instants. The block diagram for the OSTBC encoding of the speech symbol sequences is depicted in Fig. 4.7.

The code matrix X comprises of linear combinations of the k modulated speech symbols x_1, x_2, \dots, x_k and their complex conjugates $x_1^*, x_2^*, \dots, x_k^*$. For full

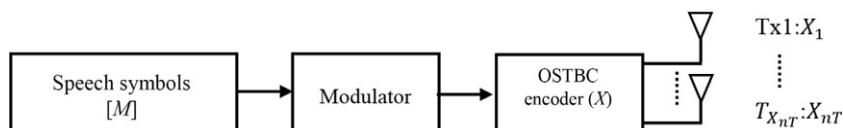


FIG. 4.7

OSTBC encoder.

transmit diversity, X is constructed on the basis of orthogonal designs such that $XX^H = c(|x_1|^2 + |x_2|^2 + \dots + |x_k|^2)$, where c is a constant, X^H represents the Hermitian of X , and I_{nT} is an $n_T \times n_T$ identity matrix. The rows of the code matrix X_{nT} are orthogonal to each other, implying that the speech symbols from any two adjacent transmit antennas are orthogonal. This orthogonality principle mitigates the Intersymbol Interference (ISI) and enhances the BER performance.

4.4.2.2 Decoding of speech symbols

The OSTBC decoder consists of a spatial combiner which spatially combines the speech components from all the receive antennas. The concatenated received speech symbols are then fed into the channel estimator to extract the speech information bits that were encoded through an OSTBC encoder at the transmitter side. The received speech symbols received at the two receive antennas are represented as r_1 and r_2 , respectively. The combiner generates a vector \vec{r} , which is the linear combination of the received speech symbols given in Eq. (4.6).

$$\vec{r} = [H_1^* r_1 + H_2^* r_2] \quad (4.6)$$

where $H = [H_1, H_2]$ represents the complex channel coefficients in vector notation, and * represents the complex conjugates of the individual channel coefficients. The received speech symbol vector is then subjected to the ML decoding algorithm to extract the transmitted speech information symbols x_1, x_2, \dots, x_k . [20]. The OSTBC decoder block diagram is shown in Fig. 4.8.

4.4.3 Maximal ratio combining

Maximal Ratio Combining (MRC) is a common beam-forming method of diversity combining in which:

1. the speech signals from multipath channels are summed up together;
2. the gain of each channel is made proportional to the root mean square signal level and inversely proportional to the mean square noise level in that channel; and
3. different proportionality constants are used for each channel.

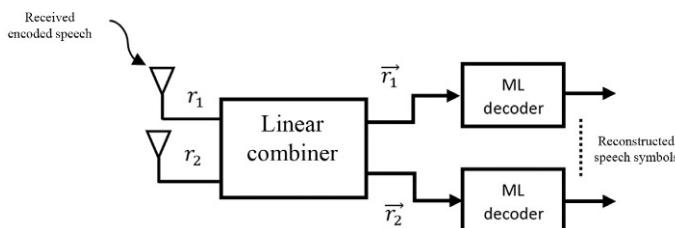


FIG. 4.8

OSTBC encoder.

For the proposed scheme, we consider a receiver endowed with N antennas. The received speech vector is represented as $y = hs + n$, where n is the noise vector. Following the ML detection criterion, the detection procedure may be written as in Eq. (4.7).

$$\hat{s} = \operatorname{argmin}_{\text{scheme}} |\hat{s} - s|^2 \quad (4.7)$$

where \hat{s} is the least square solution to the above model represented in Eq. (4.8). The least square solution on this case is known as Maximal Ratio Combining.

$$\hat{s} = (h^* h)^{-1} h^* y \quad (4.8)$$

The above solution reflects that the speech signals from each antenna are rotated and weighted according to the phase and strength of the channel, such that all the antennas are combined to yield the maximum ratio between the speech symbols and noise. All symbols are coherently combined in a vector with N_R . The MRC technique maximizes the output signal-to-noise ratio (SNR) for the intended user equipment [21].

$$SNR = \gamma = \frac{|W^H h|^2}{E|W^H n|^2} = \frac{|W^H h|^2}{\sigma^2 E|W^H W|} \quad (4.9)$$

By the Cauchy-Schwartz inequality, it is found that SNR is maximized when $W \propto h$.

4.5 Bit error rate (BER) analysis of BPSK modulated system

The probability of error in a BPSK modulated system with an AWGN channel is given as follows:

$$P_{e,BPSK} = Q\left(\sqrt{\frac{2Eb}{N_0}}\right) \quad (4.10)$$

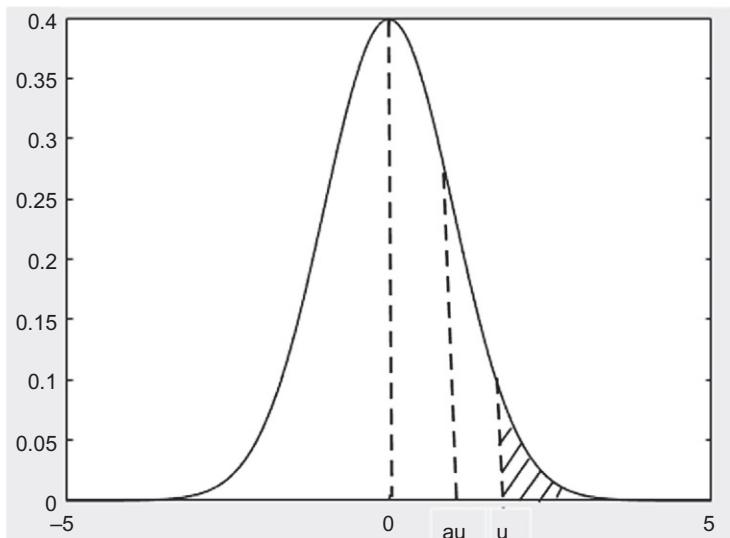
where u is the complementary function; if we want the BER to decrease, “ u ” should increase for more pronounced SNR. Various techniques like transit power scaling, diversity order enhancement, etc. can be employed to decrease the BER. In short, BER reduction implies increasing the value of “ u .” In the context of a fading environment, the probability of error in BPSK with fading parameter “ α ” is given as follows:

$$P_{e,BPSK,fading} = Q\left(\alpha\sqrt{\frac{2Eb}{N_0}}\right) \quad (4.11)$$

α has the probability of being >1 or <1 , but the higher probability of being less than $\alpha < 1$ implies that the SNR now shifts to the left, as shown in Fig. 4.9. Left shifting of SNR makes the probability of error worst. BER in a fading environment has two dependencies and can be defined implicitly as follows:

$$BER_{fading} = f\left(\frac{Eb}{N_0}, \alpha\right) \quad (4.12)$$

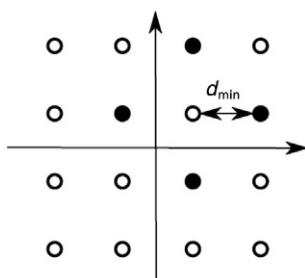
where $\frac{Eb}{N_0}$ represents the SNR and α is the fading parameter.

**FIG. 4.9**

Probability density function (PDF) of Gaussian noise.

4.6 Bit error rate (BER) analysis of a QAM modulated system

For the transmission of the speech symbols over a Gaussian channel, the QAM plays a vital role. QAM is among the most commonly used modulation schemes in multi-carrier systems like Orthogonal Frequency Division Multiplexing (OFDM). The QAM modulation scheme involves a higher data rate, generally higher than the bandwidth efficiency, at the cost of power utilization. To maintain a low BER in a QAM modulated system, we require higher SNR. The QAM scheme consists of two modulations: in-phase (real) and quadrature (imaginary). This is due to the variations in both amplitude as well as phase. The constellation of a generalized M-ary QAM system is shown in figure below. Each symbol in the phase and quadrature planes is separated by a minimum possible distance d_{min} to avoid Inter Symbol Interference (ISI).



Since the in-phase and the quadrature components of a QAM system are independent, the probability of the speech symbol being correctly demodulated is given as

$$p_c = (1 - p'_e) \quad (4.13)$$

where p_e is the probability of symbol error either in-phase or quadrature component. The BER of QAM system is given as follows:

$$p_e = (1 - p_c) \quad (4.14)$$

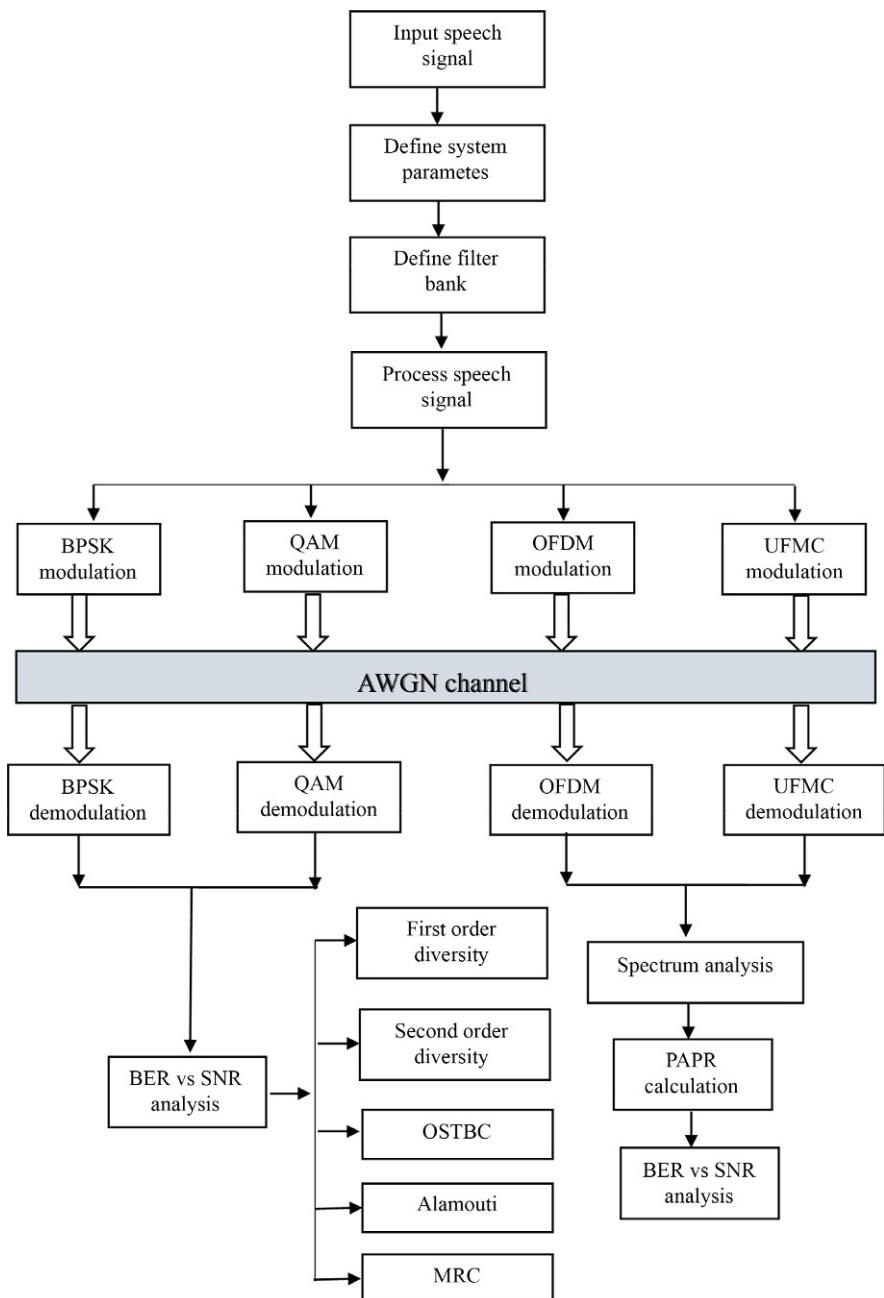
In addition, the probability of error is

$$p_e = 1 - \left[1 - \frac{2(\sqrt{M}-1)}{\sqrt{M}} Q\sqrt{\frac{3E_{av}}{2(M-1)N_0}} \right]^2 \quad (4.15)$$

where M is the order of QAM system, E_{av} is the power density of the speech symbols, and N_0 is the Gaussian noise variable with mean 0 and variance 0.5. The M-ary QAM modulated system acquires a higher data transmission rate by suffering BER degradation.

4.7 Proposed technique

The block diagram of the proposed technique was shown in Fig. 4.3. In this chapter, Bit Error Rate Analysis (BER) of a test recorded speech signal is carried out in accordance with increasing the signal-to-noise ratio (SNR). A speech signal of length 1.28 s is processed and converted into digital form. BER analysis of the same speech signal using various modulation schemes like Binary Phase Shift Keying (BPSK), Quadrature Amplitude Modulation (QAM), Orthogonal Frequency Division Multiplexing (OFDM), and Ultra-Filter Bank Multicarrier Modulation (UFMC) is carried out. The aim of the proposed work is to find out the efficient modulation technique with lesser BER for transmission of speech signal in a wireless communication system. The processed speech signal is simultaneously modulated using BPSK, QAM, OFDM, and FBMC. Each of the subcarriers assigned with some complex data symbols is sent to the receiver through an Additive White Gaussian Noise (AWGN) channel. At the receiver, each of the subcarriers is demodulated successfully using a BPSK demodulator, QAM demodulator, OFDM demodulator, and UFMC demodulator, respectively. The demodulated signal is further analyzed and processed to retrieve the original speech signal. Moreover, the spectrum of a test recorded speech signal is analyzed and compared with the OFDM and UFMC modulation techniques. Each of the subcarriers assigned with some complex data symbols is sent to the OFDM receiver through the AWGN channel. At the receiver, each of the subcarriers is demodulated successfully without ISI or ICI due to the presence of Cyclic Prefix. The same speech signal is then modulated with UFMC modulation at the transmitter and demodulated using the UFMC receiver. The implementation of the UFMC transmitter and receiver is shown in Fig. 4.10. The original and

**FIG. 4.10**

Block diagram of the proposed technique.

received speech signals are compared and BER analysis for various modulation schemes using various diversity orders is calculated. In addition, BER comparison spectrum analysis of speech signal using OFDM and UFMC modulation is also performed and analyzed. As the proposed work is suggested to be used in 5G networks, the Peak-to-Average Power Ratio (PAPR) of OFDM and UFMC is also calculated.

System parameters used in the proposed work

numFFT = 1024	% number of FFT points
subbandSize = 20	% must be >1
numSubbands = 10	% numSubbands*subbandSize <= numFFT
subbandOffset = 156	% numFFT/2-subbandSize*numSubbands/2 for band center

Parameters considered for BER analysis

Frame length	100
Number of packets	1000
SNR	[0:2:20]
Number of pilot symbols/frame	8
Diversity orders (Tx, Rx)	(1,2)(1,4)(1,8)(2,4)(2,8) etc.
Modulation schemes	BPSK, QAM,UFMC,OFDM
Combining techniques	Alamouti, OSTBC, MRC

4.8 Simulation results

Fig. 4.11 represents the original waveform of the test recorded speech signal used for analysis in the proposed technique. Figs. 4.12 and 4.13 give the comparison of the plots of the spectral densities for the OFDM and UFMC schemes. The power spectral density of the UFMC transmitted signal is plotted to highlight the low out-of-band leakage. Comparing the plots of the spectral densities for OFDM and UFMC schemes, UFMC has lower side lobes. This permits a higher usage of the allocated spectrum, leading to increased spectral efficiency compared to OFDM modulation. UFMC also shows a slightly better PAPR, as given in Table 4.1.

The Bit Error Rate Analyses as a function of SNR for different modulation schemes at different diversity orders are shown in Figs. 4.14–4.18, respectively. Fig. 4.14 gives BER vs SNR for BPSK modulation using various combining techniques. From the obtained graph, it is clear that BER performance for the MRC technique is better than the other spatial coding techniques. Similarly, in Fig. 4.15, BER for 16-QAM modulation using the MRC technique shows better results than the other techniques. In the 16-QAM modulated system, higher data rate transmission is obtained at the cost of BER performance. Fig. 4.16 gives BER vs SNR for QAM with

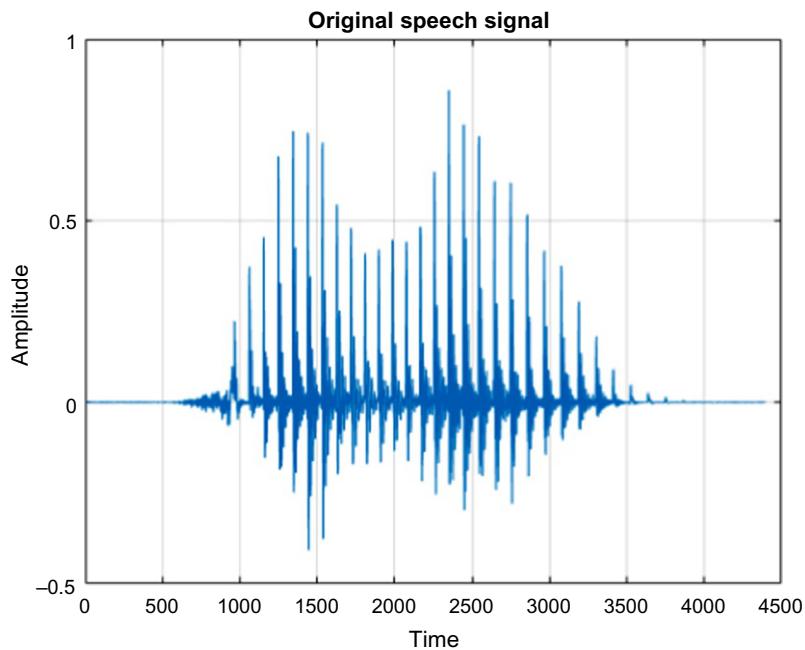


FIG. 4.11

Waveform of original speech signal.

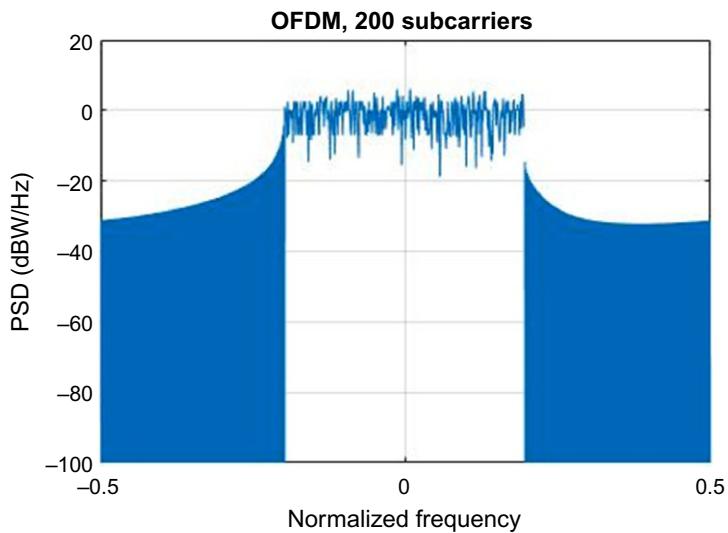


FIG. 4.12

Spectrum analysis of a speech signal using OFDM.

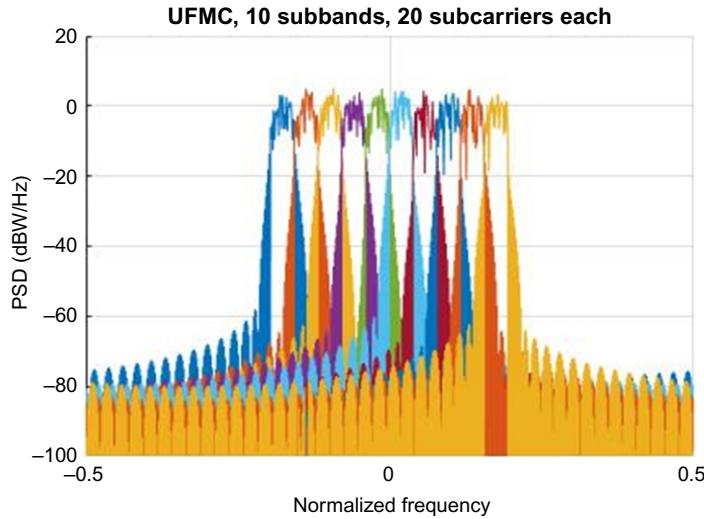


FIG. 4.13

Spectrum analysis of a speech signal using UFMC.

Table 4.1 PAPR comparison of OFDM and UFMC techniques.

PAPR	
UFMC	8.1320
OFDM	8.9211

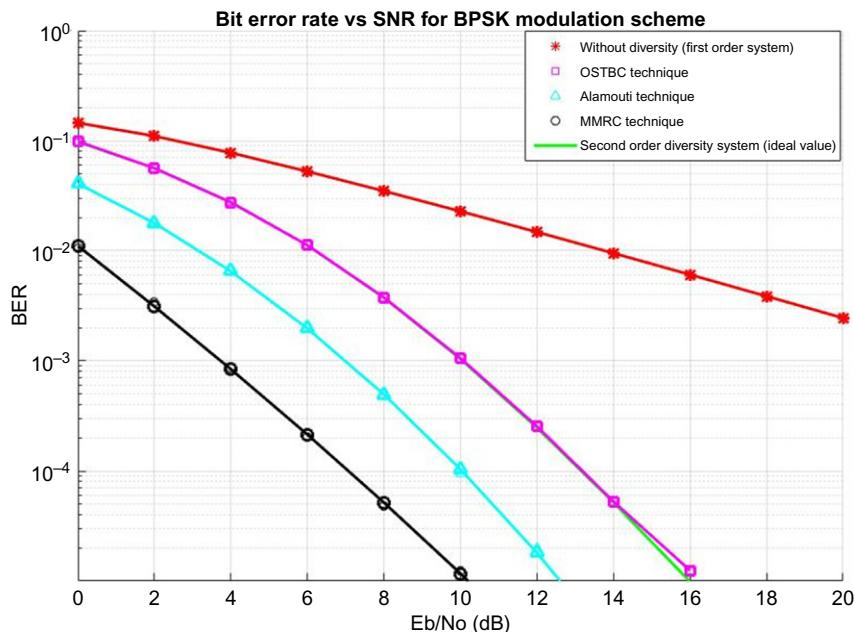


FIG. 4.14

BER vs SNR for BPSK modulation scheme.

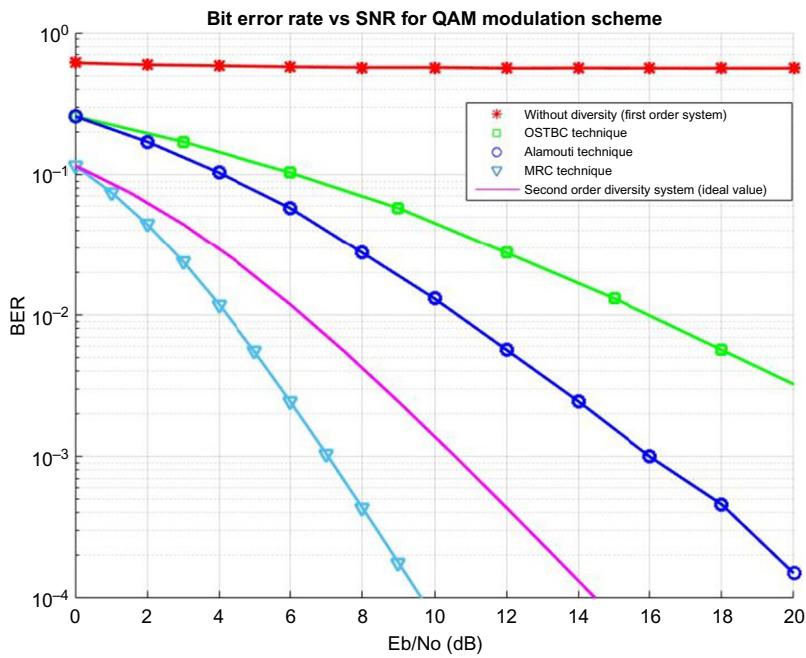


FIG. 4.15

BER vs SNR for 16-QAM scheme.

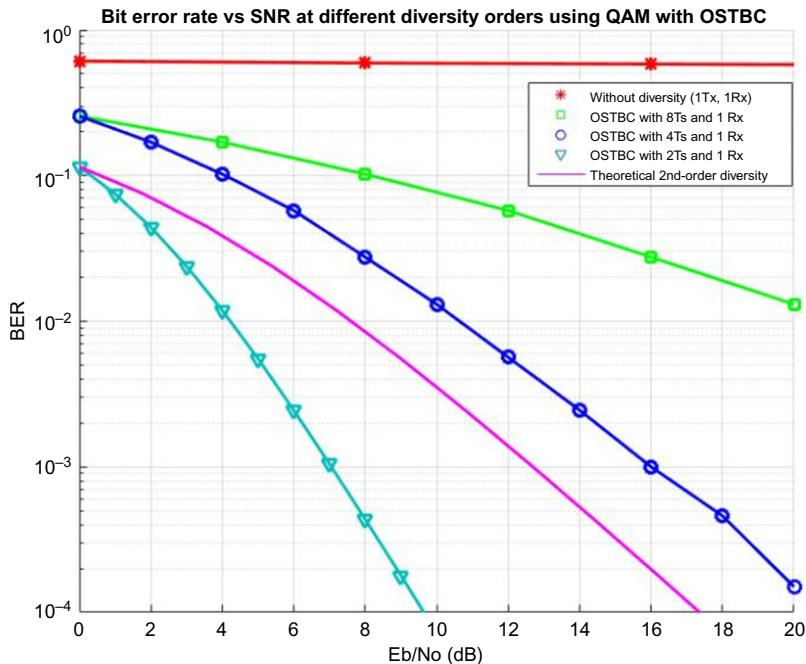
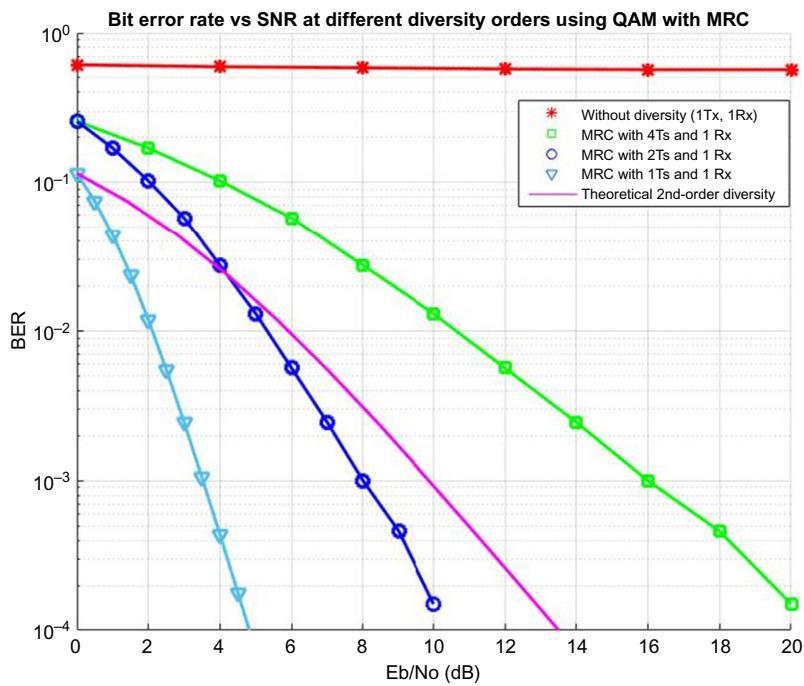


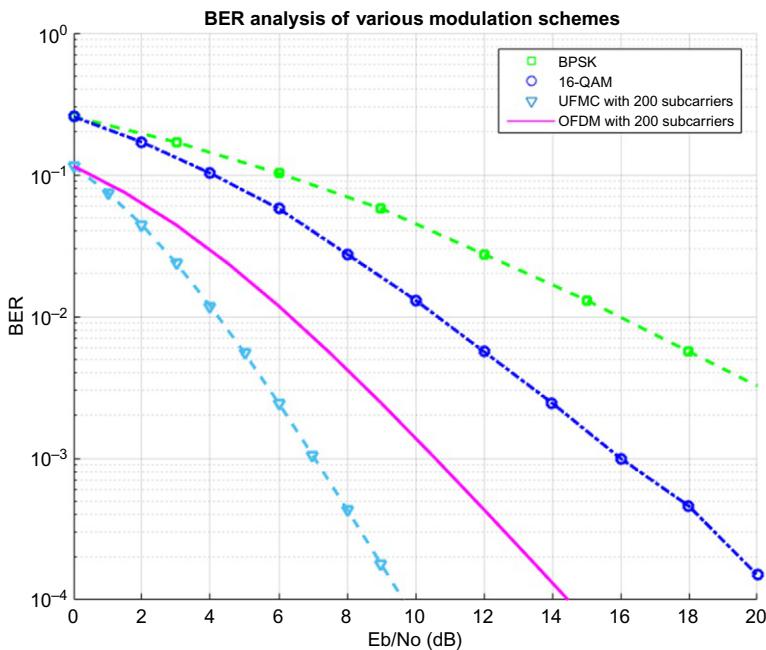
FIG. 4.16

BER vs SNR at different diversity orders using OSTBC.

OSTBC at different transmit diversity orders. From the obtained graph, it is clear that the BER performance degrades with increasing transmit diversity orders. This is due to the secondary user interference at the transmitter side. The performance can be improved by employing secondary user interference mitigation techniques like zero-forcing and precoding at the baseband side. Fig. 4.17 gives BER vs. SNR for QAM with MRC at different receive diversity orders. The obtained result reveals better BER performance with the increase in receive diversity order. MRC with 1Tx and 4Rx shows better BER performance compared to single receive antenna systems. With multiple receive antennas, the probability of a deep fade event reduces and hence the signal reception quality improves. Fig. 4.18 shows the BER comparison of various modulation schemes used in our proposed technique. From the obtained graph, it is clear that BER performance for the UFMC technique is much better than the other techniques.

**FIG. 4.17**

BER vs SNR at different diversity orders using QAM with MRC.

**FIG. 4.18**

BER vs SNR for different modulation techniques.

4.9 Conclusion

The proposed technique gives the BER analysis for various modulation schemes. It highlights the merits of Ultra-Filter Bank Multicarrier modulation (UFMC) as a new candidate for 5G networks and compares it with the existing OFDM and other modulation techniques within the generic framework. UFMC is seen as a generalization of Filtered-OFDM and FBMC modulation techniques. In OFDM, the entire band is filtered; and in FBMC, individual subcarriers are filtered. In contrast, in UFMC, groups of subcarriers are filtered. In this chapter, BER comparison at different transmit and receive diversity orders using different spatial coding techniques is carried out. From the obtained results, it is clear that the MRC technique with receive diversity has better BER performance compared to the OSTBC and Alamouti techniques. In addition, among various signal processing schemes used in this chapter, the BER performance of UFMC shows better results when compared to BPSK, QAM, and OFDM modulation techniques. Hence it can be concluded that with UFMC, 5G networks will be able to have higher data rates, lower latency, and more efficient spectrum usage.

References

- [1] G. Wunder, M. Kasparick, S. ten Brink, F. Schaich, T. Wild, I. Gaspar, E. Ohlmer, S. Krone, N. Michailow, A. Navarro, G. Fettweis, D. Ktenas, V. Berg, M. Dryjanski, S. Pietrzyk, B. Eged, 5GNOW: challenging the LTE design paradigms of orthogonality and synchronicity, in: IEEE 77th Vehicular Technology Conference, 2013, pp. 1–5, <https://doi.org/10.1109/VTCSpring.2013.6691814>.
- [2] S. JG Andrews, W.C. Buzzi, S.V. Hanly, A. Lozano, A.C.K. Soong, J.C. Zhang, What will 5G be? IEEE J. Sel. Areas Commun. 32 (6) (2014) 1065–1082, <https://doi.org/10.1109/JSAC.2014.2328098>.
- [3] Z. Wang, G.B. Giannakis, Wireless multicarrier communications, IEEE Signal Process. Mag. 17 (3) (2000) 29–48, <https://doi.org/10.1109/79.841722>.
- [4] T. Wild, F. Schaich, Y. Chen, 5G air interface design based on universal filtered UF-OFDM, in: 19th International Conference on Digital Signal Processing, 2014, pp. 699–704, <https://doi.org/10.1109/ICDSP.2014.6900754>.
- [5] J.A. Sheikh, S. Akhter, S.A. Parah, G.M. Bhat, Blind digital speech watermarking using filter bank multicarrier modulation for 5G and IoT driven networks, Int. J. Speech Technol. (2018), <https://doi.org/10.1007/s10772-018-9541-6>. Springer.
- [6] M. Van Eeckhaute, A. Bourdoux, P. De Doncker, F. Horlin, Performance of emerging multi-carrier waveforms for 5G asynchronous communications, EURASIP J. Wirel. Commun. Netw. 2017 (2017) 29, <https://doi.org/10.1186/s13638-017-0812-8>. Springer.
- [7] K. Krishna Kishore, P. Rajesh Umar, V. Jagan Navee, Comprehensive analysis of UFMC with OFDM and FBMC, Indian J. Sci. Technol. 10 (17) (2017), <https://doi.org/10.17485/ijst/2017/v10i17/114337>.
- [8] B.R. Tomiuk, N.C. Beaulieu, A new look at maximal ratio combining, in: Globecom'00—IEEE Global Telecommunications Conference. Conference Record (Cat. No.00CH37137). IEEE, 2000.
- [9] P. Tripathi, S. Dixit, R. Agarwal, M. Shukla, Maximal ratio combining diversity technique for IDMA systems, in: 2015 Fifth International Conference on Communication Systems and Network Technologies, 2015. [https://doi.org/10.1109/CSNT.2015.978-1-4799-1797-6/15 \\\$31.00 © 2015 IEEE](https://doi.org/10.1109/CSNT.2015.978-1-4799-1797-6/15 \$31.00 © 2015 IEEE).
- [10] M. Kasparick, Y. Chen, J.-B. Doré, M. Dryjanski, I.S. Gaspar, 5G waveform candidate selection D 3.2. Technical report, 5G now, http://www.5gnow.eu/wp-content/uploads/2015/04/5GNOW_D3.2_final.pdf, 2014. Accessed 30 December 2016.
- [11] A. Aminjavaheri, A. Farhang, A. RezazadehReyhani, B. Farhang-Boroujeny, Impact of timing and frequency offsets on multicarrier waveform candidates for 5G, in: IEEE Signal Processing and Signal Processing Education Workshop, 2015, pp. 178–183, <https://doi.org/10.1109/DSP-SPE.2015.7369549>.
- [12] M.G. Bellanger, Specification and design of a prototype filter for filter bank based multicarrier transmission, in: Proc. IEEE Int. Conf. Acoust. Speech, Signal Process. (ICASSP), vol. 4, Salt Lake City, UT, May, 2001, pp. 2417–2420.
- [13] F. Schaich, T. Wild, Y. Chen, Waveform contenders for 5G—suitability for short packet and low latency transmissions, in: 79th IEEE Vehic. Tech. Conf. 2014, IEEE, 2014, pp. 1–5.
- [14] G. Wunder, P. Jung, M. Kasparick, T. Wild, F. Schaich, Y. Chen, S. Brink, I. Gaspar, N. Michailow, A. Festag, et al., 5GNOW: non-orthogonal, asynchronous waveforms for future mobile applications, IEEE Commun. Mag. 52 (2) (2014) 97–105.

- [15] G. Wunder, M. Kasparick, S. Brink, F. Schaich, T. Wild, I. Gaspar, E. Ohlmer, S. Krone, N. Michailow, A. Navarro, et al., 5GNOW: challenging the LTE design paradigms of orthogonality and synchronicity, in: *Proc. Vehicular Tech. Conf.*, Dresden, Germany, 2013.
- [16] V. Vakilian, T. Wild, F. Schaich, S. Ten Brink, J.F. Frigon, Universal-filtered multi-carrier technique for wireless systems beyond LTE, in: *Globecom Workshops*, IEEE, 2013, pp. 223–228.
- [17] F. Schaich, T. Wild, Waveform contenders for 5G—OFDM vs. FBMC vs. UFMC, in: 6th Int. Symp. Commun. Cont. Sig. Proc. (ISCCSP), 2014, IEEE, 2014, pp. 457–460.
- [18] S.A. Cheema, K. Naskovska, M. Attar, B. Zafar, M. Haardt, Performance comparison of space time block codes for different 5G air interface proposals, in: *WSA 2016; 20th International ITG Workshop on Smart Antennas (Munich)*, 2016, pp. 1–7.
- [19] M. Renfors, T. Ihälainen, T.H. Stitz, A block-Alamouti scheme for filter bank based multicarrier transmission, in: 2010 European Wireless Conference (EW), 2010, pp. 1031–1037, <https://doi.org/10.1109/EW.2010.5483517>.
- [20] J.A. Sheikh, S. Akhtar, S.A. Parah, G.M. Bhat, A new method of speech transmission over space time block coded co-operative MIMO-OFDM networks using time and space diversity, *Int. J. Speech Technol.* 21 (1) (2018) 65–77.
- [21] R. Zakaria, D.L. Ruyet, M. Bellanger, Maximum likelihood detection in spatial multiplexing with FBMC, in: 2010 European Wireless Conference (EW), 2010, pp. 1038–1041, <https://doi.org/10.1109/EW.2010.5483520>.

This page intentionally left blank

PART

Smart cities/smart
homes/smart
communities

2

This page intentionally left blank

Study of robust language identification techniques for future smart cities

5

**Ravi Kumar Vuddagiri, Krishna Gurugubelli, Ramakrishna Thirumuru,
Anil Kumar Vuppala**

*Speech Processing Laboratory, LTRC, KCIS, International Institute of Information Technology,
Hyderabad, India*

5.1 Introduction

A smart city is an area that utilizes the data collected from different sorts of sensors to maintain resources efficiently. The data collected from the various sensors can be analyzed to monitor and control the traffic and transportation systems, water supply management, waste management, institutions, offices, libraries, dispensaries, power plants, information systems, intelligent automobiles, smart home appliances, and other community services. Moreover, advancements in mobile and speech technologies enabled the use of speech-based devices in the development of smart homes and smart cities. It can be fore casted that in future smart cities, people may access their electronic devices remotely (e.g., workplaces, shopping malls, during traveling on roads) through mobile voice commands; on the other hand, the smart cities have technology in such a way that one can listen to a lecture or movie or speech in one's native language; further, it can also be predicted that smart cities can assist nonnative residents and tourists in their respective native languages. In this scenario, speech technology can play a significant role in the development of smart cities. The most familiar speech technologies used in smart cities are multilingual speech recognition and speech synthesis, multilingual dialog systems, language identification, speaker verification, etc. One of the challenges of speech technologies used in smart cities can be the poor performance of that technology in real-time environments. This study investigates performance language identification systems in mobile and noisy environments.

Language identification (LID) is a process of recognizing the natural language (e.g., Hindi, English, French) from the spoken utterance. Over the last two decades, spoken language identification has received interest from the research community to be used as an ancillary technology for many multilingual services. For example, speech technologies like multilingual dialog systems, spoken language translation systems, multilingual speech recognition systems, and spoken document retrieval systems utilize the LID system as a front-end supporting technology. In [1–6], the development of the spoken language identification systems has been presented.

From the literature, it can be observed that there are three major issues concerning the performance of the LID system, these are varying background conditions, coding, and transmission errors. In the literature [7], the effects of speech coding on speaker and language recognition systems are investigated. The spoken utterance may contain information about the message, language, speaker emotion, age, gender, etc. The LID system has to be invariant, with all the other information, in addition to noise. The degradation in the performance of the LID system in varying background environments is due mostly to a mismatch in training and operating environments. The present work aims at analyzing the performance of LID due to varying background conditions and mobile environments (coding). In this work, three major approaches that have been explored to develop LID systems to improve the performance in varying background conditions are considered. They are:

- front-end noise enhancement methods;
- vowel region-based robust feature extraction; and
- inducing noise into the training corpus.

In the first approach, preprocessing level noise enhancement methods improving the LID system performance are studied. This approach uses a clean dataset in developing the LID system. During the evaluation time, if the test utterance has background noise, the front-end enhancement scheme enhances the noisy speech, and the enhanced speech signal is used to obtain the identity of a language. The performance of these approaches depends largely on the efficacy of enhancement methods employed. In the second approach, the vowel region-based robust feature extraction scheme is employed. From the literature, it is observed that vowel regions are the high signal-to-noise regions in the speech signal. Moreover, the phonotactic information is adequately maintained in the vowel regions. In this regard, a robust vowel region detection scheme is adopted as a front-end to the LID system. The features computed in these vowel regions are used in the LID system. In the second approach, a robust feature extraction scheme is considered. While in the last method a novel training strategy was investigated to improve the performance of the LID system, this final approach hypothesizes that the degradation in the performance of LID system operating in varying background environments is due to the mismatch in training and operating environments. To reduce the mismatch condition, simulated noisy and coded speech is induced into the training corpus. In this regard, different training strategies are investigated. Though these approaches have performed better in noisy and coded speech environments, their performance is superior when there is a closer match between the training and operating environments. Moreover, the real-life environments are complex, mixed, and time-varying, and are hard to simulate. It is more likely to have the mismatch between the training and operating environments. With this motivation, an LID system is developed that can operate reasonably well even in the presence of a mismatch between training and operating environments.

In this work, to reduce the mismatch between training and operating environments, multi-SNR models have been analyzed to develop LID systems. In this regard, the training data are augmented with multiple versions of synthetically generated noisy data by adding noise to the training corpus at different SNR levels. The augmented dataset is used for developing LID systems. The training method plays a crucial role in the convergence, generalization, and performance of neural networks. The use of curriculum learning (CL)-based training strategies have been explored for a variety of applications such as probabilistic linear discriminant analysis for noise-robust speaker recognition in [8, 9]. In this study, CL-based learning schedules have been explored in training multi-SNR models for developing the robust LID system. Here the CL refers to the task of training a neural network with examples in some specific, meaningful order rather than randomly sampling the training dataset. The efficient CL strategy can help the model to converge very fast and eases the optimization of the neural network [10, 11].

The rest of the chapter is organized as follows: in [Section 5.2](#), related works on LID are described. The database used in this work is described in [Section 5.3](#). [Section 5.4](#) describes the performances of various baseline LID systems and their performances in varying background environments. In [Section 5.5](#), the performance of LID for coded speech is explained. [Section 5.6](#) presents proposed work to improve the performance of LID systems, using vowel region detection, spectral subtraction and minimum mean square error (MMSE) enhancements as a front-end preprocessing method, and various CL strategies for developing robust LID systems. A conclusion and future scope are presented in [Section 5.7](#).

5.2 Related works

In the development of LID systems, the initial studies were inspired by the speaker identification frameworks. The majority of LID systems are classified into two types: explicit and implicit. The implicit approaches directly use the acoustic-level information to model the LID system [2]. The explicit LID systems initially transform the acoustic sequences to an intermediate representation such as phones, senones, or tokens and the temporal relations among them are used for developing LID systems. The LID systems that use phonotactic, phone frequency, and syntax information are described in [12, 13]. These methods use a phone-level speech recognition system followed by language modeling to model the temporal relations. Language models like SRILM and RNNLM have been explored to model the temporal relations among the tokens for developing large-scale LID systems [13]. In this work, implicit techniques for language identification are investigated.

In [14, 15], Gaussian mixture models (GMM) and Gaussian mixture models with a universal background model (GMM-UBM) are trained using spectral features for modeling LID systems. In [16], the roles of CV units and steady vowels in language identification are explored. The sonorant regions of speech are explored to train the GMM-UBM-based LID system in [17]. From the literature, it can be observed that a

kind of aggregation of acoustic-level information to model long-term temporal information can be beneficial in developing LID systems. The i-vector representation of speech can summarize the utterance-level information. Further, the i-vector representation converts the variable length sequence to fixed dimension continuous representation, which can be used as a feature to train the LID system. In addition, the i-vector-based system can model the temporal context effectively [18] in the development of LID systems. The performances of LID systems depend on the duration of the test utterances, that is, i-vectors have been modified for developing LID systems that can operate on utterances with short duration [19]. In [20], i-vector with a PLDA scoring scheme was investigated for robust LID systems in noisy environments. In [19, 21, 22], deep neural network (DNN)-based LID systems have been explored. These models have shown a significant improvement in LID system performance. At the same time, the performance of DNN-based LID systems depends on the size of the dataset. The improvement in performance is observed with massive datasets. In [23, 24], multilingual bottleneck features are explored for developing LID systems. The recent developments in neural networks have influenced the performance of LID systems, and have enriched the capabilities of neural networks to process the whole utterance. Such networks have been employed for training LID systems. In [25, 26], recurrent neural networks and convolution neural networks have been explored in the training of LID systems. Though sequential models like recurrent neural networks and long short-term memory networks performed better, they are not parallelizable. To address this, a self-attention mechanism is used to convert the variable length sequence to a fixed dimension vector and the fixed dimension representation is used to discriminate the languages. A feed-forward architecture with self-attention mechanism is proposed in [27, 28]. In this framework, the whole network can be trained as a single-framework through back-propagation. Though the DNN with self-attention (DNN-WA)-based LID systems model the entire utterance, they are sensitive to the utterance length. The performance of DNN-WA-based LID systems degrades when the utterance length varies. Further, the performance of LID systems can be influenced by the diverse background and mobile environments. In [29], the importance of vowel regions in LID is investigated in the presence of noise. To improve the LID performance in these mismatched conditions, CL-based training mechanisms are explored in [17, 30].

5.3 Database

In this study, the results are reported on the oriental language recognition challenge (AP17-OLR) database [31, 32]. The NSFC M2ASR project and Speech Ocean together developed the AP17-OLR database. It includes multiple languages spoken in east, northeast, and southeast Asia and consists of 10 languages—Korean in Korea, Cantonese on the mainland of China and Hong Kong, Uyghur, Kazakh, and Mandarin in China, Vietnamese in Vietnam, Russian in Russia, Japanese in Japan, and Indonesian in Indonesia. In total, 10 h of speech data were collected in

reading style for each language. From each language, 1800 utterances were retained for test dataset, and the rest of the utterances were utilized for the training dataset. In this study, the experiments were performed at a sampling rate of 16,000 Hz. Further, the NOISEX database is used in this study to generate the noisy data. The simulated noise speech utterances are used to analyze the LID system performance in noisy conditions. In addition, the performance of LID systems in telephonic speech environments is simulated by using the adaptive multirate coders available in SOX toolkit.

5.4 Baseline LID system

This work explored the i-vector, DNN, and DNN-WA architectures for the development of LID systems, and these are concisely described in the subsequent sections. Further, a 56-dimensional shifted delta cepstral (SDC) feature representation is used for developing LID systems [33]. In this study, the performances of LID systems are evaluated by using an equal error rate (EER). The EER is computed for each language and the average for the same is reported.

5.4.1 The i-vector-based LID system

In this work, an i-vector-based LID is considered as one of the baseline method. In this regard, the UBM is trained with 2048 Gaussian components using a 56-dimensional SDC feature representation derived from the MFCC features. A 400-dimensional total variability matrix (T) is employed in this i-vector representation. The T matrix is optimized for 10 expectation maximization iterations. Further, a principal component analysis and linear discriminant analysis are performed over the total variability matrix T to reduce the dimensionality. Later, the probabilistic linear discriminant analysis is done to get scores of the language identity. In [18], the joint factor analysis used in i-vector representation is described. The mathematical representation of the i-vector is given by

$$M_L = m_l + T w_l \quad (5.1)$$

where M_L is the super-vector obtained from the stacked mean vectors. The mean vectors are derived from the GMM-UBM modeling. The m_l is the super-vector obtained from the UBM model and w_l refers to i-vector having a standard normal prior. The baseline results with i-vector representation are provided in [Table 5.1](#).

5.4.2 LID system using DNN

In the literature, LID systems have been developed using convolution neural networks [34] and shallow architectures [35]. However, to get the advantages like discriminative capability and parallelizability, in this work DNNs-based LID systems are considered as the second baseline system. The DNN architecture used comprises

Table 5.1 EER of baseline i-vector, DNN, and DNN-WA LID systems developed using OLR database.

Language	kazak	tibet	uyghu	ct-cn	id-id	jp-ja	ko-kr	ru-ru	vi-vn	zh-cn	Average
i-vector	18.00	4.99	u4.52	18.65	32.34	10.15	26.15	9.56	8.56	16.22	12.57
DNN	18.00	2.72	6.66	17.04	30.8	10.6	22.38	5.08	4.29	10.57	13.02
DNN-WA	7.85	5	5.19	9.8	29.4	11.16	17	8.77	8	11.35	10.02

of four hidden layers and each layer contains of 700, 500, 200, or 100 units. Further, a rectified linear unit (ReLU) is used as an activation function in the hidden layers and in the output layer a softmax function is used. The network is trained with a stochastic gradient descent optimizer. The performance of the network is optimized using the hyper parameters, namely learning rate, momentum factor, and mini-batch size. The categorical entropy objective function is used to train the network. In this method, a frame-level evidence obtained by the network is averaged over the utterance to obtain the language identity of the utterance. The detailed description of the DNN architecture used developing the LID system is presented in [27, 30]. The results obtained with the DNN-based LID system are provided in [Table 5.1](#). Multiple experiments have been performed and the performances obtained with best-performing hyperparameters (learning rate = 0.001, momentum factor = 0.98, and mini-batch size = 200) are presented in row 3 of [Table 5.1](#).

5.4.3 LID system using DNN with attention

In DNN-based LID systems, the decision on language is taken at every frame and averaged over all frames to get the utterance-level decision. Moreover, the DNN-based LID systems trained over frame-level acoustic features may not model the long-term temporal patterns of speech which captures the language-specific information. To learn the long-term temporal patterns of speech, DNN architectures can incorporate the long-short-term memory networks [36], but due to the sequential nature of these networks, they are computationally slow and are not parallelizable, whereas DNN with an attention network operates in a feed-forward fashion as well as capturing the long-term temporal context. In this work, the DNN-WA model described in [27] has been implemented.

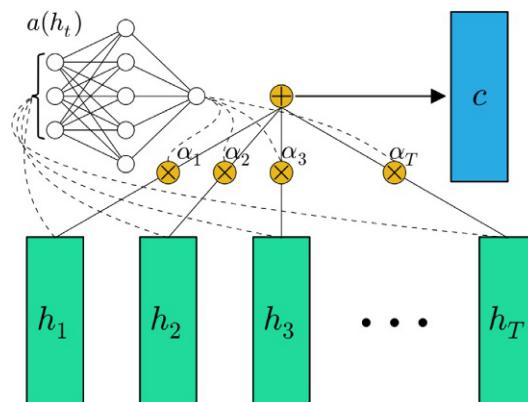


FIG. 5.1

Deep neural network with attention model [37].

The DNN-WA network comprises of four hidden layers with each layer comprising of 700, 500, 200, or 100 units with ReLU activation functions, and the attention layer comprises of a multilayer perceptron (MLP) with a single hidden layer. The entire network is trained end-to-end with a categorical entropy function. The network is trained with a stochastic gradient descent optimizer with learning rate and momentum factor as hyper parameters. The network is trained with variable batch size with respect to each utterance. For every utterance, the batch size is the same as the number of frames that correspond to that utterance. The DNN-WA model uses a 56-dimensional SDC feature representation derived from the MFCC features. The DNN-WA architecture is shown in Fig. 5.1; the output function a hidden layer is given by

$$H = f(x_t, h_t) \quad (5.2)$$

where x_t is the sequence of input feature vectors $\{x_1, x_2, \dots, x_T\}$ and the sequence of the hidden state vectors is $\{h_1, h_2, \dots, h_T\}$. The output of hidden layer h_t is computed by forward pass through regular DNN and a self-attention is computed on these hidden features. The attention mechanism $a(h_t)$ shown in Fig. 5.1 is computed using a single layer perceptron, and then a softmax operation is performed to normalize the values between 0 and 1.

$$\beta = \tanh(W_{wa}h_t + b_{wa}) \quad (5.3)$$

$$\alpha = \text{softmax}(\beta) \quad (5.4)$$

In the previous equations, α is referred to as the attention vector, and W_{wa}, b_{wa} are the parameters of the attention hidden weights, the entire network is optimized along with other parameters of using back-propagation algorithm. The attention-based model computes a “context vector” C_t is given by

$$C_t = \sum_{j=1}^T \alpha_j H_j \quad (5.5)$$

where C_t is the content-weighted mean, the state sequence of H , and T represents the total number of time steps in the input sequence. The output is computed by

Table 5.2 EER of baseline system in varying background environments.

	DNN	DNN-WA	DNN	DNN-WA
Clean	13.02	10.02		
White noise		Vehicle noise		
20 dB	37.46	31.12	30.57	22.19
15 dB	40.25	34.15	33.26	24.56
10 dB	42.64	37.96	37.24	28.46
5 dB	44.23	41.22	41.43	32.11

transforming the context vector C_t using output layer weight V followed by softmax operation

$$y_o = \text{softmax}(VC_t + b_o) \quad (5.6)$$

where b_o is the output layer bias. Note that for the entire input utterance x_t , only a single decision vector y_o is predicted.

The performance of the LID system developed with the DNN-WA network is presented in row 4 of [Table 5.1](#). Columns 2–11 are EERs attained for each language. Column 12 is the average EER for all the languages. From [Table 5.1](#), it can be observed that the DNN-WA-based LID system outperformed the i-vector and DNN-based LID systems in terms of EER.

5.5 Performance of LID in mismatched environments

5.5.1 Language identification in noisy conditions

To investigate the performance of the LID system in noisy environments, the LID systems in this study are developed using clean data and tested with a synthetically generated noisy speech at different SNR levels. In this work, the performances of LID systems are investigated in the presence of white and vehicle noises. The NOISEX dataset is used to generate the noisy speech signals at different SNR levels from 5 to 20 dB at steps of 5 dB.

From [Table 5.2](#), it can be observed that there is a significant drop in the performance of LID systems in noisy conditions. Here LID systems with DNN and DNN-WA are considered in order to analyze the performance of LID systems in noisy conditions. It is observed that the performance of LID systems in white noise is poor compared to that in vehicle noise; this can be attributed to the fact that at a fixed SNR level, the effect of white noise is more significant than vehicle noise.

5.5.2 Language identification in a mobile environment

The rapid developments in mobile and information technology may lead to the use of cell phones in the operation of electronic devices and home appliances remotely through voice commands. The difficulty with the mobile environment is that the speech codecs used in cellular technology degrade the quality of speech. Generally, the speech coders compress the speech signal to utilize the bandwidth of a wireless communication system effectively. From the literature [7], it is understood that the

Table 5.3 Performance of baseline LID systems in mobile environments.

	Clean	AMR-NB	AMR-WB
DNN	13.02	19.12	16.58
DNN-WA	10.02	16.81	13.39

increase in coding rate improves the efficiency of bandwidth utilization, but the quality of speech will become poor. One of the major goals of this work is to study the influence of speech codecs on the performance of LID systems. From the literature, the speech coders like CELP (FS-1016), MELP (TI 2.4 kbps), GSM full rate (ETSI 06.10), and AMR (ITU-T G.722.2) are popular. Since the evolution of 3G and LTE, mobile technologies have tried to improve the speech qualities by using AMR-narrowband and AMR-wideband codecs. The AMR codecs are operated at eight different bit rates in the range of 4.75–12.2 kbps, and are explicitly designed to improve link robustness. The European Telecommunications Standards Institute standardizes the AMR-NB and AMR-WB codecs, and the codecs are adopted in 3G and LTE cellular technology. In this work, we investigated AMR-NB and AMR-WB codecs to analyze the performance of LID systems. To simulate the AMR codecs at different compression levels, this work utilized the SOX toolkit. The speech utterances modified with AMR codecs are used for testing. The performances of LID systems under the mobile environment are set out in [Table 5.3](#).

From [Table 5.3](#), it can be observed that the performance of LID systems (both DNN and DNN-WA architectures) is reduced in coded speech compared to clean speech. Moreover, when the LID performances for the codecs AMR-NB and AMR-WB are compared, the degradation in performance is due more to the AMR-NB codec as it offers low speech quality compared to the AMR-WB codec.

5.6 Proposed approaches for improving performance language identification in mismatched conditions

This work addresses different strategies to improve the performance of the LID systems in adverse conditions like noisy and mobile environments. The approaches to enhance the performance of LID systems are explained in the subsequent sections.

5.6.1 Improving the performance of LID systems by vowel region-based front-end system

Variations specific to language can be sensed at various levels of the speech characterizations. The exact variations of vowels are observed even though the phonemic inventories contain common symbols. Hence, it is hypothesized that vowel regions may contain language-specific information [16]. A front-end vowel region extraction system is used in the current work. Regularly, the vowel region can be considered as a 100-ms period next to the vowel onset point (VOP). In [38], VOP is identified by joint evidence emerging from the excitation-source information, spectral energy, and modulation spectral information. In [39, 40], the resonant frequencies and the glottal closures instants (GCI) can be calculated using the group delay function and zero frequency filtering (ZFF) techniques, respectively. Spectral energy calculated around GCIs is used as a basis for determining VOPs. In another method [41], the vowel region is assessed using the evidence obtained from the predominant spectral-peaks calculated around the synchronized regions of

GCIs obtained from the ZFF signal. This approach identifies both VOPs and vowel end-points (VEPs) as an anchor point to identify vowel regions in continuous speech. In this method, the vowel regions of the speech include vowels, semi-vowels, and diphthongs.

In this work, the vowel region detection technique is used as a front-end framework for language identification, and it is described as follows: the vowel-like regions are highlighted by using the ZFF technique as it essentially emphasizes the low-frequency information of speech. The gross-level abrupt changes in the smoothed temporal envelope obtained from the zero frequency filtered signal can

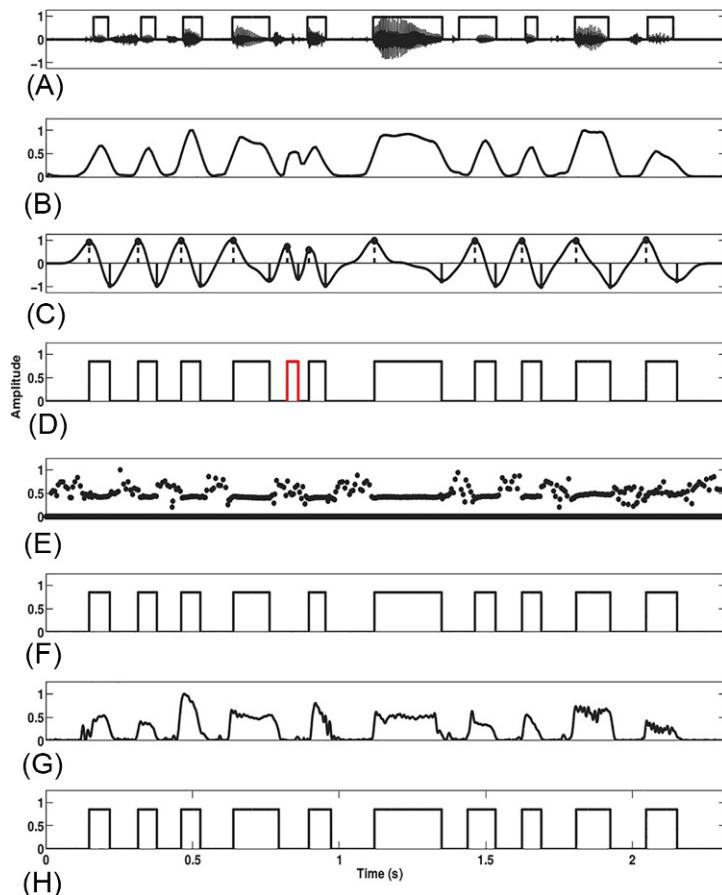


FIG. 5.2

Estimation of vowel region from speech signal using the ZFF method. (a) Speech signal with ground truth. (b) Spectral energy contour. (c) FOGD output highlighted with VOPs and VEPs. (d) Predicted vowel regions. (e) Fundamental frequency contour of the speech signal. (f) Vowel region prediction after the removal of false alarms. (g) The strength of energy contour of a speech signal. (h) Final prediction of vowel regions after postprocessing.

be treated as VOPs and VEPs. An algorithm for the identification of the vowel regions is implemented in two stages. In the first phase, VOPs and VEPs are found. In the second phase, the postprocessing of the same can be done by using the characteristics of vowels, such as uniformity epochs and the strength of the excitation (SoE) profile. During postprocessing of VOPs and VEPs, false vowel regions are removed from the estimated vowel regions and corrected using the strength of the energy contour and epoch uniformity. The algorithmic steps involved in the scheme to detect the vowel region are as follows:

- Compute ZFF to enhance low-frequency content in the speech signal.
- Spectral energy contour is computed by using the sum of 10-largest peaks estimated from the magnitude spectrum. In this regard, a 256-point discrete

Table 5.4 The performance of LID systems with and without vowel region detection-based front-end scheme.

	LID without vowel-region detection scheme		LID with vowel-region detection scheme	
	NB	WB	NB	WB
DNN	19.12	16.58	17.18	14.15
DNN-WA	16.81	13.39	15.93	12.20

Table 5.5 The performance of LID systems' mismatched conditions with and without vowel-region detection for noise-based front-end scheme.

	Without VOP		With VOP	
	DNN	DNN-WA	DNN	DNN-WA
Clean	13.02	10.02		
White noise				
20 dB	37.46	31.12	25.71	20.36
15 dB	40.25	34.15	33.08	27.03
10 dB	42.64	37.96	39.03	33.63
5 dB	44.23	41.22	42.70	37.63
Vehicle noise				
20 dB	30.57	22.19	27.26	19.14
15 dB	33.26	24.56	30.19	21.49
10 dB	37.24	28.46	33.75	25.00
5 dB	41.43	32.11	38.14	28.14

Fourier transform is computed over the speech signal having a window size of 20 ms with 10 ms overlap.

- The mean smoothed spectral energy contour is computed by using a 50-ms window.
- Further, a first-order difference operator (FOGD) is employed to enhance the smoothed spectral energy contour.
- To identify the significant changes in the spectral energy contour, FOGD is convolved with the spectral energy contour.
- The peaks and valley locations in FOGD response are related to the VOPs and VEPs. The region between VOP and VEP locations is considered as a vowel region.
- The false vowel regions are separated based on the property of vowel regions as they maintain the uniformity in fundamental frequency contour.
- Further, the estimate of vowel regions is enhanced by using the property of SoE contour as it shows positive and negative trends at VOP and VEP locations, respectively.

The vowel-region detection mechanism from a continuous speech utterance is illustrated in Fig. 5.2. Fig. 5.2a exhibits a speech signal along with the ground truth corresponding to the vowel region. The spectral energy contour computed using formants around GCIs is illustrated in Fig. 5.2b. The positive and negative peaks of the FOGD output and the hypothesized VOPs and VEPs are depicted in Fig. 5.2c. Fig. 5.2d–h is associated with the prediction in the first level (false vowel region marked red in color), fundamental frequency contour, prediction of vowel regions without false alarms, the SoE contour of the speech, and the vowel regions obtained after postprocessing of VOPs and VEPs.

Table 5.6 EER of LID systems when operated in noisy environments using front-end enhancement methods.

	Enhanced with SS		Enhanced with MMSE	
	DNN	DNN-WA	DNN	DNN-WA
White noise				
20 dB	29.45	24.17	26.11	22.54
15 dB	32.19	27.12	31.19	26.77
10 dB	33.33	31.79	33.37	29.91
5 dB	37.68	33.54	34.55	31.38
Vehicle noise				
20 dB	26.22	20.34	24.98	19.44
15 dB	29.58	22.46	25.61	20.43
10 dB	33.41	25.64	28.00	23.81
5 dB	35.33	29.35	32.5	26.44

Table 5.7 Performance of LID systems operated in matched and mismatched SNR conditions on OLR database. The bold-emphasized values represents the best performance of LID systems.

Model	DNN					DNN-WA				
	Train SNR	Clean	20	15	10	5	Clean	20	15	10
White noise										
20 dB	39.42	18.20	23.57	31.74	41.33	17.11	13.92	16.32	23.75	32.01
15 dB	42.85	24.16	20.15	27.48	37.48	22.87	14.76	13.10	16.18	26.80
10 dB	43.83	32.89	27.85	22.85	32.75	30.26	19.21	15.78	13.79	19.51
5 dB	42.79	37.43	36.02	32.39	27.05	30.69	23.54	20.20	16.80	15.14
Vehicle noise										
20 dB	30.14	14.13	14.13	18.09	24.67	17.89	12.40	14.49	17.70	24.09
15 dB	32.91	15.38	13.95	14.83	21.44	23.19	15.80	13.33	11.95	14.91
10 dB	35.24	17.26	15.57	13.57	16.18	24.96	17.62	14.45	11.83	12.16
5 dB	37.72	22.46	20.53	17.96	15.77	26.57	19.90	16.52	13.14	14.48

A mismatch between testing and training in the presence of a noisy environment being observed for LID systems leads to performance degradation. In this regard, to improve the performance of the LID systems, features are extracted within the vowel regions of the noisy test utterance. The performances of LID systems mismatched conditions with and without vowel region detection under different noise conditions are observed in [Table 5.4](#). From [Table 5.5](#), it can be observed that there is an improvement in LID system performance in noisy conditions by inclusion of vowel region detection as front-end to the LID systems. The improvement in LID systems in noisy conditions can be attributed to the high SNR property of vowel regions.

5.6.2 Improving the performance of LID systems by enhancing the speech signals

In the presence of a noisy environment, a mismatch between testing and training data used for developing LID is observed. That leads to degradation in performance. In this regard, the noisy test utterances to be enhanced to improve the performance of the LID systems. In this work, two different speech enhancement methods—the spectral subtraction method [42] and MMSE [43]—are considered. In this study, the LID systems are trained over clean speech. The noisy speech signals are enhanced using spectral subtraction and MMSE-based noise enhancement schemes, and the enhanced speech signals used obtain the language identity. The significance of efficient noise enhancement in terms of LID performance is demonstrated in [Table 5.6](#).

[Table 5.6](#) demonstrates the performance of the LID systems using front-end enhancement methods. In this connection, improvements in the performance of LID systems using speech enhancement techniques are shown at different SNR levels. At low SNR levels, the efficacy of enhancement methods is poor, which leads to a poor performance of the LID system. From [Table 5.6](#), the influence of noise on the LID system and the importance of front-end speech enhancement methods in improving the performance of the LID system are observed. Inconclusively, though the significant improvement is obtained with the use of front-end enhancement methods, it is far inferior to the performance of the LID system in clean environments.

Table 5.8 Sequence of steps involved in various CL strategies.

CLTS	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5
CL-full SNR	Clean	20 dB	15 dB	10 dB	5 dB
CL-high SNR	20 dB	15 dB	10 dB	5 dB	—
CL-low SNR	5 dB	10 dB	15 dB	20 dB	Clean

Table 5.9 EER of LID systems when operated in matched and mismatched SNR levels using OLR database. The bold-emphasized values represents the best performance of LID systems.

Model	DNN					DNN-WA				
	Clean	20	15	10	5	Clean	20	15	10	5
White noise										
Multi-SNR	16.78	20.56	26.68	29.63	33.57	12.25	17.87	22.52	25.54	32.33
CL_Full_SNR	41.46	37.29	34.41	30.25	27.64	36.44	32.45	28.72	25.72	22.89
CL_High_SNR	40.70	36.66	31.89	29.79	26.52	35.56	32.11	27.81	24.46	21.22
CL_Low_SNR	14.47	17.24	20.55	22.29	26.38	10.81	14.44	16.77	18.34	21.29
Vehicle noise										
Multi-SNR	14.23	17.25	20.97	24.33	30.24	10.56	14.24	16.54	20.41	25.56
CL_Full_SNR	37.33	32.25	28.67	26.60	22.72	31.15	27.12	23.02	20.81	17.82
CL_High_SNR	36.54	31.75	27.33	25.13	21.31	30.91	26.97	22.06	19.95	16.32
CL_Low_SNR	12.58	14.74	16.32	18.71	21.66	10.56	11.51	12.36	14.44	16

5.6.3 Improving the performance of LID systems by using CL strategies

In the matched conditions, the LID system is trained and tested with speech utterances at the same SNR level, whereas in mismatched conditions, the LID systems are trained and operated at different SNR levels. In this work, to minimize the mismatch between training and operating conditions, LID is trained with both clean and noisy data. Multiple LID systems are realized using datasets with synthetically generated noisy speech at different SNR levels, and their performances are described in [Table 5.7](#). The LID systems presented in this study are trained at a specific SNR and tested with matched and mismatched data at different SNR levels. In [Table 5.7](#), row 1 corresponds to the model employed for developing an LID system and column 1 corresponds to the SNR level of the training data. Here row 2 corresponds to SNR level of the testing data. Rows 4–7 and 9–12 are the performances of the language identification system for white and vehicle noise, respectively. In [Table 5.7](#), the performances of LID systems obtained in the matched situation are highlighted. From [Table 5.7](#), it is clear that the performance of multi-SNR LID systems is better than the LID systems trained with front-end enhancement schemes. In addition, it can be observed that the inclusion of noise data with training data decreased the mismatch among training and operating conditions and drive to achieve better performance. Further, superior performance is witnessed when there is an association between the training and operating environments. From [Table 5.7](#), a drop in the performance is observed when there is a substantial mismatch in SNR levels of training and testing data. For example, the performance of the LID system drastically diminished when the model trained with 20 dB SNR and tested at SNR of 5 dB. In real-life conditions, it is more prominent that there is a mismatch between training and operating conditions. Therefore, it is desirable to have an LID that is more robust to the mismatch between the training and testing environments.

CL approaches are task-specific, and the trained model can better generalize with an apt-CL strategy. In the present study, to develop robust LID systems, three distinct CL approaches are investigated: CL-low SNR, CL-high SNR, and CL-full SNR. In these CL approaches, the expanded dataset formed by adjusting the corpus in a sequence of SNR levels is presented in [Table 5.8](#). The results obtained by different CL approaches are presented in [Table 5.9](#).

The performances of different LID systems evolved with various CL approaches are set out in [Table 5.9](#). Column 1 is the LID system trained with learning procedures outlined in [Table 5.8](#). Columns 2–6, 7–11, and 12–16 describe the performances of LID systems developed using i-vector, DNN, and DNN-WA models, respectively. Rows 4–7 and 9–12 represent the performances of LID systems evolved by inducing white and vehicle noises to the clean dataset. From the experimental results, it is observed that the performance of the CL-based models is comparable to the performance of the LID with matched training. In some experiments, the performance is

even slightly higher than matched conditions, and this can be attributed to a large dataset augmented five to six times larger than the original dataset. Moreover, it can also be observed that the LID systems trained with the CL approaches like full-SNR and high-SNR have performed inadequately compared to the multi-SNR training approach, as they did not generalize well across different SNR levels.

From [Table 5.8](#), it can be observed that the performance progresses from low SNR levels to high SNR levels. The models that perform better while testing at low SNR levels do not perform adequately for high SNR levels. From [Table 5.9](#), it can be perceived that the performance of the LID system trained with CL-low SNR is better than the multi-SNR models. The LID systems emerged using the CL-low learning strategy generalized over the test utterances at different SNR levels. The reduction in the performance of the LID systems due to the mismatch between training and testing conditions is significantly minimized. From [Table 5.9](#), the CL-low SNR-based LID system performed better than any other CL procedures. Moreover, the CL-low SNR-based model showed better generalization across SNR levels. A similar trend can be observed both DNN and DNN-WA-based LID systems.

5.7 Conclusion and future scope

In this work, the importance of language identification (LID) in the sophistication of smart cities was addressed. Further, the influence of noisy and mobile environments on the performance of the LID system was studied. In addition, the chapter addressed the methods to improve the performance of the LID system in noisy and mobile conditions. The approaches used in this study to improve the performance of LID system were speech enhancement methods, vowel region-based feature extraction scheme, and CL-based training strategies, and were discussed. The NOISEX database was used to simulate the LID system performance in noisy environments, and the adaptive multirate codecs were used to simulate the performance of LID systems in mobile environments. A significant improvement in performance was observed by employing speech enhancement and vowel region-based front-end methods. However, this is limited to the low SNR levels, whereas the multirate SNR-based CL approach showed a significant improvement in the performance of LID systems in noisy environments. Moreover, this approach showed a generalization over different SNR levels.

This work addressed the implicit model for LID. The performance of explicit LID systems in noisy and mobile environments has to be addressed. In the literature, sequence summarization networks have shown their significance in capturing long-term temporal patterns. The efficacy of sequence summarization networks in language identification has to be addressed in clean and noisy environments.

Acknowledgment

The authors would like to thank the Science and Engineering Research Board (SERB) for funding the “Language Identification in Practical Environments” project (YSS/2014/000933).

References

- [1] E. Ambikairajah, H. Li, L. Wang, B. Yin, V. Sethu, Language identification: a tutorial, *IEEE Circuits Syst. Mag.* 11 (2) (2011) 82–108. <https://doi.org/10.1109/MCAS.2011.941081>.
- [2] M.A. Zissman, K.M. Berkling, Automatic language identification, *Speech Commun.* 35 (1) (2001) 115–124.
- [3] M.A. Zissman, Comparison of four approaches to automatic language identification of telephone speech, *IEEE Trans. Speech Audio Process.* 4 (1) (1996) 31.
- [4] H. Li, B. Ma, C.-H. Lee, A vector space modeling approach to spoken language identification, *IEEE Trans. Audio Speech Lang. Process.* 15 (1) (2007) 271–284.
- [5] Y.K. Muthusamy, E. Barnard, R.A. Cole, Reviewing automatic language identification, *IEEE Signal Process. Mag.* 11 (4) (1994) 33–41.
- [6] B. Ma, C. Guan, H. Li, C.-H. Lee, Multilingual speech recognition with language identification, in: Proceedings of INTERSPEECH, 2002.
- [7] T.F. Quatieri, E. Singer, R.B. Dunn, D.A. Reynolds, J.P. Campbell, Speaker and language recognition using speech codec parameters, Massachusetts Institute of Technology, Lincoln Laboratory, Lexington, 1999. Technical Report.
- [8] S. Ranjan, A. Misra, J.H.L. Hansen, Curriculum learning based probabilistic linear discriminant analysis for noise robust speaker recognition, in: Proceedings of INTERSPEECH, 2017, pp. 3717–3721.
- [9] S. Ranjan, J.H.L. Hansen, Curriculum learning based approaches for noise robust speaker recognition, *IEEE/ACM Trans. Audio Speech Lang. Process.* 26 (2017) 197–210.
- [10] Y. Bengio, J. Louradour, R. Collobert, J. Weston, Curriculum learning, in: Proceedings of the 26th Annual International Conference on Machine Learning, ACM, 2009, pp. 41–48.
- [11] S. Braun, D. Neil, S.-C. Liu, A curriculum learning method for improved noise robustness in automatic speech recognition, in: Proceedings of 25th European Signal Processing Conference (EUSIPCO), IEEE, 2017, pp. 548–552.
- [12] M.A. Zissman, E. Singer, Automatic language identification of telephone speech messages using phoneme recognition and n-gram modeling, in: Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, IEEE, 1994, pp. 305–308.
- [13] S. Brij Mohan Lal, V. Hari Krishna, A.K. Vuppala, M. Shrivastava, Significance of neural phonotactic models for large-scale spoken language identification, in: 2017 International Joint Conference on Neural Networks (IJCNN), IEEE, 2017.
- [14] P.A. Torres-Carrasquillo, D.A. Reynolds, J.R. Deller, Language identification using Gaussian mixture model tokenization, in: Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 1, IEEE, 2002, p. I–757. vol.
- [15] E. Wong, S. Sridharan, Methods to improve Gaussian mixture model based language identification system, in: Proceedings of the Seventh International Conference on Spoken Language Processing, 2002.

- [16] D. Nandi, A.K. Dutta, K.S. Rao, Significance of CV transition and steady vowel regions for language identification, in: 2014 Seventh International Conference on Contemporary Computing (IC3), IEEE, 2014, pp. 513–517.
- [17] V. Ravi Kumar, V. Hari Krishna, J.V. Bhupathiraju, V.G. Suryakanth, A.K. Vuppala, Improved language identification in presence of speech coding, in: Proceedings of International Conference on Mining Intelligence and Knowledge Exploration, India, Springer, 2015, pp. 312–322.
- [18] N. Dehak, P.A. Torres-Carrasquillo, D. Reynolds, R. Dehak, Language recognition via i-vectors and dimensionality reduction, in: Proceedings of the Twelfth Annual Conference of the International Speech Communication Association, 2011.
- [19] F. Richardson, D. Reynolds, N. Dehak, A unified deep neural network for speaker and language recognition, in: Proceedings of INTERSPEECH, 2015, pp. 1146–1150.
- [20] M.K. Rai, M.S. Fahad, J. Yadav, K.S. Rao, Language identification using PLDA based on i-vector in noisy environment, in: 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE, 2016, pp. 1014–1020.
- [21] V. Ravi Kumar, K. Gurugubelli, P. Jain, H.K. Vydana, A.K. Vuppala, IIITH-ILSC speech database for Indian language identification, in: Proceedings of the Sixth International Workshop on Spoken Language Technologies for Under-Resourced Languages, pp. 56–60.
- [22] I. Lopez-Moreno, J. Gonzalez-Dominguez, O. Plchot, Automatic language identification using deep neural networks, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014.
- [23] R. Fé, P. Matějka, F. Grézl, O. Plchot, J. Černockí, Multilingual bottleneck features for language recognition, in: Proceedings of the Sixteenth Annual Conference of the International Speech Communication Association, 2015.
- [24] P. Matejka, L. Zhang, T. Ng, H.S. Mallidi, O. Glembek, J. Ma, B. Zhang, Neural network bottleneck features for language identification, in: Proceedings of the IEEE Odyssey, 2014, pp. 299–304.
- [25] Y. Lei, L. Ferrer, A. Lawson, M. McLaren, N. Scheffer, Application of convolutional neural networks to language identification in noisy conditions, in: Proceedings of the Odyssey-14, Joensuu, Finland, 2014.
- [26] W. Geng, W. Wang, Y. Zhao, X. Cai, B. Xu, End-to-end language identification using attention-based recurrent neural networks, in: Proceedings of INTERSPEECH, 2016, pp. 2944–2948.
- [27] K.V. Mounika, A. Sivanand, H.R. Lakshmi, S.V. Gangashetty, A.K. Vuppala, An investigation of deep neural network architectures for language recognition in Indian languages, in: Proceedings of INTERSPEECH, 2016, pp. 2930–2933.
- [28] C. Raffel, D.P.W. Ellis, Feed-forward networks with attention can solve some long-term memory problems, arXiv preprint arXiv:1512.08756 (2015).
- [29] R. Thirumuru, R. Vuddagiri, K. Gurugubelli, A.K. Vuppala, Significance of accuracy in vowel region detection for robust language identification, in: 2018 Fifth International Conference on Signal Processing and Integrated Networks (SPIN), IEEE, 2018, pp. 826–830.
- [30] V. Ravi Kumar, H.K. Vydana, A.K. Vuppala, Curriculum learning based approach for noise robust language identification using DNN with attention, Expert Syst. Appl. (2018), <https://doi.org/10.1016/j.eswa.2018.06.004>.

- [31] D. Wang, L. Li, D. Tang, Q. Chen, AP16-OL7: a multilingual database for oriental languages and a language recognition baseline, in: 2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), IEEE, 2016, pp. 1–5.
- [32] Z. Tang, D. Wang, Y. Chen, Q. Chen, AP17-OLR challenge: data, plan, and baseline, in: Proceedings of Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, IEEE, 2017, pp. 749–753.
- [33] V. Ravi Kumar, H.K. Vydana, A.K. Vuppala, Improved language identification using stacked SDC features and residual neural network, in: Proceedings of the Sixth International Workshop on Spoken Language Technologies for Under-Resourced Languages, pp. 205–209.
- [34] S. Ganapathy, K. Han, S. Thomas, M. Omar, M.V. Segbroeck, S.S. Narayanan, Robust language identification using convolutional neural network features, in: Proceedings of the Fifteenth Annual Conference of the International Speech Communication Association, 2014, pp. 1846–1850.
- [35] J. Gonzalez-Dominguez, I. Lopez-Moreno, P.J. Moreno, J. Gonzalez-Rodriguez, Frame-by-frame language identification in short utterances using deep neural networks, *Neural Netw.* 64 (2015) 49–58.
- [36] J. Gonzalez-Dominguez, I. Lopez-Moreno, H. Sak, J. Gonzalez-Rodriguez, P.J. Moreno, Automatic language identification using long short-term memory recurrent neural networks, in: Proceedings of INTERSPEECH, 2014, pp. 2155–2159.
- [37] C. Raffel, D.P.W. Ellis, Feed-forward networks with attention can solve some long-term memory problems, *CoRR abs/1512.08756* (2015). <http://arxiv.org/abs/1512.08756>.
- [38] S.R.M. Prasanna, B.V.S. Reddy, P. Krishnamoorthy, Vowel onset point detection using source, spectral peaks, and modulation spectrum energies, *IEEE Trans. Audio Speech Lang. Process.* 17 (4) (2009) 556–565.
- [39] A.K. Vuppala, J. Yadav, S. Chakrabarti, K.S. Rao, Vowel onset point detection for low bit rate coded speech, *IEEE Trans. Audio Speech Lang. Process.* 20 (6) (2012) 1894–1903.
- [40] A.K. Vuppala, K.S. Rao, Vowel onset point detection for noisy speech using spectral energy at formant frequencies, *Int. J. Speech Technol.* 16 (2) (2013) 229–235.
- [41] R. Thirumuru, S.V. Gangashetty, A.K. Vuppala, Improved vowel region detection from a continuous speech using post processing of vowel onset points and vowel end-points, *Multimed. Tools Appl.* 77 (4) (2018) 4753–4767.
- [42] S. Boll, Suppression of acoustic noise in speech using spectral subtraction, *IEEE Trans. Acoust. Speech Signal Process.* 27 (2) (1979) 113–120.
- [43] R. Martin, Statistical methods for the enhancement of noisy speech, in: *Speech Enhancement*, Springer, New York, NY, 2005, pp. 43–65.

This page intentionally left blank

Effective natural interaction with our sensorized smart homes[☆]

6

António Teixeira, Nuno Almeida, Maksym Ketsmur, Samuel Silva

*Department of Electronics Telecommunications and Informatics/IEETA, University of Aveiro,
Aveiro, Portugal*

6.1 Introduction

The IoT (Internet of Things) and ubiquitous computing trends affect all human environments, including the buildings we work in as well as our homes, which are becoming increasingly smarter via their connection to a myriad of sensors, devices, and smart appliances. The literature on smart homes identifies numerous advantages for its occupants including enhancements to comfort, energy efficiency, improvements in health and assisted living, and security [1, 2]. Nevertheless, while smart homes can potentially offer users help with the automation of tasks and routine activities, those automations should favor the functionalities that are important to the user without disregarding their ability to influence it. Experts believe that “computers should not make choices for users, but the other way around” [3], fostering a sense of control and an improved adaptation to the user’s needs and expectations. The key to “control” is proper interaction with the smart home ecosystem.

The household occupant has also an increasing access to numerous interactive devices, such as tablets, smartphones, and smartwatches. Many of these devices support technologies that enable richer and more natural ways of interaction, such as speech and gestures, a diverse form of multimodality that needs to be explored properly so that it can promote increased usability and acceptance. Humans communicate with each other using their full range of senses to transmit and receive information, for instance, as in a conversation, articulating gestures with speech and facial expressions. The communication with machines (and the smart home) should likewise be natural and encompass such diversity of options. In this context, harnessing the full potential of these different technologies for interaction poses numerous challenges.

Multimodal Interaction (MMI), i.e., interaction considering a wide set of possibilities such as touch, speech, and gestures, has become a possibility given the currently available technology [4], but the development of truly MMI applications is still a challenging task. In this regard, the W3C proposes a new standard for multimodal architectures [5, 6] defining standard markup for communication and

[☆] Supported by Smart Green Home project (<http://www.ua.pt/smартgreenhomes/>).

specifying a set of internal components of the architecture. Despite having some open points, such as discovery of new modalities in the system or its focus on web scenarios, it seems a good starting point to create multimodal applications as it serves the principles of a decoupled architecture, enabling easy integration of modalities, and is open enough to encompass new evolutions.

Nevertheless, while multimodal interaction serves increased adaptability, additional layers are required, on top of this versatility, to increase naturalness and, most importantly, provide the user with an interaction with the home as a whole, and not as a set of isolated interactive artifacts or devices. To this end [7], many people feel that conversational assistants and related approaches have the potential to become an important part of many households providing various forms of interaction like text and voice. Accordingly, home assistants, and particularly those supporting conversational features, have exploded in popularity over the past 3 years and are now commonly used in millions of homes to perform small chores such as setting up a meeting or checking the weather, as well as more complex tasks involving booking a flight or calling a taxi. Notable examples of conversational assistants are Google Assistant, Amazon Alexa, Siri, and Cortana [8–13].

While smart home assistants facilitate the performance of a wide variety of tasks, there remain lacunae that must be recognized and addressed. For example, it is still not possible to get certain critical information about the smart home such as its resource consumption and costs, and widespread languages such as Portuguese are yet to be supported.

Overall, the increase in sensors, devices, and appliances in our homes has transformed the home into a rather complex environment with which to interact. This characteristic cannot be merely addressed by a matching set of device-dependent applications, turning the smart home into a set of isolated interactive artifacts. Hence, there is a strong need to unify this experience, blending this diversity into a unique interactive ecosystem. This can be tackled to a large extent by the proposal of a unique, integrated, ubiquitous distributed smart home application capable of handling a dynamic set of sensors and devices and providing the different house occupants (e.g., children, teenagers, young adults, and elderly) with natural and simple ways of controlling and accessing information. However, the creation of such application presents a challenge, particularly due to the need to support natural and adaptive forms of interaction beyond a simple home dashboard application.

In this chapter, we propose a conceptual vision to tackle interaction with the smart home, inspired by human-human communication. In so doing, we describe our ongoing work in next generation smart homes within the context of our Smart Green Homes project—a partnership between the University of Aveiro and Bosch Thermotechnology. Some important aspects of our work consist of the deployment of a multimodal interaction architecture and a prototype of a home assistant, supporting the interaction with the household ecosystem and endowed with conversational features in European Portuguese.

The remainder of this chapter starts by looking into the literature on related work involving smart environments and the challenges of tackling interaction with them. Next, [Sections 6.3 and 6.4](#) present our conceptual vision for addressing human-home

interaction and discussing the overall infrastructure to support it, respectively. Section 6.5 details our current work, following the presented vision, and covering the design and development of a multimodal interactive home assistant. Several illustrative application examples are presented in Section 6.6. Finally, Section 6.7 presents some conclusions and discusses future directions.

6.2 Related work

Taking in consideration the complexity resulting from the recent evolutions in homes—particularly the increase in sensing and connected devices—several domains need to be considered in order to foster natural human-home interaction. Considering these challenges as well as our proposals to meet them head on, the related work deemed relevant is in fact varied, covering a broad range of topics: sensors and devices targeting living spaces, smart appliances, ubiquitous computing, interaction with multiple devices, human interaction with buildings, and smart homes.

6.2.1 Home sensors and actuators

Sensors and actuators are essential devices to build a smart home. While sensors capture some aspect of the physical world, converting it to digital data, actuators do the opposite, converting instructions to some mechanical action [14].

There is a wide range of sensor types available, each measuring different factors of the environment, such as motion, air quality, amount of light, temperature, humidity, door state, orientation, and location [14]. The structure of the retrieved data varies depending on the types of sensors and it ranges from simple Boolean values (e.g., if a door is open or closed), to values of temperature or humidity, to even more complex data, e.g., for locations.

Smart lights, door lockers, and smart valves to cut water or gas are examples of actuators that can be installed and controlled in the smart home context.

Given the diversity of devices and the places where they are installed, it is important to consider how these devices communicate with the system. Most devices use wireless sensor networks and the technologies and protocols used depend on the energy consumption, range, and required network bandwidth. Among the most used protocols are ZigBee, BLE (Bluetooth low energy), and Wi-Fi [15]; ZigBee devices work in mesh facilitating the installation of some devices in farther locations having, however, a lower bandwidth [16].

6.2.2 High level handling of sensor networks

Among the plethora of sensing devices from many manufacturers, it is a common approach that each manufacturer has, for each device (or a subset of its catalog), a custom application to communicate with it. This makes it especially hard to integrate new devices into new systems. In this context, there are two approaches worth noting: the Semantic Sensor Network (SSN) [17] and the IPSO Smart Object Guidelines [18].

The Semantic Sensor Network (SSN) ontology [17] is recommended by the W3C, providing the basis to describe and store everything about a sensor, including, among others, the sensor identification information, observations/results, and procedures. It aims to “provide flexible but coherent perspectives for representing the entities, relations, and activities involved in sensing, sampling, and actuation” [17].

The IPSO Smart Object Guidelines [18] provide a common design pattern, and the recommendation aims to enable the interoperability between smart devices and applications or services, using a markup language to describe a sensor’s parameters. These guidelines can be considered along with a communication protocol, such as the Constrained Application Protocol (CoAP), providing the methods to request and respond serving resource identification and discovery mechanisms.

6.2.3 “Smart” appliances

The number of smart appliances has increased rapidly, in the past few years, with more and more manufacturers upgrading their product range to bring innovative features of “artificial intelligence” to their products. In addition, while they deliver appliances that can adapt to the users’ needs and are easier to use, such AI-driven appliances potentially equip the user with more control over the devices. That is because these smart devices are connected to the network, enabling their remote control, and accessing information about their state of operation with some control functions and information that can be provided by these smart appliances, opening a new set of possibilities when compared with conventional appliances.

The range of smart appliances commercially available includes refrigerators, washing machines, vacuum cleaners, etc. Some notable examples from major brands are:

- a refrigerator [19] that notifies the user if the door stays accidentally open, knows what is inside the refrigerator and expiration dates can be set, and allows information to be viewed on a smartphone;
- a robotic vacuum cleaner [20] that can be controlled with a smartphone or a virtual assistant such as Amazon Alexa, and can map the house and cleans multiple rooms avoiding obstacles; and
- a washing machine [21] with Wi-Fi support, which allows the user to monitor the remaining time to finish, gives automatic recommendations, and monitors the washing machine’s condition to predict maintenance.

6.2.4 Ubiquitous computing, interaction, and multidevice contexts

Technological growth has boosted the appearance of different types of devices, with different form factors, such as smartphones, wearables, smart sensors, and other embedded invisible computers, which enabled the era of ubiquitous computing [22]. Ubiquitous computing aims to provide access to a system at all times, in any place. This is a feature that should also be provided for interaction with smart

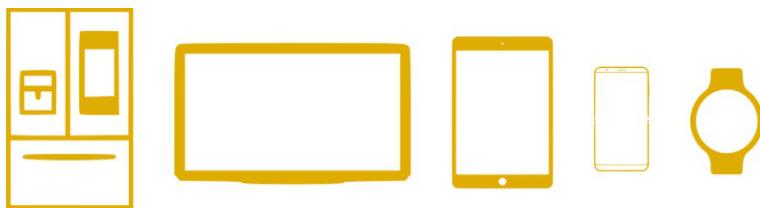


FIG. 6.1

Representation of the diversity of devices, screen sizes, and formats. From left to right: smart refrigerator with display, smart TV, tablet, smartphone, and smartwatch.

home technologies, making the house accessible in any place while inside the house as well as outside, and at any time [23].

A wide range of research can be found in the literature, exploring ubiquitous computing in the context of smart homes with notable recent examples being a system to access home appliances remotely using a mobile device [24] and a ubiquitous home controller system for the smart home [25].

From all the devices used in ubiquitous computing systems, each has different characteristics, different networking interfaces, and can have input or output capabilities. Some of these devices may be aimed for user interaction [22], with one or more devices used to interact with the whole system. Among these interactive devices, there is a strong heterogeneity, each having different capabilities, display sizes, and formats, such as the ones illustrated in Fig. 6.1. The displays where information is presented to us can be a part of the devices we carry around, but can also be in devices with fixed places, around the house, such as smart TVs or displays installed in smart appliances. This variety poses several challenges when it comes to providing the user with a transparent and seamless user interaction experience. Some approaches have been taken to try solving some of those challenges, and the literature refers to two notable lines of research: (1) ubiquitous multidevice interactions; and (2) migratory multimodal interfaces.

For the first topic, several architectures and frameworks are described, notable examples being HIPerFace [26], which has an architecture allowing rapid prototyping and experimentation with different devices, supporting multiple devices at once, and the Interactive Workspaces project [27] architecture, which allows interaction through multiple devices in collaborative scenarios.

Regarding migratory multimodal interfaces, Berti and Paternò [28] describe work allowing users to move from one device to another while seamlessly continuing the task they were performing on the first device; and Blumendorf et al. [29] present a multimodal system enabling the adaptation of the user interface to fit the different devices.

6.2.5 Human building interaction

With the proliferation of technology supporting novel and ubiquitous interaction paradigms, interaction must be designed to blend with the surrounding environment as a means to foster a more natural usage, with interactive systems as an integral part of

the environment instead of as artifacts (e.g., mobile devices) that serve particular purposes despite their surroundings. In this regard, beyond human-computer interaction, a large role can also be played by buildings and urban architecture. The physical and spatial dimensions can be conceptualized for a better integration of interactive technologies, delivering the conditions for new interaction possibilities that blend more effectively with the environment. Additionally, the social perspective of a changing society, where the paradigms of work, mobility, and environmental awareness have dramatically been altered [30], must be brought into the equation as a shaper of user expectations [31] and behavior [32].

Human building interaction (HBI) aims to gather all these different perspectives. Potential scenarios fostering initial exploration of these matters, although not embodying its full richness and complexity [33], include smart homes [34–36] or offices, which are a possible setting for raising discussion, establishing concrete challenges and putting to test this desirable symbiosis between users and the built environment supported on a multidisciplinary, user-centered perspective [37]. Additionally, the field of HBI is even more challenging because its complexity and broad range of disciplines involved demand that a multiplicity of perspectives can be articulated. This will require that HCI places further emphasis on the building (an increasingly customizable space), considering it as an entity with which the user sets in a dialogue (a communication in both directions [38] with a certain level of mutual understanding [39]) with a varying range of goals.

Hence, HBI stands to gather a multidisciplinary perspective to increase understanding of human-building dynamics toward shaping improved awareness of aspects such as energy efficiency balanced by user expectations and needs. This poses numerous challenges for HCI, and how it can provide the methods and tools to foster this interdisciplinary effort toward a more natural interaction with smart environments: (a) support easy instantiation of novel (complex) interactive ecosystems; (b) integrate user devices into existing ecosystems (e.g., the smart home); (c) bring the building into the user's context (e.g., daily routines or personal devices); and (d) adapt to different user profiles, contexts, and behaviors.

6.2.6 Interaction with smart homes

A smart home is “a residence equipped with a communications network, linking sensors, domestic appliances, and devices, that can be remotely monitored, accessed or controlled, and which provides services that respond to the needs of its inhabitants” [40].

Although research on the different aspects involved in smart homes is quite prolific, ironically users are not typically the focus [41], even though users need to interact [42], access information, and control the smart home. In fact, how users interact with their smart devices is one of the key aspects of a well-performing smart home, and therefore must not be ignored or underplayed. In this regard, Eggan et al. [43] identify three questions that encompass the major challenges that need to be addressed when designing smart homes:

- What makes a worthwhile user experience?
- How to design for user experience?
- How to design for user experiences that can be seamlessly integrated in everyday life?

While some authors argue that the number of interactions between users and the smart home must be kept to a minimum (e.g., Ref. [1]), where the smart home operates almost independent of the home owner, others propose, in contrast (e.g., Ozkan et al. [44]), that users should have total control of the smart home. The reasoning behind this is that when users do not have control of a home once it has been turned into a fully automated smart home, the whole concept of smart homes might be difficult for users to accept in the first place. If the goal is to remove obstacles in the adoption of smart home systems, preserving the autonomy of the user may seem like the most sensible course of action, at least to those arguing for a user-centric smart home that gives total control to the user over its functions and operations.

Usually, the interface with a smart home is provided through smartphones or similar devices, using touch applications or voice assistants [45]. Some more traditional applications can be found, such as Casalendar [46] where the interface of the application is based on a calendar and integrates the status of the smart home enabling its configuration. Another example is Control Home Easily [47], providing a friendly interface for mobile devices that allows the control of such devices using widgets generated based on the information available for the devices. There are other applications to intermediate between smart home devices and users, with most being proprietary solutions, but allowing the configuration of new devices. Among them, the most known and used are:

- IFTTT [48], which allows connecting applications and devices and the control of devices is enabled using triggers;
- Gideon [49], which enables integration of multiple devices and allows supervision of the smart home; in addition, the user can chat with Gideon;
- Wink [50], which integrates multiple devices and the user can control them using a touch interface;
- Philips Hue app [51], which is used to control the lights from their brands; and
- Google Assistant, Siri, Cortana, and Alexa, which are conversational assistants, enabling the control of devices using speech.

A summary of the capabilities of these applications is presented in Table 6.1. Among these applications, there are those that are more generic and can, therefore, integrate and use smart devices from multiple brands. However, the interactions with these more generic applications fall within two separate modes of interaction: they are either exclusively touch or speech, but certainly not a combination of both modalities. Despite that, applications, such as IFTTT, can react according to some context variables; however, it is limited to IF/THEN reactions.

Table 6.1 Small sample of IOT applications and their capabilities.

	Mobile	Works without mobile in other devices (browser)	Multiple brands	Voice interaction	Control	Visualize information
Philips Hue	✓					
Nest	✓				✓	✓
IFTTT	✓	✓	✓		✓	
Gideon	✓		✓	✓	✓	✓
Wink	✓		✓		✓	✓

6.2.7 Assistants

Conversational assistants are mostly intended to perform daily tasks more easily. In this way, communication with the assistants (machines) must be as natural as possible for the user. Humans have been taught how to interact with technology through click, type, and touch, whereas humans communicate with each other using their full range of senses to transmit and receive information, as in a conversation when using gestures with speech and facial expressions. The main goal of conversational assistants is to reduce the gap between the human and machine, and to map out how to program the computer to simulate human behavior.

Conversational interfaces allow users to interact with systems through text or spoken (natural) language [52]. Interfaces that allow speech input are often referred to as voice user interfaces (VUI), conversational agents, and virtual or intelligent personal assistants, and have exploded in popularity over the past 3 years, now being accessible for most people via their integration in smartphones (e.g., Siri, Google Assistant) and in stationary intelligent assistant devices (e.g., Google Home, Amazon Echo). These devices are commonly used in millions of homes to perform small chores, such as setting up a meeting or checking the weather, as well as more complex tasks such as booking a flight or calling a taxi [53].

Conversational interfaces supporting interaction only through text are called chatbots and are less intelligent than assistants, since they follow well-defined navigation charts and responses are given only to specific commands and do not go beyond what they were strictly programmed for [52]. The main goal of conversational assistants is to achieve a real conversation with the user. This can ultimately be achieved by a careful understanding of human-to-human conversation since it is the most comprehensive way for humans to encode, transmit, and reason about knowledge. Thus, conversational assistants aim not only to perform the small chores and control some devices, but also to allow the user to have a dialogue with the assistant, fostering the feel of “common ground” [54]. This feeling of a user having a nice rapport with the system is one of the most complicated goals to achieve for any

conversational assistant, since the machine must understand the user's intent, actions, personal preferences, and many other factors that humans use when joined in human-to-human dialogue.

In the face of these recent technological developments, conversational agents have become very appealing to the public as a means for humans to interact with machines because of their more natural feel. In fact, many people consider them to have the potential to become an important part of their households, providing various forms of interaction, such as text and voice. In this context, the community has been actively working in this field because it is driven by a public need to interact with conversational agents in an intuitive and natural manner. Table 6.2 lists a set of recently proposed conversational assistants and a brief summary of their main features, namely the supported types of interaction (chat, speech, and multimodal—gestures, facial expressions, and others), the target audience (elderly people, workers/employees, generic, nonvisuals, and scientists), and the main tools used for their development.

From the listed tools, it is possible to observe that there are more systems in which the interaction is carried out using speech, but not much more than the systems in which the interaction is done using chat. Although speech is one of the easiest forms of interaction for the human, there are cases where speech is not convenient (in an office setting), which explains the insignificant difference in the number of speech and chat systems. There are also multimodal systems that, in addition to using speech, take into account, for instance, gestures, facial expression, and touch. This not only helps to simplify the interaction, but also can potentially make the systems accessible/adaptable to a wider range of users.

As is shown in Table 6.2, most assistants are developed for generic users, like health assistants or personal assistants that help manage the user's calendar or set an alarm. The more specific systems are those proposed for elderly or nonvisual people, supporting speech and multimodal interaction. Normally, these conversational assistants are implemented in users' homes to help them perform daily tasks and enable lonely elderly people to have a conversation.

Usually, the conversational chat systems are developed using Slack's framework and speech systems use IBM Watson or similar tools, since these can also analyze natural language text and speech, identifying the user's intent and entities present in it. The last system listed, Valletto, was developed using the Spacy.io framework, that can do more (identify the dependency tree) than Slack and IBM Watson, which allows the system not only to identify the intent and entities, but also to identify a very important element, specifically, the focus of the phrase [69].

Having analyzed a wide set of assistants, we were able to form the following conclusions: (1) interaction with all the assistants is only possible in English; (2) only a few assistants allow user initiative; (3) a large number of assistants have predefined questions and responses (in the case of medical assistants and personal assistants); and (4) there is a strong focus on the development of multimodal assistants since they cover speech, gestures, facial expressions, and touches that humans usually use in real human-to-human dialogue.

Table 6.2 Conversational assistants.

Name/reference	Interaction				Target audience					Tools
	Chat	Speech	Multimodal	Elderly	Workers	Generic	Nonvisual	Scientists		
Robota [13]	✓	✓			✓				Slack	
Ella Jones [55]	✓				✓				-	
Evorus [56]	✓				✓				-	
Kephart et al. [57]		✓	✓					✓	IBM Watson	
Bieliauskas et al. [58]	✓					✓	✓		Slack	
Shim [59]	✓					✓	✓		Phone App	
Albadawi et al. [60]		✓				✓	✓		-	
Pasupalak et al. [61]		✓				✓	✓		Phone App	
Billie [62]		✓		✓					-	
Brown et al. [63]		✓				✓			-	
Foodie [64]	✓	✓				✓			-	
Iris [65]	✓	✓				✓			IBM Watson	
Koseki et al. [66]		✓		✓					-	
Bhalerao et al. [67]		✓					✓		Google	
Kassel and Rohs [68]		✓	✓	✓		✓			Spacy.io	

6.2.8 Design challenges in IoT and data contexts

The subject of designing for smart environments and the IoT opens a range of new challenges that need careful consideration. The wide range of sensors and data available are instruments in a much wider scope and, while they may shape the way we look into problems, nowadays we must also be able to see beyond them and concentrate in who will take advantage of their existence.

When most people hear about the “Internet of Things,” they think about a range of sensors, appliances, and other devices that are connected and share the information via the Internet. The dominant idea that is circulated is of an ideal world where the data collection is automated and centralized, and where things will be known and will happen accordingly. However, the real value of IoT resides not only in this physical and data layer, but also, to a strong extent, in how users will take a part in it. The IoT says nothing about the people who are also a fundamental part of the network [70].

The ways we can interact with a product are what makes us appreciate it and its ability to serve our needs naturally is what makes that interaction memorable. People will not remember products because of the data collection that they perform, but because of how these products will enhance and simplify our day-to-day life and our communication with each other.

6.2.8.1 Design for IoT

Designing connected products is impossible without addressing both interoperability and interusability. That is, without the first, we face a chaotic future of incompatible devices and standards proposed by each competing brand. In addition, without the second, core aspects of how users relate with such devices (e.g., connect, play, stop) will need to be relearned repeatedly with each new interface [14]. Naturally, the users will be dissuaded from using wireless devices when confronted with such varied and diverse ways of interacting with such devices.

The “Internet of Things” refers to the constant growing of everyday objects acquiring connectivity, sensing abilities, and increased computing power. Since our work is related to smart homes, we will talk about home technology such as thermostats, home appliances, lights, energy monitoring, air quality sensors, etc. Some of these devices have screen displays (e.g., heating controllers, and washing machines), some communicate with users via flashing LEDs or sounds, while others possess no output capabilities. The main goal is, despite the differences in physical dimensions, to make the users feel as if they are using a coherent service rather than a set of disjointed user interfaces—in essence, “interusability” [14, 70].

6.2.8.2 Design for data

Design for Data describes the data collected by the system and defines the relationships between different system components: e.g., which data and how they are handled within the chosen domain “data model.” Data models are useful to improve the system’s interoperability, organize controls, and make data meaningful in order to determine how to act in an autonomous way using domain knowledge about user

needs, activities, and context. Additionally, the data provide the system with a “mental model” of how things (IoT devices) work (and should work) together, even as it scales to larger complexities.

To use data models, in the context of smart homes and IoT devices, these may include at the very minimum system-generated metadata about devices, device context, domain model, and knowledge about the house occupants and their activities [14].

Information about the device type, its capabilities, and the data it gathers (device’s metadata) results in standard ways of information sharing, thus enabling devices to describe themselves and their data, so applications and users can find useful and relevant devices and know how to work with them. Nevertheless, having metadata about the devices ensures a basic aspect of their diversity, in the smart home. The device may also include metadata about device context, including name, location, its owner, or description of its purpose. Using that information makes it possible to present devices and their features in a smarter way (e.g., aggregated or by purpose) to the user.

A domain model is responsible for describing particular user activities and supports the description of the system’s logical components. Furthermore, a wider domain knowledge about the activities or context of use is necessary to make the system learn how to behave in different situations.

Finally, since the house inhabitants are the most important part of the system, knowledge about the user is paramount for system adaptation.

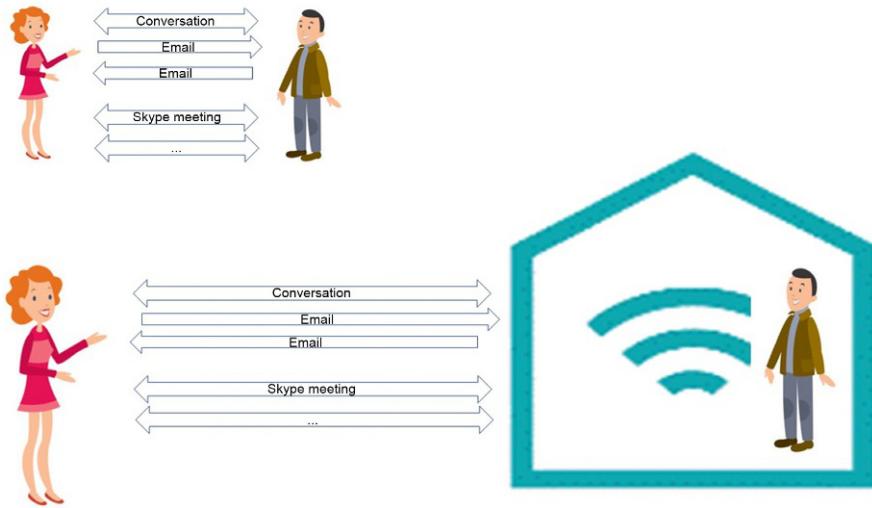
6.3 An integrated vision for human-home interaction

The vision we propose for the interaction of humans with (their) homes is directly inspired by human-human interaction, particularly using natural language (spoken or written).

Human-human interaction encompasses several distinctive characteristics, such as [71–73]: (1) bi-directionality, with initiative by any of the participants; (2) use of multiple turns, in many situations, being one participant in charge, in each turn; (3) acts; and (4) use of multiple senses.

These characteristics of direct human-human interactions, such as a conversation, are adopted in new forms of communication such as chats and video-conference meetings. In this regard, we argue that it can also be advantageous to adapt the essential characteristics of these human-human communications to interaction with inanimate entities such as homes. Technologies can potentially promote these inanimate entities to interlocutors, capable of answering to our orders or queries. Consequently, we see the communication of humans with their homes, in physical presence or remotely, as illustrated in Fig. 6.2.

When at home, as in face-to-face communication, humans can have a conversation with an assistant or use any of the devices available (e.g., tablets, smartphones,

**FIG. 6.2**

Authors' vision for human-home interaction.

or real or virtual controllers) to establish bidirectional communication. At a distance, interaction can be performed in ways that are analogous to the ones we use to communicate, for instance, with a family member, at home, from our workplace: by email, phone calls, or videoconference meetings, where the choice is made according to need and context. For example, alone in the car, we can call home, while, during a meeting a message sent through a chat service or an email are much more suitable alternatives to making a call.

For this vision to be a reality, our homes need to: (1) have multiple means to provide information to humans; (2) be capable of decoding and acting according to the information they receive; and (3) implement the several forms of exchanging information (interaction) commonly used by humans.

6.4 Integrated ubiquitous distributed solution for a smart home

The consideration of the smart home as an interlocutor for interaction with humans supporting a wide range of alternatives including conversational features, shaped as a multimodal assistant, requires the availability of several important resources, the most prominent being: (1) access to data and information regarding the house, to enable answering user queries; (2) capability to decode and act according to the received information; and (3) an infrastructure supporting multimodal interaction.

6.4.1 Smart home's information structuring

To allow application of Design for Data and to enable rich queries, a semantic-based approach was adopted for the information.

Our ontology (Fig. 6.3), based in [14, 74, 75], consists of five main classes: (1) *Partition*, describing home divisions, such as bedroom, kitchen, or living room; (2) *Device*, for home appliances, lights, and plugs present on each home division; (3) *Environment*, for environment measurements (e.g., humidity, temperature) of each home division; (4) *Resource*, that represents resources that are consumed by

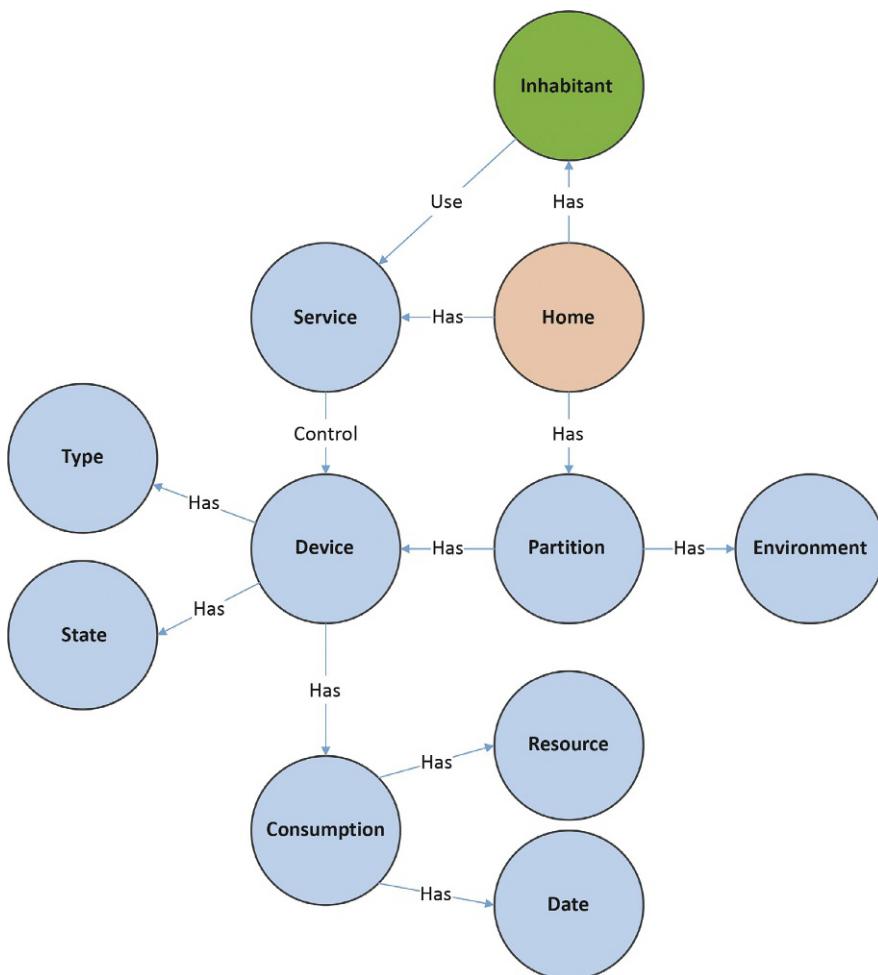


FIG. 6.3

Ontology used for home information.

each device, such as water, energy, or gas; and (5) *Inhabitant*, representing the home occupants who can interact with the home appliances, lights, and plugs through the available services.

6.4.2 Users and context

A simple user and context model have been integrated, mainly to address the needs of concrete applications such as the proof-of-concept scenarios presented later on.

The user information is available through a cloud service that, when requested, queries a database for the user and its model. The database is flexible and stores JSON-like documents (MongoDB), allowing new parameters to be added in the future.

Our first user model includes some information of the users (e.g., age and language) and preferences (e.g., preferred room temperature).

The context stores information of the house and its rooms, such as the available devices and interaction modalities, the users in each division, and lighting status. Fig. 6.4 illustrates an example of the structure and data for the living room.

```
1 {   "divId" : "livingroom",
2   "devices" : [
3     "minipc",
4     "speaker",
5     "projector",
6     "kinect",
7     "lights",
8     "heater" ],
9   "modalities" : [
10    "speech_in",
11    "speech_out",
12    "user_identification",
13    "GUI"],
14   "dynamicDevices" : [],
15   "dynamicModalities" : [],
16   "lightN" : 1,
17   "current_context" : {
18     "currentUsers" : [],
19     "numberOfUsers" : 0,
20     "lightState" : 0,
21     "heaterState" : 0,
22     "roomTemperature" : 20 } }
```

FIG. 6.4

Example in JSON of the structure and contents of the user context regarding the living room.

6.4.3 Home control

To control all the devices—such as smart lights and smart plugs—a new service was created to aggregate and communicate with all devices more transparently. This service has access to the context model where are stored the configuration of the devices and their locations. For instance, if the application needs to turn on the light, it can request this service to turn on the light in a specific division, and the service is responsible to find which light to turn on. Most requests are done with HTTP GET with the parameters of the request encoded in the URL.

The service supports functions as: (1) lights control (on/off/set brightness) and information regarding lights; (2) plugs control (on/off); (3) retrieve sensors information (such as temperature); and (4) retrieve consumption information (such as water or gas).

6.4.4 Multimodal interaction support

Support for multimodal interaction, in the smart home, was accomplished by adopting the multimodal framework proposed previously by the authors [76, 77]. This framework, compatible with the W3C recommendations for multimodal architectures, was the result of a long-standing effort in facilitating development of multimodal interaction with notable applications in domains such as Ambient Assisted Living [78]. To integrate the multimodal framework in the context of smart homes, some improvements were made to the framework, namely, the extension to multiplatform, enhancements to multidevice support and addition of adaptation mechanisms considering, at the moment, simple user and context models.

An Interaction Manager (IM), running in the home server, exchanges Life Cycle events [6] with the other modules and modalities using the HTTP protocol. Since browser technology (HTTP/JavaScript) is one simple way to develop and deploy applications for multiplatform, the IM can continue using the HTTP requests.

In addition, to target the smart home context, several modalities were developed or adapted from previous work for AAL [76]:

- **Speech**—The considered speech modality is based on a previously developed Generic Speech Modality included in the adopted framework [79].
 - **Speech input**—The target modality can be used within a browser; it was developed using HTML5 and JavaScript. The local browser component of the modality is capable of recording the audio from the device’s microphone. Using a voice activity detection algorithm (VAD), speech is detected and sent to a service to process the voice signal through the WebSockets protocol. The service receives the stream signal and recognizes the sentence. It then informs the browser with the recognized sentence and extracted semantics.
 - **Speech output**—Like the speech input, the speech output was created in HTML5, and communicates with a service via WebSockets. After receiving the message from the IM, it requests the service to convert the text to speech signal and then reproduces it.

- **Touch**—Usually touch modalities are tied to the graphical user interface modality, i.e., the area where we touch is where graphics are rendered. Considering that, in the smart home scenario, the place where information is shown (e.g., projected on a wall) is not, necessarily, the place where users will interact, we have experimented to create a completely independent touch modality that creates an event such as “tap in coordinates x, y” with the information about the area or coordinates where the user touched the display (touch sensor).
- **Pointer**—This works as an alternative to touch that can be used at a distance from where the content is being shown by using a device to point to the area we want to touch. Since most portable devices, such as tablets and smartphones, are equipped with accelerometers and gyroscopes, using this kind of sensors it is possible to get the direction to which the device is pointing. Similarly to what can be done with the touch modality, we have proposed a modality that creates events with the coordinates to which the device is pointing.
- **Touch gestures**—This modality was also developed in HTML5/JavaScript. The user can draw simple symbols with his/her finger in the screen that are then interpreted, and an event is generated to the IM.
- **User face identification**—When a user is detected and identified, this generates an event, which is used to update the current context. Therefore, the information can be used to adapt the system to the current user or to its current location, e.g., flashing the lights in the room where the user is to signal some dangerous situation. This modality uses a Microsoft Kinect One to track the users and detect changes in their number—detecting arrivals and departures—and provides IDs for the occupants based in face identification. This modality, after being initiated by a message from the Interaction Manager, continuously tracks skeletons and the changes in their number. When the number of detected skeletons increases, the modality tries to track the face and identify the person using (although this can be replaced by any other equivalent or custom service) the face API from Azure cognitive services [80].
- **Speaker verification**—This modality is implemented in HTML5, JavaScript, and Azure Speaker Recognition service [81]. Using a Voice Activity Detector (VAD) algorithm, it sends the audio signal to the service only when the user stops speaking.
- **Graphical output modality**—This modality was also developed in HTML5 and JavaScript to be multiplatform and has some adaptation mechanisms, based on the context and the house divisions. It is capable of rendering a representation of the house with all the divisions, the temperature, lights, and name of the division. It can also render simple text, which can be used to transmit messages to the user and render a graphic to depict, for instance, the energy consumption for a certain month.

6.4.5 Smart home test bench

Encompassing the different resources described above, and to enable an actual deployment of our proposals in a real environment, a test bench for a smart home was created in the authors' research institute, capable of gathering information

and providing control for essential functionalities, and with the addition of integrated multimodal interaction support. Fig. 6.5 illustrates the overall scenario, obtained by distributing devices along several offices and meeting rooms mapped to common home spaces.

This test bench smart home uses ubiquitous computing at its core. We have deployed a home server, containing the set of components according to the multimodal interaction architecture, and these can be complemented by cloud services. All the rooms in the house were populated with a set of smart devices such as light bulbs, plugs, heater, and multiple technologies to support modalities. Fig. 6.5 provides a schematic depiction of our current setting, showing the home server and the different devices, in each room. The interaction modalities supported are, in each of the represented devices are: (1) microphone: speech input modality; (2) Kinect: user identification modality and speech input modality; (3) loudspeaker: speech output modality; (4) projector: graphical output modality; and (5) tablet: touch modality, speech input modality, graphical output modality, and speech output modality.

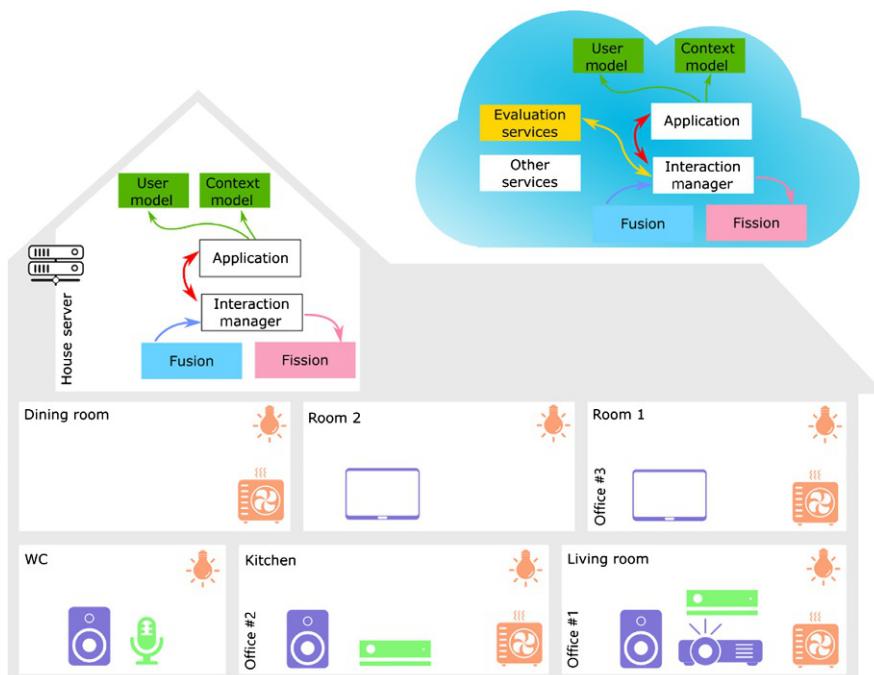


FIG. 6.5

Test bench smart home, showing the overall resources available, including the home server, cloud services, smart devices, and interaction modalities. For the sake of simplicity, the connections between devices were omitted.

6.5 Ubiquitous human-home interaction

The integrated house, particularly the data and interaction infrastructure, provides the basis for the assistant that handles all interaction with the house, from inside or outside. Despite being conceptually a unique assistant—with the working name of Igor—depending on the use, they can be perceived by inhabitants as different: one exploring language and speech, the other multimodal interaction (see Fig. 6.6). These are the limited cases, in general the assistant multimodal, which makes strong use of spoken and written language. To simplify presentation, in the following subsections we describe the two faces of the assistant.

6.5.1 Multimodal interaction

The first proof-of-concept for the multimodal interactive part of the assistant was built on top of the multimodal framework, thus profiting from a rich set of interaction modalities. For these first developments, we concentrated our efforts on the scenario in which a house occupant is using a tablet.

The application logic runs on the house server and receives the events from the interaction manager reacting accordingly by interfacing with the home control service. The application can read and update the user and context models. For instance, if the user changes location and the application have this knowledge, it updates the context model. Alternatively, if the assistant needs the user location it can query the context model to try to find it.

During the development of Igor, only some minor, but necessary configurations on the modalities had to be made to adapt to the specific context of use, notable examples being the definition of a proper grammar for the speech input modality and the rules for the combination of events in the fusion engine.

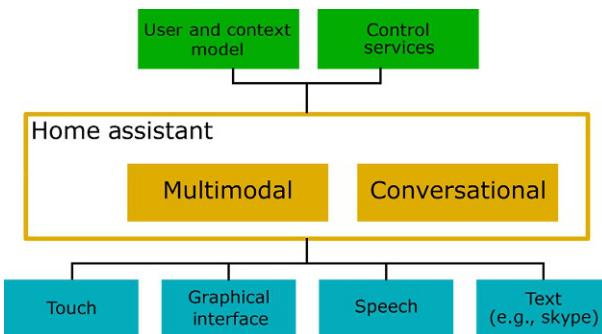
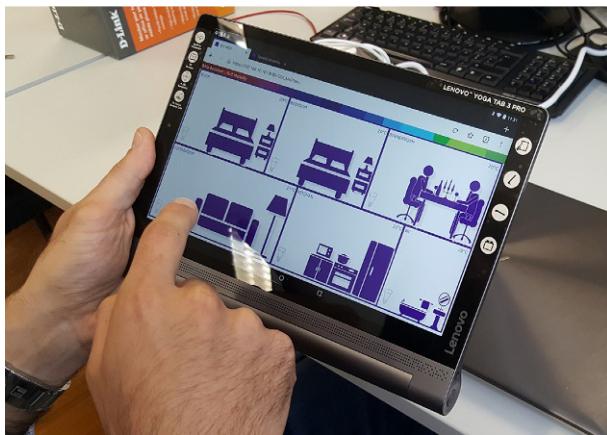


FIG. 6.6

Diagram of overall system elements with the assistant including a wide range of multimodal interaction capabilities and including a conversational module as alternative paths for similar tasks.

**FIG. 6.7**

User interacting with the multimodal assistant using a tablet. A graphical user interface, providing a depiction of the main rooms is presented.

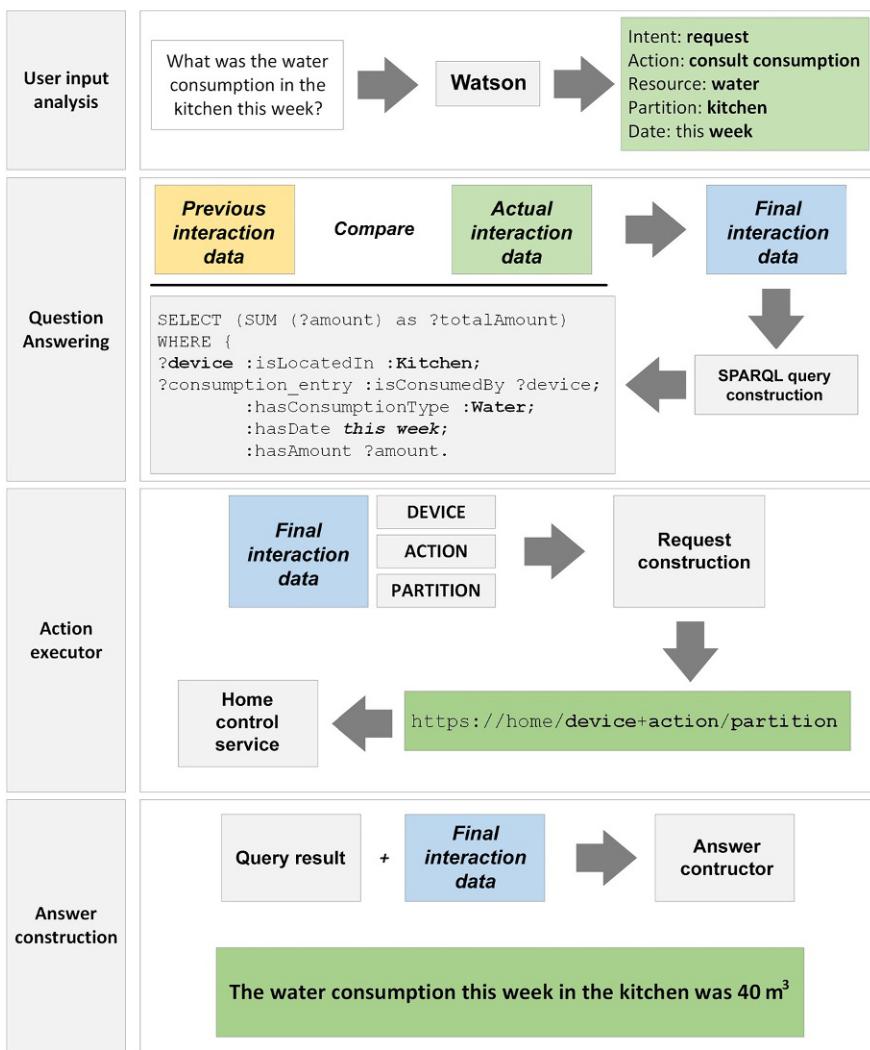
Fig. 6.7 depicts a user interacting with the assistant using a tablet. Through the assistant, the interaction with the home is centralized, aggregating the multidevice complexity of the smart home in an interactive ecosystem, and is potentially more adaptable to different audiences. However, to fulfill our vision, we needed conversational features, as described in [Section 6.5.2](#).

6.5.2 Conversational capabilities

The architecture of the developed system consists of four distinct modules (see [Fig. 6.8](#)), which allow processing user's requests and commands and returning a response to the user: (1) user input analysis; (2) question answering; (3) action executor; and (4) answer construction. Although the modules are distinct, they are tightly interconnected, since they share the output information with each other.

For a better understanding of the system workflow, [Fig. 6.8](#) depicts a concrete example of analysis for a possible user interaction: “What was the water consumption in the kitchen this week?,” with the first column referring to the module and the second a depiction of the process undergoing in that module as detailed in what follows:

- (a) **User input analysis** is the first processing step and aims to identify the user's intent, what action to take and the involved entities using the Watson service. Identifying the user's intent is an important task, since it allows knowing if the user wants to ask about something or control any of the appliances or other devices. Analyzing Watson's output, it is possible to understand that it identified that the user's intent was to request some information about water (resource) consumption (action) in a specific partition of the house and in a specific date range.

**FIG. 6.8**

Example of system modules processing.

- (b) The second row in Fig. 6.8 represents the **Question Answering** module that, using the relevant identified information, from the previous module, compares it to information from previous interaction (if exists) to identify the final user's intent.

After comparison, the system automatically composes the final SPARQL query to obtain the desired information from the system's database (managed by Apache Jena triple store).

Comparing information between interactions is what makes our system able to maintain the dialogue with the user and perform requests in multturn interactions.

In the case of the presented example, as no previous interactions existed, the final interaction data is the same as what Watson identified. The resulting SPARQL query from user's input is also shown. The system automatically constructs the semantic query from the user's natural language input. The presented SPARQL query can be read as follows: select and sum all the consumption registries of the appliances located in the kitchen that consumed water this week.

- (c) As previously mentioned, the system is able to identify requests and commands. In the case of commands, some control action must occur, for the appliances or other devices, and is handled by the **Action Executor** module. The developed system is able to control the state (on/off) of the different home devices, such as lights. These actions are executed by the Home Control Service and can be called using the automatically constructed URL consisting of the device, action, and partition information. The presented example is not handled by this module since the user's intent is to request some information and not command.
- (d) The last module, the **Answer Construction**, generates the final answer using the results from the previous modules: the Question Answering module, in case of an information request, or the Action Executor module, in case of commands. To generate the natural language response to requests, the query result from the Question Answering module is used and, for commands, the module considers the confirmation of correct command execution from the Home Control Service. To construct natural language responses, the system fills predefined templates using the words from the user's input and random synonyms (e.g., energy, electricity, power) for these words in order to make the answer more natural.

Finally, the answer is presented to the user in synthesized voice or text depending on the initial user interaction form.

6.5.2.1 Watson training

As previously explained, Watson is used in the analysis of user's input with the main purpose of identifying the intent and involved entities. However, this needs to be trained to the specific scenario at hand. Watson allows configuring three different types of information:

- Intents—adding new intentions (e.g., requests) and a set of phrases that correspond to this intent (e.g., “What was the water consumption?,” “How much water did I spend?”).
- Entities—adding new entities for any context, with our scenario involving entities related with the context of intelligent houses (e.g., partition, resource).

- Dialogue—adding a set of default phrases that will be returned when a specific intent is detected. For example, if the input phrase is “Hello assistant!,” Watson classifies it as a “Greet” intention and returns a random phrase from the predefined set of phrases for this intent. The “Dialogue” configuration is what differentiates Watson from other tools and, in our opinion, is extremely useful for the development of a conversational assistant without additional programming.

Watson was configured based on the previously presented ontology in order to make the system consistent. Watson can be trained in two ways: either manually, through the web interface, or automated by using its API. It is important to note that the initial training of Watson is not limiting to what can be performed or added to the system. For instance, when new appliances are added to the house (registered), Watson is trained online to consider further this new content expressed by the following fields:

- (a) **name**—the appliance ID;
- (b) **ontology type**—the ontology type for the device, since it is possible to have several devices of the same type;
- (c) **location**—the specification of the home partition where the appliance is located;
- (d) **synonyms**—a set of synonyms of the device (e.g., computer, PC, laptop);
- (e) **error**—representing the list of possible appliance errors, their description, and possible solutions; and
- (f) **interactions**—listing the possible interactions with the appliance, such as: consulting the appliance consumption, getting or changing the appliance status (on/off), and knowing the current errors, their descriptions, and their solutions.

6.5.2.2 Device registering

Taking into account that one of the main characteristics of smart homes (and smart environments, overall) is the dynamic nature of the pool of devices that it contains, our goal of blending the different devices into an interactive ecosystem requires a systematic method to expand the capabilities of the assistant of integrating novel devices that it can recognize and handle. Therefore, the system allows the addition of new equipment in an easy and generic way. Using an XML file, it is possible to provide basic information about the equipment (metadata) so that the system understands what kind of skill is added and how the assistant can interact with it. As an example, Fig. 6.9 illustrates the structure and elements that constitute the XML file for the registration of a new piece of household equipment.

Specifically, the information present in Fig. 6.9 is related to the addition of a new water heater named “GEOS,” from Bosch, which allows making requests about status, temperature, resources consumption, and errors of the equipment. The field *ontology type* is important to identify the type of the appliance, as the house can integrate a set of the appliances of the same type like TV and lights, which can easily be grouped or differentiated using the ontology type. A set of synonyms is also very important to make the system understand the various forms that the user can call the appliance (e.g., geos, water heater). If the appliance supports error registering,

```

<device>
  <name>geos</name>
  <ontology_type>waterHeater</ontology_type>
  <location>Kitchen</location>
  <synonyms>
    <synonym>geos</synonym>
    <synonym>esquentador</synonym>
  </synonyms>
  <errors>
    <error>
      <id>...</id>
      <description>...</description>
      <solution>...</solution>
    </error>
  </errors>
  <interactions>
    <interaction>consult_consumption</interaction>
    <interaction>consult_state</interaction>
    <interaction>consult_error</interaction>
    <interaction>control_on</interaction>
  </interactions>
</device>

```

FIG. 6.9

XML structure to new appliance registering.

it is mandatory to provide the set of possible errors, their descriptions, and their solutions, information which can normally be found in the user's manual. In this way, all this information is inserted into the database and, in the future, the user can consult the appliance's errors, ask the system about the error definition, and receive suggestions to resolve these errors.

6.6 Illustrative scenarios

Illustrating the potential of the proposed vision, of the developed test bench smart home and of the assistant, in what follows a few representative examples of interacting with the smart home are presented. The main goal of these examples is not complexity, but the illustration of actual features already supported in the test bench smart home.

6.6.1 Chatting with the smart home

In this scenario, the user wants to obtain some information about the smart home and uses the Skype application for this purpose. The use of this messaging service allows full control over the smart home, from anywhere in the world, obtaining the status

and controlling home appliances and other devices remotely using the chat mode (Fig. 6.10). In addition, it is extremely helpful to prevent the undesired consumption of resources by appliances.

One important stage of the communication between the user and the home is onboarding. It consists in providing an informative first message (or sequence of messages) that enable the user to have a first idea of what can be done. Fig. 6.11 illustrates the system on-boarding, where the user says “Hello” and the assistant responds with the explanations of his capabilities and gives some examples of use.

Fig. 6.12 depicts a situation of information completing. The user asks about the consumption, “What was the consumption?” and, since it is not possible to infer the target resource, the assistant replies asking, “of water, energy or gas?” In this way, a

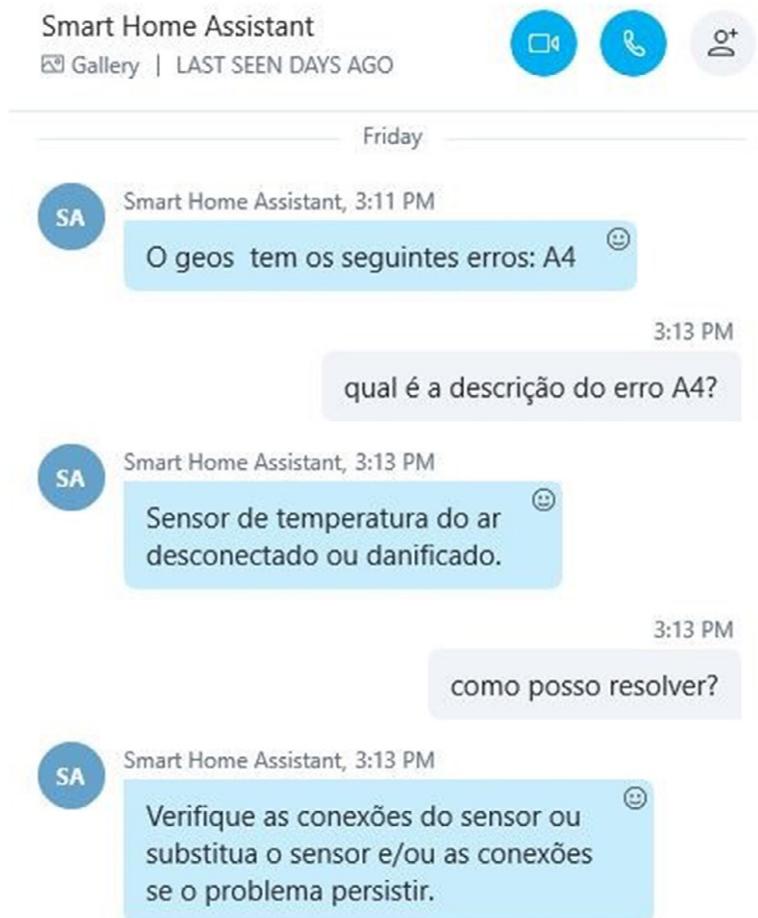


FIG. 6.10

Interacting remotely with the home via a chat in Skype.

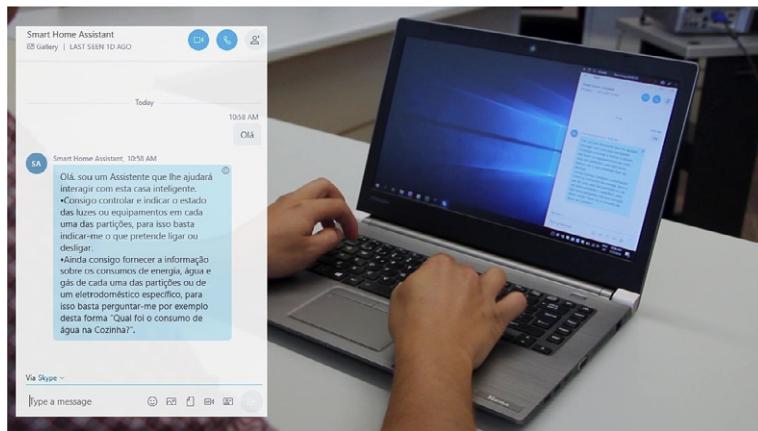


FIG. 6.11

Assistant onboarding. In this and the following figures, since our assistant works in European Portuguese, please refer to the text for an overall description of what is happening.

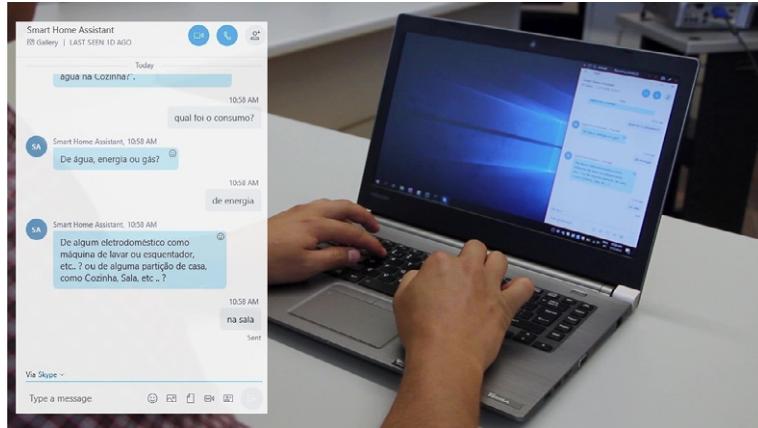


FIG. 6.12

User answering to an assistant request.

more natural dialogue is established between the user and the system and, also, our tests revealed that it is the best way to make the user respond only with a short “of energy” instead of repeating the entire question: “What was the energy consumption?”

Afterwards, the assistant asks the user if he is interested in a specific room of the house or appliance, to which the user responds, “In the living room.”

The conversational assistant provides the possibility to consult resource consumption for the appliances that rely on any resource (e.g., gas or electricity) to work.

In Fig. 6.13, the user asks, “What was the water consumption for the washing machine, in the kitchen?”, specifying the appliance’s location since the house has more than one washing machine. If the user had not specified the location, the system would ask for it.

Finally, Fig. 6.14 shows the user asking about the water heater’s state, and the assistant responds that the water heater has some errors: A1, A0, and A4. These represent the errors’ IDs defined by the manufacturer and mean nothing to the user.

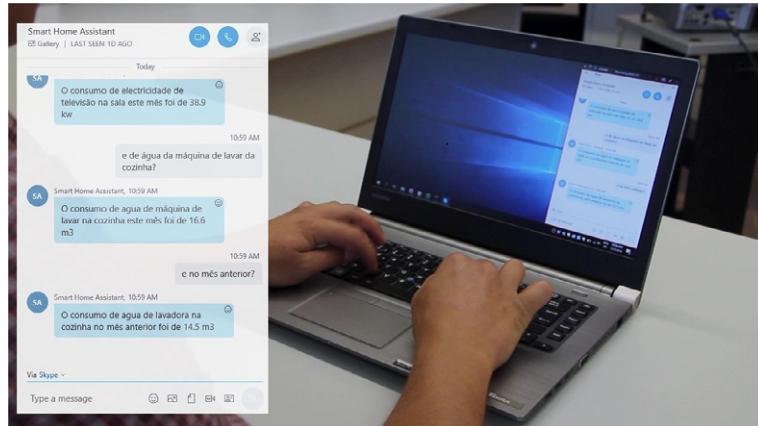


FIG. 6.13

Specific appliance water consumption consulting.

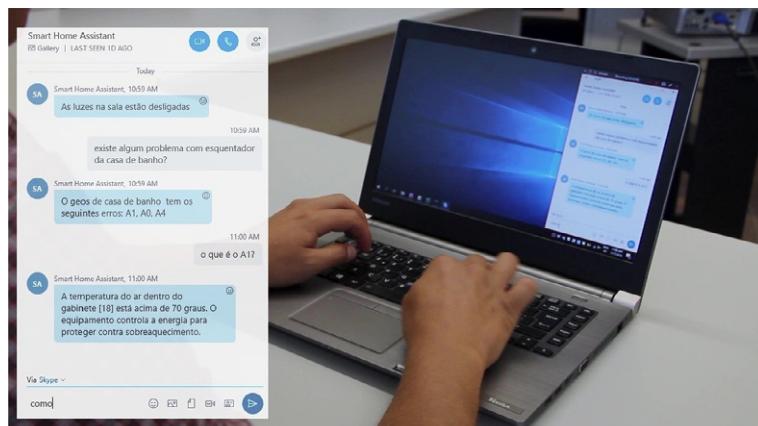


FIG. 6.14

Consulting the water heater errors and their meaning.

Therefore, the user asks, “What does A1 stand for?,” and the system responds with the detailed error description. In addition, the user can inquire about the error’s solution and try to implement it.

6.6.2 Sending an email to the home

When the user (MK) is away from home, in a meeting, or simply intends to make a command without enrolling in a conversation with assistant (smart home assistant), the use of email can be a simple and convenient method. In this regard, the assistant is also able to receive and send emails to the user. The email interaction was implemented using the Gmail API, which is a simple way to implement this feature. The user can send email with any request or command and, after analysis, the assistant replies with the result.

In Fig. 6.15 the user (MK) sends an email to the assistant with the content “Turn off the lights in the kitchen” and a short time later, the assistant (SA) replied with the confirmation email: “The lights in the kitchen have been turned off!”

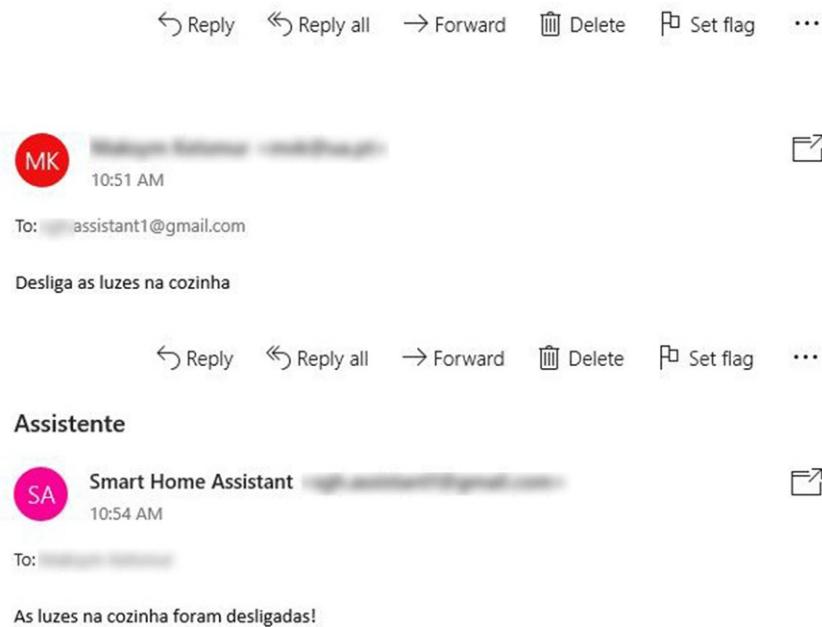


FIG. 6.15

Communicating with assistant through email. In the example shown, using Portuguese, the user sends an email (at top) requesting turning off the kitchen lights and receives an answer email confirming that they have been turned off.

6.6.3 Assisting the very young and the elderly

Fig. 6.16 depicts one example of interaction among the assistant, a grandmother called Maria, and young António. They decide to check and compare the electricity consumption among home partitions where they spend most of their time in order to see who spends the least amount of energy in the house. They can both interact, in the same conversation, and the assistant asks for missing information and understands the conversation context.

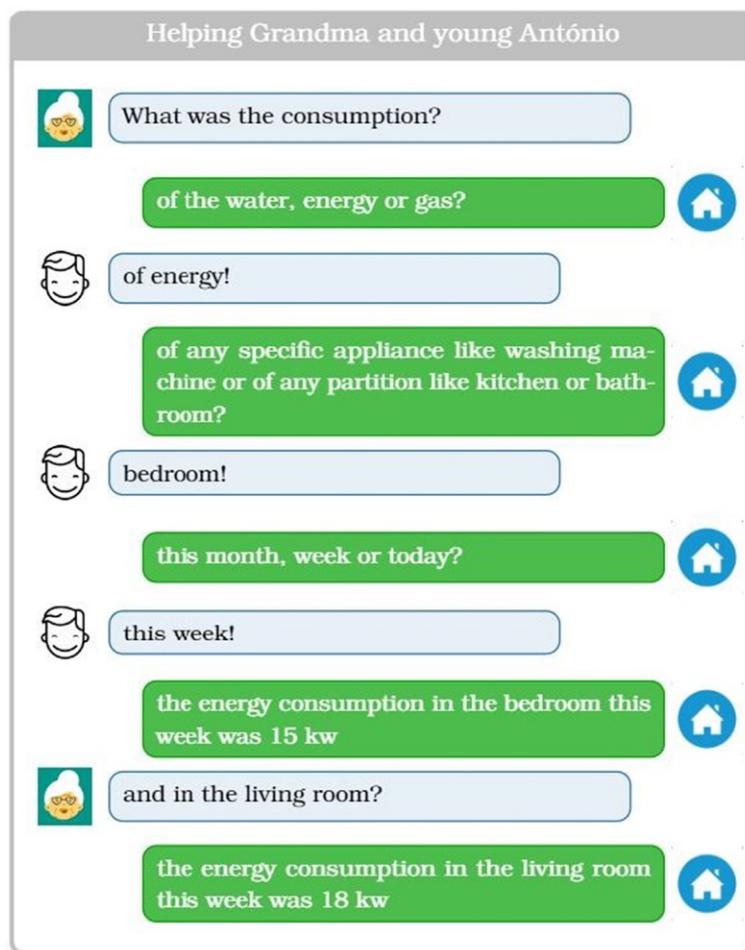


FIG. 6.16

Illustrative example showing the assistant in use by Grandma and young António, demonstrating the use of context of discourse to simplify interaction.

6.6.4 Multimodal interaction across rooms

The multimodal assistant can be used in any room of the house, for instance, if a tablet is available or any other interactive device such as a media center. Using one of more of the different possible ways to interact, the user can, for example, use speech and relying on the system's context model turn on the lights of the room where the user is. Fig. 6.17 illustrates this scenario. The figure is divided into four parts. On the top left the user is seen using the assistant through a tablet while in the living room; on the top right the image of the (test bench) office; on the bottom left the action performed by the user, in this case, what the user spoke; and on the bottom right the user interface for the assistant. After issuing the command, the light in the living room is turned on while, in the office (top right), everything stays the same.

Fig. 6.18 illustrates the scenario to turn on the light in the office. Since the office is not the room where the user is located, he needs to provide more information to turn on the light. As such, after the user utters a voice command to turn on the light he then touches the select area on the tablet corresponding to the part of the office where the light should be turned on. The two events are combined, creating a new event to turn on the light in the office even though the user is in a different room at the time.

Afterwards, the user can ask for information about the lights. Fig. 6.19 illustrates a scenario where the user asks the assistant information regarding the lights' state, particularly, in which rooms they are switched on. The assistant responds to the user using speech synthesis.

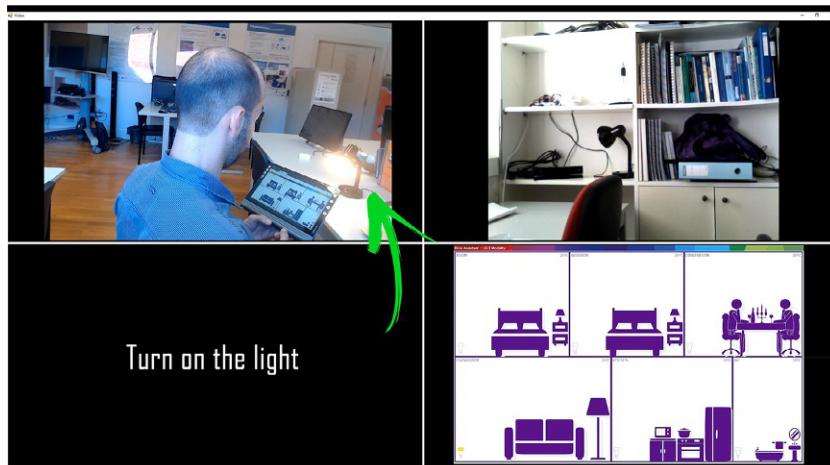


FIG. 6.17

Illustration of a usage scenario of the assistant to turn on the light of the current room using speech.

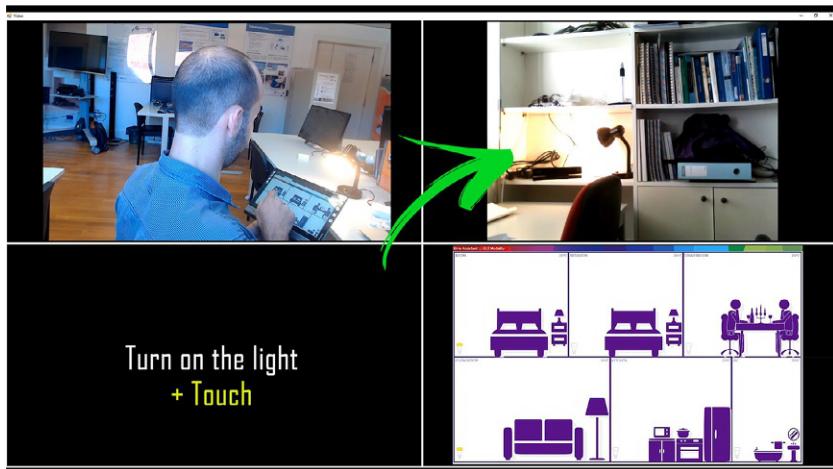


FIG. 6.18

Illustration of a usage scenario of the assistant to turn on the light of the virtual living room using speech and touch, creating a fusion of events of the two modalities.

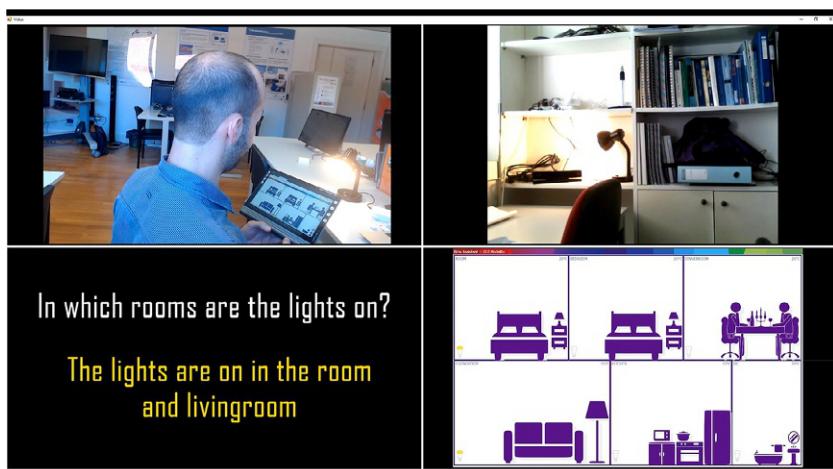


FIG. 6.19

Illustration of a usage scenario in which the assistant, used through a tablet, retrieves and informs the user about the light state.

Fig. 6.20 presents another feature of the assistant that, being asked about the water consumption of the current month, shows a chart depicting the available information. This is actually one of the aspects in which the assistant is planned to evolve, in the future, by choosing the most suitable way to present different information depending on its nature and on the available output modalities.

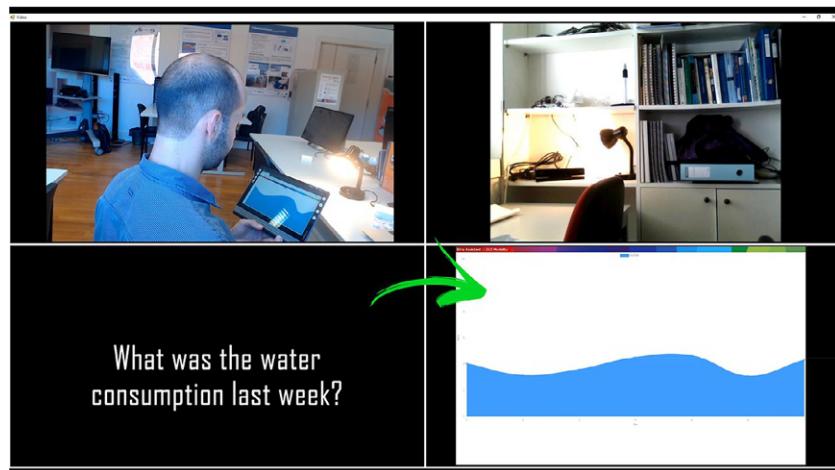
**FIG. 6.20**

Illustration of a usage scenario: the user asked for the water consumption for the past week and the assistant depicts the information to the user using a simple chart.

6.7 Conclusion

This chapter describes ongoing work on the proposal and deployment of an integrated view for multimodal interaction of humans with their homes, inspired in human-human interaction, not only considering when the user is physically at home, but also when at a distance.

One of the notable characteristics of our proposal is that the interaction with the home is mediated by an assistant that supports interaction through the articulation of two important characteristics: (a) multimodality, providing a wide range of alternatives for interacting with the home, exploring (but not limited to) the capabilities and advantages of mobile devices; and (b) conversational capabilities, to bring human-home interaction to a more natural setting, potentially widening the ability to adapt to a larger audience and reducing the gap between the user and the “behind the curtains” complexity of the smart home. Another notable point of our contributions is the proposed home infrastructure, integrating sensors, servers, services, semantic databases, and a multimodal multiplatform interaction framework. It is essential to support a large part of the assistant’s features, particularly how it can access and convey information to the occupants.

Even though the assistant is at an early stage of development, working as a first proof of-concept for our vision and as a test bench of new ideas, it is already possible to have a useful conversation (dialogue) with it, as illustrated in the results section. Our current efforts are being focused on harnessing the assistant with additional functionalities, particularly regarding its conversational features, such as the

capability of handling phone or Skype calls and, most importantly, from our perspective, having initiative.

One important future line of development, for which we have already taken a few decisive steps, is the assessment of the proposed approaches in place and their evaluation by users, feeding the design and development cycle with data on the refinement and definition of requirements. This is not a simple task given the distributed multimodal nature of the proposed vision and the diversity of contexts and audiences. At this time, we are focusing on integrating a dynamic evaluation system with the multimodal framework [82, 83], to obtain contextualized data on the interaction patterns and have harnessed the conversational assistant with an annotated data collection feature for subsequent analysis on performance and adaptability to the most common user's inquiries. This, along with an increasingly complex smart home test scenario—more sensors, actuators, and interactive devices—should enable a continued user-centered evolution of our proposal.

The consideration of the house as an interlocutor, and the adoption of conversational assistants as a more natural way of interaction, can also motivate the consideration and evolution an audiovisual component to complement those alternatives already discussed. An approach to pursue such a path has been proposed by the authors in Ref. [84].

Acknowledgment

Research partially funded by project Smart Green Homes (POCI-01-0247-FEDER-007678), a co-promotion between Bosch Termotecnologia S.A. and the University of Aveiro, and IEETA Research Unit funding (UID/CEC/00127/2013). Samuel Silva's work is funded by Portugal 2020 under the Competitiveness and Internationalization Operational Program, and by the European Regional Development Fund through project SOCA—Smart Open Campus (CENTRO-01-0145-FEDER-000010).

References

- [1] D.J. Cook, How smart is your home? *Science* 335 (6076) (2012) 1579–1581.
- [2] T. Hargreaves, C. Wilson, Analytical framework for research on smart homes and their users, in: *Smart Homes and Their Users*, Springer International Publishing, Cham, 2017, pp. 15–34.
- [3] T. Koskela, K. Kaisa Väänänen-Vainio-Mattila, Evolution towards smart home environments: empirical evaluation of three user interfaces, *Pers. Ubiquitous Comput.* 8 (3–4) (2004) 234–240.
- [4] M. Turk, Multimodal interaction: A review, *Pattern Recogn. Lett.* 36 (2014) 189–195.
- [5] M. Bodell, D.A. Dahl, I. Kliche, J. Larson, B. Porter, *Multimodal Architecture and Interfaces*, W3C, 2012.
- [6] D.A. Dahl, The W3C multimodal architecture and interfaces standard, *J. Multimodal User Interfac.* 7 (3) (April 2013) 171–182.

- [7] A. Sciuto, A. Saini, J. Forlizzi, J.I. Hong, Hey Alexa, What's up?: a mixed-methods studies of in-home conversational agent usage, in: *Proceedings of the 2018 on Designing Interactive Systems Conference 2018*, Pages 857–868. ACM, 2018.
- [8] V. Kępuska, G. Bohouta, Next-generation of virtual personal assistants (Microsoft Cortana, Apple Siri, Amazon Alexa and Google Home), in: *Computing and Communication Workshop and Conference (CCWC), 2018 IEEE 8th Annual*, pages 99–103. IEEE, 2018.
- [9] M. Porcheron, J.E. Fischer, S. Reeves, S. Sharples, Voice interfaces in everyday life, in: *Proc. CHI'18*. ACM, 2018.
- [10] A. Vtyurina, A. Fournier, Exploring the role of conversational cues in guided task support with virtual assistants, (2018).
- [11] S. Young, M. Gašić, B. Thomson, J.D. Williams, POMDP-based statistical spoken dialog systems: a review, *Proc. IEEE* 101 (5) (2013) 1160–1179.
- [12] Amazon, Amazon Alexa, <https://developer.amazon.com/alexa>. (Accessed 27 February 2018).
- [13] R. Kocielnik, D. Avrahami, J. Marlow, D. Lu, G. Hsieh, Designing for workplace reflection: a chat and voice-based conversational agent, in: *Proceedings of the 2018 on Designing Interactive Systems Conference 2018*, pages 881–894. ACM, 2018.
- [14] C. Rowland, E. Goodman, M. Charlier, A. Light, A. Lui, Designing connected products: *UX for the consumer internet of things*, O'Reilly Media Inc., 2015.
- [15] B.L. Risteska Stojkoska, K.V. Trivodaliev, A review of internet of things for smart home: challenges and solutions, *J. Clean. Prod.* 140 (2017) 1454–1464.
- [16] K. Shahzad, B. Oelmann, A comparative study of in-sensor processing vs. raw data transmission using ZigBee, BLE and Wi-Fi for data intensive monitoring applications, in: *2014 11th International Symposium on Wireless Communications Systems (ISWCS)*, pages 519–524. IEEE, August, 2014.
- [17] A. Haller, S. Cox, D. Le Phuoc, K. Taylor, M. Lefrançois, Semantic Sensor Network Ontology, Dec, 2016.
- [18] IPSO Smart Objects—OMA SpecWorks, <https://www.omaspecworks.org/develop-with-oma-specworks/ipso-smart-objects/>. (Accessed 13 September 2018).
- [19] LG, LG Smart Refrigerators: Powered by SmartThinQ IOT|LG USA, <https://www.lg.com/us/discover/smarthinq/refrigerators.jsp>. (Accessed 17 September 2018).
- [20] Samsung, Samsung POWERbot™ Robot Vacuum, <http://samsungpowerbot.com/>. (Accessed 17 September 2018).
- [21] Samsung, What is Q-Rator and how does it work?, <https://www.samsung.com/sg/support/home-appliances/what-is-q-rator-and-how-does-it-work/>, 2018. (Accessed 17 September 2018).
- [22] J. Krumm, *Ubiquitous Computing Fundamentals*, Chapman & Hall/CRC Press, 2010.
- [23] T. Hargreaves, C. Wilson, R. Hauxwell-Baldwin, Learning to live in a smart home, *Build. Res. Inform.* 46 (1) (2018) 127–139.
- [24] D.G. Lokhande, S.A. Mohsin, Internet of things for ubiquitous smart home system, in: *2017 1st International Conference on Intelligent Systems and Information Management (ICISIM)*, Pages 314–320. IEEE, October, 2017.
- [25] S. Pizzagalli, D. Spadolore, S. Arlati, M. Sacco, L. Greci, HIC: An interactive and ubiquitous home controller system for the smart home, in: *2018 IEEE 6th International Conference on Serious Games and Applications for Health (SeGAH)*, Pages 1–6. IEEE, May, 2018.
- [26] N. Weibel, A. Satyanarayan, A. Lazar, R. Oda, S. Yamaoka, K.-U. Doerr, F. Kuester, W.G. Griswold, J.D. Hollan, Hiperface: a multichannel architecture to explore

- multimodal interactions with ultra-scale wall displays, in: *ICSE'11: Proceedings of the 33rd International Conference on Software Engineering*, 2011.
- [27] B. Johanson, A. Fox, T. Winograd, The interactive workspaces project: experiences with ubiquitous computing rooms, *IEEE Pervasive Comput.* 1 (2) (2002) 67–74.
 - [28] S. Berti, F. Paternò, Migratory multimodal interfaces in multidevice environments, in: *Proceedings of the 7th International Conference on Multimodal Interfaces*. ACM, 2005.
 - [29] M. Blumendorf, D. Roscher, S. Albayrak, Dynamic user interface distribution for flexible multimodal interaction, in: *International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction on—ICMI-MLMI'10*, page 1, New York, New York, USA, November. ACM Press, 2010.
 - [30] S.A.M. Yusof, N. Zakaria, N.A.R. Muton, Timely trust: the use of IoT and cultural effects on swift trust formation within global virtual teams, in: 2017 8th International Conference on Information Technology (ICIT), May, 2017, pp. 297–303.
 - [31] A.N. Tuch, P. Van Schaik, K. Hornbæk, Leisure and work, good and bad: The role of activity domain and valence in modeling user experience, *ACM Trans. Comput.-Hum. Interact.* 23 (6) (2016) 35:1–35:32.
 - [32] J. von Grabe, How do occupants decide their interactions with the building? from qualitative data to a psychological framework of human-building-interaction, *Energy Res. Soc. Sci.* 14 (Supplement C) (2016) 46–60.
 - [33] H.S. Alavi, D. Lalanne, J. Nembrini, E. Churchill, D. Kirk, W. Moncur, Future of human-building interaction, in: *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA'16, pages 3408–3414, New York, NY, USA. ACM, 2016.
 - [34] F. Benzi, D. Fogli, G. Guida, Towards natural interaction with smart homes, in: *Proceedings of the 12th Biannual Conference on Italian SIGCHI Chapter*, CHItaly'17, pages 10:1–10:6, New York, NY, USA. ACM, 2017.
 - [35] J.N.A. Brown, A.J. Fercher, G. Leitner, Building an Intuitive Multimodal Interface for a Smart Home—Hunting the SNARK, Springer, 2017.
 - [36] T. Hargreaves, C. Wilson, *Smart Homes and Their Users*, Springer, 2017.
 - [37] H. Verma, H.S. Alavi, D. Lalanne, Studying space use: bringing HCI tools to architectural projects, in: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI'17, pages 3856–3866, New York, NY, USA. ACM, 2017.
 - [38] R.M. Tetlow, C. Philip Beaman, A.A. Elmualim, K. Couling, Simple prompts reduce inadvertent energy consumption from lighting in office buildings, *Build. Environ.* 81 (Supplement C) (2014) 234–242.
 - [39] D. Lallanne, et al., Human-building interaction in the smart living lab, in: *Future of Human-Building Interaction workshop, 34rd ACM Conference on Human Factors in Computing Systems (CHI 2016)*, 2016.
 - [40] N. Balta-Ozkan, B. Boteler, O. Amerighi, European smart home market development: public views on technical and economic aspects across the United Kingdom, Germany and Italy, *Energy Res. Soc. Sci.* 3 (2014) 65–77.
 - [41] A.-G. Paetz, E. Dütschke, W. Fichtner, Smart homes as a means to sustainable energy consumption: a study of consumer perceptions, *J. Consum. Policy* 35 (1) (2012) 23–41.
 - [42] M. Herczeg, The smart, the intelligent and the wise: roles and values of interactive technologies, in: *Proceedings of the First International Conference on Intelligent Interactive Technologies and Multimedia*, IITM'10, pages 17–26, New York, NY, USA. ACM, 2010.

- [43] B. Eggen, E. van den Hoven, J. Terken, *Human-Centered Design and Smart Homes: How to Study and Design for the Home Experience?* Springer International Publishing, Cham, 2014, pp. 1–9.
- [44] N. Balta-Ozkan, R. Davidson, M. Bicket, L. Whitmarsh, Social barriers to the adoption of smart homes, *Energy Policy* 63 (2013) 363–374.
- [45] S. Tirado Herrero, L. Nicholls, Y. Strengers, Smart home technologies in everyday life: do they address key energy challenges in households? *Curr. Opin. Environ. Sustain.* 31 (2018) 65–70.
- [46] S. Mennicken, J. Hofer, A. Dey, E.M. Huang, Casalendar: a temporal interface for automated homes, in: *CHI'14 Extended Abstracts on Human Factors in Computing Systems*, CHI EA'14, pages 2161–2166, New York, NY, USA, ACM, 2014.
- [47] S. Kartakis, M. Antona, C. Stephanidis, Control smart homes easily with simple touch, in: *Proceedings of the 2011 International ACM Workshop on Ubiquitous Meta User Interfaces*, Ubi-MUT'11, pages 1–6, New York, NY, USA, ACM, 2011.
- [48] IFTTT, IFTTT Helps Your Apps and Devices Work Together, <https://ifttt.com/>. (Accessed 17 September 2018).
- [49] Gideon, Home—Gideon Smart Home, 2018.
- [50] Inc WL, Wink|A Simpler, Smarter Home, <https://www.wink.com/>, 2017. (Accessed 17 September 2018).
- [51] Philips, Smart Home Lighting|Philips Hue, <https://www2.meethue.com/en-us>, 2018. (Accessed 17 September 2018).
- [52] C. Torres, W. Franklin, L. Martins, Accessibility in chatbots: the state of the art in favor of users with visual impairment, in: *International Conference on Applied Human Factors and Ergonomics*, Springer, 2018, pp. 623–635.
- [53] S. Yarosh, S. Thompson, K. Watson, A. Chase, A. Senthilkumar, Y. Ye, A.J. Brush, Children asking questions: speech interface reformulations and personification preferences, in: *Proceedings of the 17th ACM Conference on Interaction Design and Children*, Pages 300–312. ACM, 2018.
- [54] C. Pearl, *Designing Voice User Interfaces: Principles of Conversational Experiences*, O'Reilly, 2016.
- [55] Q. Vera Liao, M. Davis, W. Geyer, M. Muller, N. Sadat Shami, What can you do?: studying social-agent orientation and agent proactive interactions with an agent for employees, in: *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*, Pages 264–275. ACM, 2016.
- [56] T.-H.K. Huang, J.C. Chang, S. Swaminathan, J.P.B. Evorus, A crowd-powered conversational assistant that automates itself over time, in: *Adjunct Publication of the 30th Annual ACM Symposium on User Interface Software and Technology*, Pages 155–157. ACM, 2017.
- [57] J.O. Kephart, V.C. Dibia, J.B. Ellis, B. Srivastava, K. Talamadupula, M. Dholakia, A Cognitive Assistant for Visualizing and Analyzing Exoplanets, AAAI, 2018.
- [58] S. Bieliauskas, A. Schreiber, A conversational user interface for software visualization, in: *Software Visualization (VISSOFT), 2017 IEEE Working Conference on*, Pages 139–143. IEEE, 2017.
- [59] K.H. Ly, A.-M. Ly, G. Andersson, A fully automated conversational agent for promoting mental well-being: a pilot RCT using mixed methods, *Internet Interv.* 10 (2017) 39–46.
- [60] H. Albadawi, Z. Liu, Computationally-Efficient Human-Identifying Smart Assistant Computer, August 16, 2018. US Patent App. 15/636,422.

- [61] S. Pasupalak, J.R. Pantony, W. Hsu, W. Zhiyuan, P. Tregenza, K. Suleman, J. Simpson, A. McNamara, T. Ismail, Conversational Agent, February 21. US Patent 9,575,963 (2017).
- [62] S. Kopp, M. Brandt, H. Buschmeier, K. Cyra, F. Freigang, N. Krämer, F. Kummert, C. Opfermann, K. Pitsch, L. Schillingmann, et al., Conversational assistants for elderly users—the importance of socially cooperative dialogue, in: *AAMAS Workshop on Intelligent Conversation Agents in Home and Geriatric Care Applications*, 2018.
- [63] F.A. Brown, M.G. Lawrence, V. O'Brien Morrison, Conversational virtual healthcare assistant, (2017)January 3. US Patent 9,536,049.
- [64] P. Angara, M. Jiménez, K. Agarwal, H. Jain, R. Jain, U. Stege, S. Ganti, H.A. Müller, J. W. Ng, Foodie fooderson a conversational agent for the smart kitchen, in: *Proceedings of the 27th Annual International Conference on Computer Science and Software Engineering*, IBM Corp., 2017, pp. 247–253.
- [65] E. Fast, B. Chen, J. Mendelsohn, J. Bassen, M.S. Bernstein, Iris: a conversational agent for complex tasks, in: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 473. ACM, 2018.
- [66] T. Koseki, T. Kosaka, Multimodal spoken dialog system using state estimation by body motion, in: *Consumer Electronics (GCCE), 2017 IEEE 6th Global Conference on*, Pages 1–4. IEEE, 2017.
- [67] A. Bhalerao, S. Bhilare, A. Bondade, M. Shingade, Smart Voice assistant: A Universal Voice Control Solution for Non-visual Access to the Android Operating System, (2017).
- [68] J.-F. Kassel, M. Rohs, Valletto: a multimodal interface for ubiquitous visual analytics. in: Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems, 2018, pp. 1–6, <https://doi.org/10.1145/3170427.3188445>.
- [69] M. Ketsmur, M. Rodrigues, A. Teixeira, DBpedia based factual questions answering system, *IADIS Int. J. WWW/Internet* 15 (1) (2017).
- [70] B. Monika, F. Miroslav, Internet of things, in: InvEnt 2018: Industrial Engineering—Invention for Enterprise, 2018.
- [71] D. Jurafsky, J.H. Martin, *Speech and Language Processing*, third ed., (2017)draft edition.
- [72] R.S. Nickerson, On conversational interaction with computers, in: *Proc. ACM/SIGGRAPH Workshop on User-oriented Design of Interactive Graphics Systems*, UODIGS'76, 1977.
- [73] M. Quinderé, Comunicação Humano-Robô através de Linguagem Falada, (Ph.D. thesis), Universidade de Aveiro, 2013.
- [74] A. Lyazidi, S.M. Ondar, An ontology for home automation, in: *Conference on Intelligent Systems Design and Applications (ISDA), 2015 15th International*, Pages 260–265. IEEE, 2015.
- [75] J. Xu, Y.-H. Lee, W.-T. Tsai, W. Li, Y.-S. Son, J.-H. Park, K.-D. Moon, Ontology-based smart home solution and service composition, in: Proc. IEEE Int. Conf. Embedded Software and Systems, 2009 (ICESS'09), 2009, pp. 297–304.
- [76] Nuno Almeida, Multimodal Interaction—Contributions to Simplify Application Development, (Ph.D. thesis)Universidade de Aveiro, 2017.
- [77] N. Almeida, S. Silva, A. Teixeira, D. Vieira, Multi-device applications using the multimodal architecture, in: D. Dahl (Ed.), *Multimodal Interaction With W3C Standards—Toward Natural User*, first ed., Springer International Publishing, 2017, pp. 367–383.
- [78] A. Teixeira, N. Almeida, C. Pereira, M.O.e. Silva, D. Vieira, S. Silva, Applications of the multimodal interaction architecture in ambient assisted living, in: D. Dahl (Ed.),

- Multimodal Interaction with W3C Standards—Toward Natural User, first ed., Springer International Publishing, 2017, pp. 271–291.
- [79] N. Almeida, S. Silva, A. Teixeira, Design and development of speech interaction: a methodology, in: *Proceedings of HCI International 2014*, 2014.
 - [80] A. Del Sole, *Microsoft Computer Vision APIs Distilled: Getting Started with Cognitive Services*, Apress, Berkeley, CA, 2017.
 - [81] Microsoft, Speaker Recognition API, <https://docs.microsoft.com/en-us/azure/cognitive-services/speaker-recognition/home>, 2016. (Accessed 15 June 2018).
 - [82] C. Pereira, Dynamic Evaluation for Reactive Scenarios, (Ph.D. thesis)Universidade de Aveiro (MAPi), 2016.
 - [83] C. Pereira, N. Almeida, A.I. Martins, S. Silva, A.F. Rosa, M.O.e. Silva, A. Teixeira, Evaluation of complex distributed multimodal applications evaluating a TeleRehabilitation system when it really matters, in: Human Aspects of IT for the Aged Population. Design for Everyday Life, 2015.
 - [84] S. Silva, A. Teixeira, An anthropomorphic perspective for audiovisual speech synthesis, in: *Proc. BIOSIGNALS*, 2017, pp. 163–172.

A study on the emotional state of a speaker in voice bio-metrics

7

K.N.R.K. Raju Alluri, Anil Kumar Vuppala

*Speech Processing Laboratory, LTRC, KCIS, International Institute of Information Technology,
Hyderabad, India*

7.1 Introduction

Speaker recognition (SR) is a process of identifying a person from his or her characteristic voice [1, 2]. No two individuals sound identical because of their vocal tract shapes, larynx sizes, and other different voice production organs. In addition to these physical differences, each individual has his or her own speaking style, pronunciation pattern, choice of vocabulary, and so on. Because of all these factors, one can use voice as a bio-metric in addition to fingerprint and retinal scans. The SR task can be subdivided into speaker verification and speaker identification. Speaker verification is the task of determining whether the person is the one it is claimed to be or not. Speaker identification is the task of determining who is speaking in a set of known speakers [3]. The SR systems can be further classified as either text-dependent or text-independent based on the text to be spoken. In the text-dependent mode, both training and testing use the same text, whereas in the text-independent case there are no such restrictions [4]. In the present work, the text-independent speaker identification approach is considered. Applications of SR include access control, transaction authentication, law enforcement, speech data management, etc.

Development of the SR system uses different features which are extracted from the speech signal. A speech signal is the result of exciting a time-varying vocal tract system with time-varying excitation [5]. The information regarding speaker will be present in both system and source features. Most of the SR systems are modeled using Mel-frequency cepstral coefficients (MFCCs) and linear prediction cepstral coefficients, which represent vocal tract characteristics [1, 2]. In [6–8], features are extracted from the linear prediction (LP) residual as it contains significant information about the excitation source to model the SR systems. The features that capture excitation information are real cepstral coefficients. The state-of-the-art SR systems such as i-vector [9], GMM-UBM [10], and Gaussian Mixture Model-Support Vector Machine (GMM-SVM) [11] have achieved significant performance. In [12], DNNs are used for acoustic modeling in speech recognition. Speaker-specific characteristics using DNNs are explored in [13]. The success of DNNs in speech recognition enabled researchers to use DNNs for other speech-related tasks like SR and language identification [14–17].

The performance of SR depends on the features, models, and the conditions in which training and testing are done. The inconsistency in training and testing models causes performance degradation in SR tasks. The factors that cause performance degradation include background noise, channel effect, and speaker-based variability [18]. Speaker-based variability includes the health of the speaker, phonation style, vocal effort, disguise, and emotional state of the speaker [19].

SR in emotional conditions is a significant research problem in human-computer interaction because most humans use emotions extensively when conveying their message to others. Therefore, there is a need for recognizing the person in any emotional condition like sad, angry, happy, etc. This study can be used in many applications [20, 21], and can be used to enhance the speech recognition performance in telecommunications. The major advantage of this study can be observed in call centers; with emotionally intelligent automated systems, the service of a call center will go to the next level to satisfy the customer.

Most of the works in the area of SR are developed for neutral/normal speech, that is, the training and testing is done on neutral speech, but in practical scenarios, it is not always possible to speak in neutral mode. In the literature, it is reported that the performance of SR systems was degraded when there was a mismatch in training and testing emotions [18, 22, 23]. This performance degradation is mainly due to two reasons; they are mismatched emotions between the speaker models and the test utterances, and the articulating styles of certain emotions that create intense intraspeaker vocal variability [18]. In order to improve SR system performance, emotion state conversion incorporated into the models was reported in [24] and the neural network-based feature transformation was performed in [25]. The effect of adding emotional data to the neutral data in training is analyzed in [26, 27]. In [18], an emotional normalization technique is applied to remove the emotion bias in scoring. In our previous studies, Raju Alluri et al. [28, 29] reported on SR in emotional conditions on a fused database of German emotional speech database (EMO-DB) [30], IITKGP-SESC: Hindi [31], and IITKGP-SESC: Telugu databases [32]. Analysis of source and system features is done in [28] and it is observed that the source features are more affected due to the emotional state of the speaker. In [29], different modeling techniques are compared.

In [18, 24, 25], the relative improvement in the performance of SR performance is low. To improve the performance to a greater extent, in this study the importance of the need for emotional data in building the SR system is explored. Experiments are performed on the Amrittha emotional database [33]. Four classes (neutral, anger, happiness, and sadness) of emotions are considered for this study. Analysis of SR in emotional conditions is done using the parameters like strength of excitation (SoE), energy of excitation (EoE), fundamental frequency (F0), and duration. Further analysis is done using the distance between the GMMs employing Kullback-Leibler divergence. A two-stage emotional SR system is proposed to enhance the recognition accuracy: first, the emotional state of the speaker is detected using an emotion recognition system, and then the utterance is validated using the speaker models of detected emotion.

The rest of the chapter is organized as follows. Databases used in this study are described in [Section 7.2](#). The baseline system is described in [Section 7.3](#). Analysis of SR in emotional conditions is explained in [Section 7.4](#). [Section 7.5](#) presents different strategies used for SR in emotional conditions followed by the results of the present work. A conclusion and future scope are addressed in [Section 7.6](#).

7.2 Emotional database

The experiments are conducted using the Amritha emotional database [33]. This database is a combination of three languages (Tamil, Malayalam, and Indian English); each language has 10 speakers data in multiple sessions. The basic emotions present in this database are neutral, anger, happiness, and sadness. This emotion database is developed using the emotionally biased utterances. Each speaker has 220 utterances in each emotion, that is, 30 speakers with 4 emotions contain 26,400 ($30 * 4 * 220$) utterances. The sentences considered for recording are taken from different social media sources and sentences from the commonly occurring contexts. Based on the context, these prompts are divided into each emotional state. All speech recordings are done at a 48 kHz sampling rate on dual channel. In each language the data is collected from five males and five females except in Indian English where it is six males and four females.

7.3 Baseline emotional SR system

In this study, the SR system used for experiments is based on GMM-UBM [10]. The basic components of a generic GMM-UBM system are described in [Fig. 7.1](#). The SR system will have two phases in its development: training/enrolling and testing/recognition.

The enrollment phase contains two basic steps. The first one is feature extraction and the second one is modeling. In the feature extraction step, the speech signal is passed through a speech activity detector to remove the silence and nonspeech regions from the signal; later the features are extracted from small segments of the signal to ensure the quasistationary nature of the speech signal. In the case of the GMM-UBM system, a large GMM is built using a large number of speakers' data that are different from that of enrolled speakers. This large GMM is known as the Universal Background Model (UBM); later the speaker GMM models are adapted from the UBM with the MAP by adapting means. At the end of the enrollment phase, each speaker will have one model.

In the testing phase, the test speaker speech signal is passed through the feature extraction step to remove nonspeech regions, and the features are extracted in the same way as in the enrollment phase. These features are considered to get scores from each model present in the enrollment phase; the model which gives maximum score will be the claimed speaker.

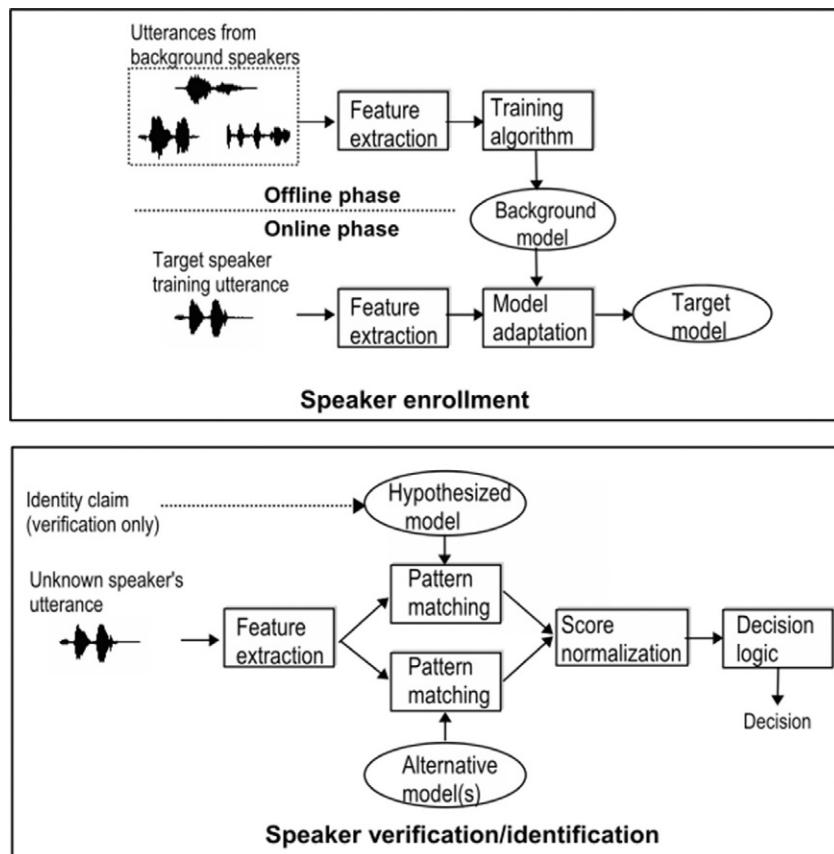


FIG. 7.1

Components of generic speaker recognition system using GMM-UBM.

Adapted from T. Kinnunen, H. Li, *An overview of text-independent speaker recognition: from features to supervectors*, *Speech Commun.* 52 (1) (2010) 12–40.

First, we considered 30 speakers' neutral data from the subset of the Amrittha emotional database. Each speaker has 50 utterances each of approximately 3-s duration. We consider 40 utterances for training and the remaining 10 utterances for testing. That means for training, we have considered 120-s data and testing 30-s data. An energy-based threshold is used for speech activity detection. In this study, MFCCs are used to represent speaker-specific information. Thirteen-dimensional MFCCs plus velocity and accelerated coefficients are computed for 20 ms frame length and 10 ms frameshift. The UBM is built on the omitted neutral utterances of all speakers 5100 ($30 * 170$) using 512 Gaussian mixtures. Individual speaker models are adapted from UBM with the MAP by means only. During testing, log likelihoods are computed for each model and are removed from the UBM score and the one that gives the max score is assigned as the claimed speaker.

Table 7.1 Performance (%) of SR system in emotional conditions using MFCC and UBM.

Accuracy (%)	Emotion type of test utterances			
	Neutral	Anger	Happiness	Sadness
Emotion type of speaker models	Neutral	99	48.33	54.30
	Anger	47	96.33	57
	Happiness	60	51	95
	Sadness	42.66	24.66	38.60
				97.33

A similar experimental setup is repeated for all the emotions present in the database. The preliminary results of the SR system using GMM-UBM trained on one emotion and tested with different emotions are reported in [Table 7.1](#). For example, the entry in row 2, column 3 in [Table 7.1](#) represents the training models on anger, and the testing emotion is happiness.

From the results in [Table 7.1](#), it can be seen that recognition performance declines to a greater extent when there is a mismatched emotion. For example, when speaker models are trained with anger and tested with neutral, anger, happiness, and sadness, the accuracies are 47, 96.33, 57, and 23, respectively. The performance of the matched case is 99, 96.33, 95, and 97.33 for neutral, anger, happiness, and sadness, respectively. The reasons for the performance are mainly due to mismatched emotions between the speaker models and the test utterances, and the articulating styles of certain emotions that create intense intraspeaker vocal variability [18]—for example, the result obtained with sadness trained on anger. Further analysis of these results is carried out in the next section.

7.4 Analysis of SR system performance in emotional conditions

In this section, SR in emotional conditions is analyzed using different factors. The results in [Table 7.1](#) suggest that the recognition performance tends to improve when the training and testing emotion is the same. These results indicate to us that the mismatched emotion is one of the primary reasons for performance degradation. The vocal feature characteristics extracted from neutral speech deviate from the emotional speech, just as we can observe from the results in [Table 7.1](#). In [34, 35], vocal features of different emotions are compared, and this comparison is reported in [Table 7.2](#). From this table, it can be observed that different emotions show different characteristics.

To validate the evidence in [Table 7.2](#), a set of parameters are computed from the speech signal. The parameters investigated in this study are F0, SoE, EoE, and duration of utterance [36–38]. To extract these features, zero frequency filtering (ZFF) analysis [39] and LP analysis [6] are used.

Table 7.2 Emotions and speech parameters.

	Anger	Happiness	Sadness
Rate	Slightly faster	Faster or slower	Slightly slower
Pitch average	Very much higher	Much higher	Slightly lower
Pitch range	Much wider	Much wider	Slightly narrower
Intensity	Higher	Higher	Lower
Voice quality	Breathy, chest	Breathy, blaring tone	Resonant
Pitch changes	Abrupt on stressed	Smooth, upward inflections	Downward inflections
Articulation	Tense	Normal	Slurring

Source: Taken from I.R. Murray, J.L. Arnott, *Toward the simulation of emotion in synthetic speech: a review of the literature on human vocal emotion*, *J. Acoust. Soc. Am.* 93 (2) (1993) 1097–1108, and R. Cowie, E. Douglas-Cowie, N. Tsapatsoulis, G. Votsis, S. Kollias, W. Fellenz, J.G. Taylor, *Emotion recognition in human-computer interaction*, *IEEE Signal Process. Mag.* 18 (1) (2001) 32–80.

7.4.1 Fundamental frequency

The fundamental frequency (F_0) is extracted using the ZFF method reported in [40]. In this method, the speech signal is passed through a cascade of two ideal digital resonators located at 0 Hz, followed by trend removal. The resultant signal is called the ZFF signal. The negative to positive zero crossings of the ZFF signal correspond to the instants of significant excitation (glottal closure instants [GCIs]) of the vocal tract. The interval between two successive GCIs gives the fundamental period T_0 , and thereby the instantaneous fundamental frequency is given by $F_0 = 1/T_0$.

7.4.2 Strength of excitation

The SoE is a correlate of the impulse like excitation at GCI. SoE is calculated from the slope of the ZFF signal at epoch locations.

7.4.3 Energy of excitation

The LP residual gives an approximation to the excitation component of the speech signal. EoE is computed using the samples of the Hilbert envelope of the LP residual over 2 ms around each epoch.

7.4.4 Duration

The ratio between the total number of samples to sampling frequency is taken as the duration of the speech signal. This parameter can be correlated to the speech rate.

In this study, a male (M006M) and female (F001E) speaker are considered to evaluate the vocal source characteristics in different emotions in terms of the above-mentioned features. Here F001E represents a female (F) speaker with id

(006) in the Indian English (E) database. For each speaker, the mean and variance of SoE, EoE, and F0 are computed along with the average duration for training neutral data (40 utterances) and testing data in each emotion (10 utterances). The results for M006M and F001E are reported in **Tables 7.3** and **7.4**, respectively.

From the results in **Table 7.3**, it can be seen that the average fundamental frequency for anger and happiness test utterances are higher, and for the sadness, it is a bit lower than that of neutral utterances. A similar observation can be observed for the standard deviation of F0, which is wider for anger and happiness, whereas for sadness it is narrower than that of neutral utterances. From the duration parameter, we can assess that anger and happiness are of shorter duration, whereas sadness is of slightly higher duration. F0-mean, F0-std, and duration parameters in **Table 7.4** can be correlated to that of pitch average, pitch range, and speech rate in **Table 7.2**. In addition to F0 and SoE, the emotions can be differentiated with EoE as shown in **Table 7.2**.

The results in **Table 7.4** also follow a similar pattern to those of **Table 7.3**. These results give us an idea of how the vocal features of different emotions change. The changes in vocal features due to the emotional state of a speaker will reflect in the corresponding signal there by the features extracted from these emotional speech signal will be a distinct feature distribution to that of neutral feature distribution. Because of these deviations, the performance of SR in emotional conditions will degrade. The extent of degradation depends on the intensity of vocal-effort mismatch. If this mismatch is higher, there will be a greater aggravation in SR system performance. To support further the statement “The feature distribution of emotional utterances have deviated from that of neutral utterances,” KL-divergence distance

Table 7.3 Emotions and speech parameters for M006M.

	F0-mean	F0-std	SOE-mean	SOE-std	EOE-mean	EOE-std	Avg-duration
Train-neutral	147.60	25.14	458.50	125.12	0.05	0.04	3.95
Test-neutral	148.72	24.65	472.47	130.18	0.05	0.04	3.84
Test-anger	179.86	27.96	295.33	78.09	0.07	0.04	1.67
Test-happiness	167.97	28.87	286.08	81.60	0.07	0.04	1.99
Test-sadness	145.58	20.34	467.85	119.21	0.05	0.03	4.33

Table 7.4 Emotions and speech parameters for F001E.

	F0-mean	F0-std	SOE-mean	SOE-std	EOE-mean	EOE-std	Avg-duration
Train-neutral	205.72	37.29	183.67	60.18	0.05	0.05	2.05
Test-neutral	202.42	36.74	195.44	61.09	0.05	0.05	1.98
Test-anger	223.86	48.75	134.27	48.09	0.04	0.04	1.27
Test-happiness	268.65	56.11	98.19	38.37	0.05	0.07	1.90
Test-sadness	202.79	37.48	155.03	59.38	0.04	0.05	2.22

Table 7.5 KL-divergence distance between neutral GMM and emotional GMMs of few speakers.

	Neutral	Anger	Happiness	Sadness
Speaker 1	0.00	1.02	1.24	1.48
Speaker 2	0.00	2.89	1.21	1.09
Speaker 3	0.00	2.48	1.12	2.46
Speaker 4	0.00	1.48	1.12	1.64

between neutral GMM and emotional GMMs of a few speakers (five speakers) are computed, and the results are reported in [Table 7.5](#).

From the results in [Table 7.5](#), it can be seen that the KL-divergence between neutral GMM and neutral GMM is 0, whereas for the other GMMs it is greater than 1. The lower KL-divergence distance suggests that the feature distribution is more similar, whereas the higher value of KL-divergence distance reflects the large variation between the feature distributions. In general, different speakers will emote differently; it can be observed from [Table 7.5](#) that speaker 3 emoted better than other speakers.

7.5 Proposed strategies for SR in emotional conditions

As mentioned in the previous sections, the major reason for the performance degradation of the SR system is a mismatch between training and testing emotion. In this study, a series of strategies explore the performance improvement of the SR system in emotional conditions.

- Build the UBM with emotional data and adapt the speaker models with the corresponding emotion.
- Build the UBM with emotional data and adapt the speaker model with fused emotional data of the speaker.
- Build an emotion recognition system and choose the corresponding matched emotional speaker models.

7.5.1 SR system with emotional UBM

The experimental setup in this strategy is similar to that of the baseline system described in [Section 7.3](#), except for the selection of data for UBM. In this study, the UBM is built on all emotional utterances (170 from each emotion for each speaker), and the speaker models are adapted with neutral data (40 utterances) of each speaker. The testing setup is exactly same as that of baseline system, that is, testing is done on all emotional utterances (10 from each emotion). The same setup is repeated for all emotions, and the results are reported in [Table 7.6](#).

Table 7.6 Performance (%) of SR system in emotional conditions using MFCC and UBM (contains emotional data).

Accuracy (%)		Emotion type of test utterances			
		Neutral	Anger	Happiness	Sadness
Emotion type of speaker models	Neutral	99.30	49.30	56	41
	Anger	51	96.33	58.33	23.33
	Happiness	59	52	95.33	48
	Sadness	44.33	22.66	39.66	97.66

From the results in **Table 7.6**, it can be seen that there is not much improvement in the present strategy compared to the baseline system. In this case, we can also observe that the performance of the matched condition (see the diagonal entries in **Table 7.6**) is superior to that of the mismatched condition. In order to overcome this problem, in the next strategy, the speaker models are adapted with emotional data.

7.5.2 SR system with emotional data

In this section, the UBM is built on emotional data, which is similar to that of the previous study. However, the speaker models are adapted with all emotional data of each speaker (160 utterances) rather than individual emotional data. The testing is done in a similar way to that of the previous one and the results are reported in row 3 of **Table 7.7**.

For comparing the different strategies studied in this study, the baseline result of the neutral case is reported in row 1 of **Table 7.7**, and the neutral case result of the previous study is reported in row 2. Here it can be observed that, if we use emotional utterances in training, there is a significant improvement in the performance of an emotional SR system. However, this result is inferior to that of the matched condition results of the baseline system reported in the last row of **Table 7.7** (isolated row). The best results for the SR in emotional conditions are obtained when the tested utterance is tested on the emotional state of it, that is, in a matched condition. To achieve it, a

Table 7.7 Performance (%) of SR system in emotional conditions using different strategies.

Accuracy (%)		Emotion type of test utterances			
UBM	Train data	Neutral	Anger	Happiness	Sadness
Neutral	Neutral	99	48.33	54.30	39.66
Emotional	Neutral	99.30	49.3	56	41
Emotional	Emotional	96.3	92	89.66	92.66
Matched condition		99	96.33	95	97.33

two-stage emotional SR system is proposed; first, the emotional state of the speaker is detected using an emotion recognition system, and then the utterance is validated using the speaker models of detected emotion. This is explained in the next section.

7.5.3 Selection of speaker model based on emotion recognition

As shown in Fig. 7.2, the emotional state of a testing speaker is detected using the emotion recognition system and thereby the speaker is tested on the corresponding speaker models of a particular emotion. Here, there is a need for a highly accurate emotion recognition system, as it plays a crucial role in selecting speaker models.

Here the emotional recognition system is build using the GMM-UBM approach. It is similar to that of the baseline SR system reported in Section 7.3. Here the classes are emotion, so the training utterances are taken from each speaker and of a particular emotion. For training each emotion model, we have considered 4200($30 * 140$) utterances for each emotion and testing 900($30 * 30$) utterances are considered. The UBM is built on all 16, 800($4 * 4200$) utterances. MFCCs are considered for the baseline emotional recognition system. The performance of the emotional recognition system and the performance of the two-stage emotional SR system are reported in Table 7.8.

From the results in Table 7.8, it can be seen that the performance of the two-stage emotional SR system is inferior to that of the matched emotion system performance. The main reason for this result is the emotion recognition performance is only 66%. For ideal results, there needs to be a highly accurate emotion recognition system. An improved emotion recognition system is developed using the parameters SoE, EoE, and F0 appended with MFCC, and the results are reported in Table 7.9.

From the results in Table 7.9, it can be seen that the proposed two-stage system resulted in a superior performance to that of the previous two-stage approach. The performance of emotional SR can be further improved with a high accurate emotion recognition system.

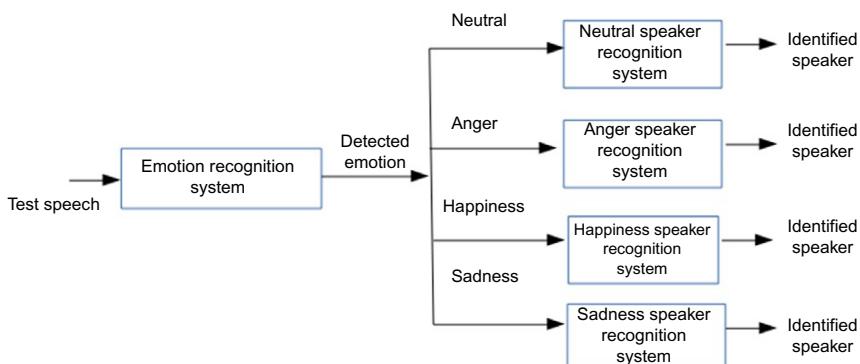


FIG. 7.2

Block diagram of two stage emotional SR system.

Table 7.8 Performance % of emotion recognition system and SR system in emotional conditions using two-stage approach.

Emotion recognition based on MFCC	66
Speaker recognition based on detected emotion	
Neutral	84
Anger	90.33
Happiness	81.66
Sadness	91.66

Table 7.9 Performance % of improved emotion recognition system and SR system in emotional conditions using two-stage approach.

Emotion recognition based on MFCC appended with SoE, EoE, and F0	82
Speaker recognition based on detected emotion	
Neutral	92.66
Anger	94.33
Happiness	89.33
Sadness	95.66

7.6 Summary and conclusions

In this chapter, we studied SR in emotional conditions. This study can be used to enhance speech recognition performance in telecommunications. One of the major advantages of this study can be observed in call centers; with emotionally intelligent automated systems, the service of call centers will be enhanced to a greater extent. This application is very useful in building smart cities. The baseline SR system is developed using the GMM-UBM framework on MFCC features. Performance of the baseline system is significantly reduced in mismatched conditions, that is, the tested speaker emotional state is different from that of a trained emotion of speaker model. The reasons for the performance degradation are assessed through different parameters like F0, SoE, EoE, and duration. The conclusions obtained from these parameters are compared with the vocal feature changes in different emotions. In addition to this, the deviation in feature distribution of emotional utterances of the speaker is studied with Kullback-Leibler divergence distances. From these distances, it is observed that different speakers will emote in different ways. To improve the performance of SR in emotional conditions, different strategies are followed in the data selection of UBM and trained models. When the UBM is built using different emotions, and the speaker models are adapted with emotional data, the performance of SR has improved to a greater extent but is behind the matched condition performance. In order to achieve the performance of a matched condition, a two-stage emotional SR system is proposed; first, the emotional state of the speaker is detected using an emotion recognition system, and then the utterance is validated using the

speaker models of recognized emotion. The MFCC-based emotion recognition system is developed with an accuracy of 66%. The proposed two-stage performance with the developed emotion recognition system is some way behind the matched condition. An improved emotion recognition system is developed using the parameters SoE, EoE, and F0 along with MFCC. This system resulted in a performance comparable with that of matched conditions.

In this work, SR in emotional conditions is studied in the following aspects: (i) the reasons for the performance degradation are analyzed with different parameters which are voice feature correlates; (ii) importance of emotional data in training is explored with different strategies; and (iii) a two-stage approach is proposed for SR in emotional conditions.

Acknowledgments

The authors thank Mr. Govind Divu, and Ms. Pravena Duplex of the Center for Computational Engineering and Networking (CEN), Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Amrita University, Coimbatore, India for providing the Amritha emotional database. The first author would like to thank the Department of Electronics and Information Technology, Ministry of Communication and IT, Government of India for granting the Ph.D. Fellowship under the Visvesvaraya Ph.D. Scheme.

References

- [1] B.S. Atal, Automatic recognition of speakers from their voices, Proc. IEEE 64 (4) (1976) 460–475.
- [2] D. Oshaughnessy, Speaker recognition, IEEE ASSP Mag. 3 (1986) 4–17.
- [3] D. Reynolds, An overview of automatic speaker recognition, in: Proceedings of ICASSP, 2002, pp. 4072–4075.
- [4] D.A. Reynolds, R.C. Rose, Robust text-independent speaker identification using Gaussian mixture speaker models, IEEE Trans. Speech Audio Process. 3 (1) (1995) 72–83.
- [5] D. O'shaughnessy, *Speech Communication: Human and Machine*, Universities Press, USA, 1987.
- [6] J. Makhoul, Linear prediction: a tutorial review, Proc. IEEE 63 (4) (1975) 561–580.
- [7] B.S. Atal, Automatic speaker recognition based on pitch contours, J. Acoust. Soc. Am. 52 (6B) (1972) 1687–1697.
- [8] S.R.M. Prasanna, C.S. Gupta, B. Yegnanarayana, Extraction of speaker-specific excitation information from linear prediction residual of speech, Speech Commun. 48 (10) (2006) 1243–1261.
- [9] N. Dehak, P.J. Kenny, R. Dehak, P. Dumouchel, P. Ouellet, Front-end factor analysis for speaker verification, IEEE Trans. Audio Speech Lang. Process. 19 (4) (2011) 788–798.
- [10] D.A. Reynolds, T.F. Quatieri, R.B. Dunn, Speaker verification using adapted Gaussian mixture models, Digital Signal Process. 10 (1) (2000) 19–41.
- [11] W.M. Campbell, D.E. Sturim, D.A. Reynolds, Support vector machines using GMM supervectors for speaker verification, Signal Process. Lett. IEEE 13 (5) (2006) 308–311.

- [12] G. Hinton, L. Deng, D. Yu, G.E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups, *IEEE Signal Process. Mag.* 29 (6) (2012) 82–97.
- [13] A. Salman, K. Chen, Exploring speaker-specific characteristics with deep learning, in: *Proceedings of IJCNN*, IEEE, 2011, pp. 103–110.
- [14] I. Lopez-Moreno, J. Gonzalez-Dominguez, O. Plchot, D. Martinez, J. Gonzalez-Rodriguez, P. Moreno, Automatic language identification using deep neural networks, in: *Proceedings of ICASSP*, 2014, pp. 5337–5341.
- [15] F. Richardson, D. Reynolds, N. Dehak, Deep neural network approaches to speaker and language recognition, *IEEE Signal Process. Lett.* 22 (10) (2015) 1671–1675.
- [16] K.V. Mounika, S. Achanta, H.R. Lakshmi, S.V. Gangashetty, A.K. Vuppala, An investigation of deep neural network architectures for language recognition in Indian languages, in: *Proceedings of INTERSPEECH*, 2016, pp. 2930–2933.
- [17] H.R. Lakhsni, S. Achanta, P. Bhavya, S.V. Gangashetty, An investigation of end-to-end speaker recognition using deep neural networks, *Int. J. Eng. Res. Electron. Commun. Eng.* 3 (1) (2016) 42–47.
- [18] M. Wegmuller, J.P. von der Weid, P. Oberson, N. Gisin, Study on speaker verification on emotional speech., in: *Proceedings of INTERSPEECH*, 2006.
- [19] J.H.L. Hansen, T. Hasan, Speaker recognition by machines and humans: a tutorial review, *IEEE Signal Process. Mag.* 32 (6) (2015) 74–99.
- [20] I. Shahin, Using emotions to identify speakers, in: *The 5th International Workshop on Signal Processing and Its Applications (WoSPA 2008)*, 2008.
- [21] I. Shahin, Speaker identification in emotional environments, *Iran. J. Electr. Comput. Eng.* 8 (1) (2009) 41–46.
- [22] S.G. Koolagudi, K. Sharma, K.S. Rao, Speaker recognition in emotional environment, in: *Eco-Friendly Computing and Communication Systems*, Springer, 2012, pp. 117–124.
- [23] M.V. Ghiurcau, C. Rusu, J. Astola, Speaker recognition in an emotional environment, in: *Proceedings of the Signal Processing and Applied Mathematics for Electronics and Communications*, 2011, pp. 81–84.
- [24] D. Li, Y. Yang, Z. Wu, T. Wu, Emotion-state conversion for speaker recognition, in: *Affective Computing and Intelligent Interaction*, Springer, 2005, pp. 403–410.
- [25] S.R. Krothapalli, J. Yadav, S. Sarkar, S.G. Koolagudi, A.K. Vuppala, Neural network based feature transformation for emotion independent speaker identification, *Int. J. Speech Technol.* 15 (3) (2012) 335–349.
- [26] T. Wu, Y. Yang, Z. Wu, Improving speaker recognition by training on emotion-added models, in: *Affective Computing and Intelligent Interaction*, Springer, 2005, pp. 382–389.
- [27] K.R. Scherer, T. Johnstone, G. Klasmeyer, T. Bänziger, Can automatic speaker verification be improved by training the algorithms on emotional speech? in: *Proceedings of INTERSPEECH*, 2000, pp. 807–810.
- [28] K.N.R.K. Raju Alluri, V.V. Vidyadhara Raju, S.V. Gangashetty, A.K. Vuppala, Analysis of source and system features for speaker recognition in emotional conditions, in: *2016 IEEE Region 10 Conference (TENCON)*, IEEE, 2016, pp. 2847–2850.
- [29] K.N.R.K. Raju Alluri, S. Achanta, R. Prasath, S.V. Gangashetty, A.K. Vuppala, A study on text-independent speaker recognition systems in emotional conditions using different pattern recognition models, in: *International Conference on Mining Intelligence and Knowledge Exploration*, Springer, 2016, pp. 66–73.

- [30] F. Burkhardt, A. Paeschke, M. Rolfs, W.F. Sendlmeier, B. Weiss, A database of German emotional speech, in: Proceedings of INTERSPEECH, 5, 2005, pp. 1517–1520. vol.
- [31] S.G. Koolagudi, R.S. Krothapalli, Two stage emotion recognition based on speaking rate, *Int. J. Speech Technol.* 14 (1) (2011) 35–48.
- [32] S.G. Koolagudi, S. Maity, V.A. Kumar, S. Chakrabarti, K.S. Rao, IITKGP-SESC: speech database for emotion analysis, in: *Contemporary Computing*, Springer, 2009, pp. 485–492.
- [33] D. Pravena, D. Govind, Development of simulated emotion speech database for excitation source analysis, *Int. J. Speech Technol.* 20 (2) (2017) 327–338.
- [34] I.R. Murray, J.L. Arnott, Toward the simulation of emotion in synthetic speech: a review of the literature on human vocal emotion, *J. Acoust. Soc. Am.* 93 (2) (1993) 1097–1108.
- [35] R. Cowie, E. Douglas-Cowie, N. Tsapatsoulis, G. Votsis, S. Kollias, W. Fellenz, J.G. Taylor, Emotion recognition in human-computer interaction, *IEEE Signal Process. Mag.* 18 (1) (2001) 32–80.
- [36] P. Gangamohan, S.R. Kadiri, B. Yegnanarayana, Analysis of emotional speech at sub-segmental level, in: Proceedings of INTERSPEECH, 2013, pp. 1916–1920.
- [37] P. Gangamohan, S.R. Kadiri, S.V. Gangashetty, B. Yegnanarayana, Excitation source features for discrimination of anger and happy emotions, in: Proceedings of INTERSPEECH, 2014, pp. 1253–1257.
- [38] S.R. Kadiri, P. Gangamohan, S.V. Gangashetty, B. Yegnanarayana, Analysis of excitation source features of speech for emotion recognition, in: Proceedings of INTERSPEECH2015, pp. 1324–1328.
- [39] K.S.R. Murty, B. Yegnanarayana, Epoch extraction from speech signals, *IEEE Trans. Audio Speech Lang. Process.* 16 (8) (2008) 1602–1613.
- [40] B. Yegnanarayana, K.S.R. Murty, Event-based instantaneous fundamental frequency estimation from speech signals, *IEEE Trans. Audio Speech Lang. Process.* 17 (4) (2009) 614–624.

Further reading

- [41] T. Kinnunen, H. Li, An overview of text-independent speaker recognition: from features to supervectors, *Speech Commun.* 52 (1) (2010) 12–40.

PART

Ecological
monitoring

3

This page intentionally left blank

Ubiquitous computing and biodiversity monitoring

8

Todor D. Ganchev

Department of Computer Science and Engineering, Technical University of Varna, Varna, Bulgaria

... It is a study which will surely lead them to an increased appreciation of the beauty and the harmony of Nature, and to a fuller comprehension of the complex relations and mutual interdependence, which link together every animal and vegetable form, with the ever-changing Earth which supports them, into one grand organic whole.

Alfred Russel Wallace [1]

Gaia, the living Earth, our Home!—She provides the habitats and resources needed for the emergence, existence, and thrive of life. Since the early days of humanity, our species has been intuitively aware of this strong bond with Earth and accordingly established various worship rituals to pay tribute to Earth and prayers for abundance in forthcoming seasons.

In modern times, science provides solid evidence and an improved appreciation of the direct dependence between the biodiversity richness and human health and well-being. An analysis of various scenarios for future development of the Earth ecosystems, depending on different models of action which humanity may follow, have been performed [2]. The conclusions, derived by the interdisciplinary board of experts, submit that the decline of biodiversity undoubtedly will translate to loss of ecosystem services and this will have well pronounced consequences to our material well-being, social relations, health, security, and freedom of choice. This improved understanding about the consequences of particular human activities has led to a wider public and governmental support to habitat and biodiversity preservation initiatives [3].^{a,b} Moreover, nowadays it is well understood that the efforts invested in biodiversity preservation, conservation planning, and environmental

^aThe Intergovernmental Science-Policy Platform on Biodiversity and Ecosystem Services (IPBES), an intergovernmental body that implements assessment of the state of biodiversity in response to requests from decision-makers: <https://www.ipbes.net>.

^bThe Strategic Plan for Biodiversity 2011–20 provides an overarching framework on biodiversity for the entire United Nations system and all other partners engaged in biodiversity management and policy development. cf. Aichi Biodiversity Targets, Strategic Goals A–E: <https://www.cbd.int/sp/targets>.

impact assessment bring positive effects to all human economic activities and contribute to improving the quality of life. Finally yet importantly, it is our inherent responsibility to take all actions required for the preservation of Earth ecosystems for future generations of people.

In this regard, the intensification of efforts invested in biodiversity monitoring and habitat preservation is profoundly dependent on the (i) availability of rapid and precise biodiversity assessment survey methods, (ii) feasibility of continuous long-term monitoring of certain biodiversity indicators [4], and (iii) scalability of biodiversity monitoring efforts [5]. Therefore, adequate technological support would be crucial for improved biodiversity monitoring and attaining sustainable results of habitat preservation efforts.

Recent advances in ubiquitous computing, computational bioacoustics and ecoacoustics, remote sensing, and related research and technology development initiatives have the potential to support well the implementation of habitat preservation projects and provide tools that further improve the spatial resolution and the temporal coverage of biodiversity monitoring efforts. This chapter provides a brief overview of the current and emerging technologies that are fundamental for the accomplishment of scalable biodiversity monitoring, pest control, disease transmission, and environment monitoring and habitat preservation actions. A common requirement of these tasks is the need to collect, store, process, and interpret vast amounts of data originating from sources spread over large areas for prolonged periods. This requires on-the-spot data storage and processing, reliable networking and communication infrastructure, and intelligent data analysis and interpretation methods that can resolve contradictions and uncertainty in data. From this perspective, we perceive ubiquitous computing as one of the essential instruments in support of such applications.

The remainder of this chapter is divided into six sections. In the following, we summarize recent methods and technological advances in support of marine and terrestrial biodiversity monitoring ([Sections 8.1 and 8.2](#)), pest control ([Section 8.3](#)), healthcare and disease transmission control ([Section 8.4](#)), and urban ecosystems monitoring applications ([Section 8.5](#)), and eventually provide a brief discussion, outlook of future trends and implications ([Section 8.6](#)). In a brief concluding remark, we comment on some risks related to the potential use of such technology in attempts made to sustain an ecosystem balance (at different balance points).

8.1 Marine biodiversity monitoring

Due to high public sensitivity to the ethical and ecological dimensions of commercial whaling and dolphin hunting, the exploitation of marine resources remained in the spotlight of debate for the majority of the 20th century. Numerous cases of Cetacean mass-stranding events and whale and dolphin deaths have been linked to military wargames, sonar equipment tests, offshore oil-search, specific fishing practices, and other human activities. Growing public concerns about the survival of large

marine mammals prompted the allocation of resources in support of Cetacean population assessment and protection. International agreements for bans on commercial whaling were developed, and although not consistently respected, eventually helped to improve the estimated conservation status of many Cetacean species.

The establishment of an international legal framework, known as the Law of the Sea Convention,^c and the never-ending struggle of developed countries to maintain their political and economic interests, eventually contributed to the advance of worldwide regulations on the exploitation of marine resources. Gradually, the average citizen of the planet became aware of the role of marine environments and their significance to the fluid and dynamic balance in the Earth's ecosystems, which support contemporary lifeforms.

The solid public support empowered the establishment of marine monitoring networks with coverage and functionality allowing continuous monitoring along coastal areas. These efforts find consistent support by the activities of NOAA^d in the United States, EEA^e and EC Environment^{f,g} in the EU, JNCC^h in the United Kingdom, AIMSⁱ in Australia, NIWA^j in New Zealand, the Ministry of EFCH^k of India, and other institutions, which funded or implemented marine biodiversity monitoring projects. Many of these environment monitoring and protection programs created networks^l of surface and underwater data collection stations deployed in specific protected areas and sensitive habitats or economically important coastal areas, or made use of satellite-supported monitoring and tracking with tiny GPS devices attached to individual animals. Often, such monitoring networks make use of distributed data acquisition and recording, and may incorporate data transmission equipment, which incorporates intelligent data sensing, processing, and communication functionalities. However, currently the commonly used marine recording units are still bulky and expensive, which is mainly due to the requirements for waterproof sealing of the electronic equipment and for extended autonomy of operation. Still, the recent advances in ubiquitous computing technology are expected to facilitate the scalability and coverage of marine monitoring networks.

^cThe United Nations Convention on the Law of the Sea: A historical perspective: https://www.un.org/Depts/los/convention_agreements/convention_historical_perspective.htm.

^dNational Oceanic and Atmospheric Administration, U.S. Department of Commerce: <https://www.noaa.gov>.

^eEuropean Environment Agency: <https://www.eea.europa.eu/themes/biodiversity>.

^fEC Environment unit: http://ec.europa.eu/environment/marine/eu-coast-and-marine-policy/marine-strategy-framework-directive/index_en.htm.

^gEC Maritime affairs and fisheries: https://ec.europa.eu/info/topics/maritime-affairs-and-fisheries_en.

^hJoint Nature Conservation Committee, Marine Biodiversity Monitoring and Surveillance: <http://jncc.defra.gov.uk/page-3356>.

ⁱAustralian Institute of Marine Science: <https://www.aims.gov.au>.

^jNational Institute of Water and Atmospheric Research: <https://www.niwa.co.nz/our-science/coasts-and-oceans>.

^kMinistry of Environment, Forest and Climate Change, India: <http://envfor.nic.in/content/vulture>.

^lDEVOTES Catalogue of Monitoring Networks, 3rd edition: <http://www.devotes-project.eu/devotes-release-new-version-catalogue-monitoring-networks>.

Thanks to the large amounts of data collected through these marine environment monitoring networks, and the improved understanding of the importance of human-related pressures and the global weather changes on the marine biodiversity, the United States, the EU, Japan, and other countries started to take measures in an attempt to reduce the levels of pollution and to ensure sustainable exploitation of marine resources. A critical overview of monitoring networks in the EU seas is available in Patricio et al. [6]. Despite these efforts, to this end only around 5% of the world ocean is studied, and less than 2% is protected.^m Large-scale initiatives [7] aimed at mapping the ocean's seafloor,ⁿ though not directly linked to biodiversity, hold great potential to facilitate prospective marine monitoring and conservation efforts in accordance with the vision postulated in the United Nations vision for biodiversity monitoring and protection [3, 8].

8.2 Terrestrial biodiversity monitoring

Terra firma accounts for a small part of the Earth's surface, which combined with the diversity of relief and climatic factors well explains the relatively smaller size of terrestrial habitats in comparison to the large aquatic areas. This smaller size, and the availability of transportation and communication infrastructure with wide coverage, translates to inexpensive logistics and easier penetration of modern human activities in most terrestrial habitats. All this sets the preconditions for the great exposure and susceptibility of terrestrial habitats to human-induced pressures.

The intensification of land use, industrial pollution, and other negative effects of human activities adds to the effects of climate change and causes extreme pressure on many terrestrial species. With few exceptions, human economic activities are not sustainable or eco-friendly, cause damage to habitats and fading out of wild animal populations, and accelerate biodiversity's degradation and loss. The escalation of this problem brought the necessity of prompt allocation of resources to biodiversity monitoring and conservation projects. In that connection, biodiversity monitoring, assessment, and conservation initiatives are perceived as crucial, and thus receive high public appraisal. Public pressure prompted the establishment of international agencies and funding bodies in support of local, regional, and global biodiversity-related research and conservation actions. Among these in Europe are EC LIFE +,^o Biodiversa,^p EU Structural and Investment Funds,^q and INTERREG Europe^r; in the USA the primary support comes through the National Science Foundation^s (NSF), and in Brazil through the National Council for Scientific and Technological

^mAtlas of Marine Protected Areas: <https://earther.gizmodo.com/it-turns-out-weve-only-protected-2-percent-of-the-ocean-1823806943>.

ⁿGEBCO Seabed2030 Project: <https://seabed2030.gebco.net>.

^oThe EC LIFE+ program: <http://ec.europa.eu/environment/life/funding/lifeplus.htm>.

^pThe Biodiversa network: <http://www.biodiversa.org>.

^qThe EC Structural and Investment Funds: http://ec.europa.eu/regional_policy/en/funding.

^rINTERREG Europe: <http://www.interregeurope.eu>.

^sThe National Science Foundation (NSF) of United States: <http://www.nsf.gov>.

Development^t (CNPq), CAPES Foundation,^u etc. These and other funding sources across the world provide the financial instruments in support of large-scale international and intercontinental initiatives^{v,w,x} that coordinate diverse interdisciplinary activities aiming at slowing down the global trend of rapid biodiversity loss.

The technological advances in support of biodiversity preservation are exemplified through the achievements of some large-scale research and technology development projects, which took advantage of interdisciplinary international collaboration and achieved important social impact. These made a confident use of some early elements of the emerging ubiquitous computing technology in order to address a wide range of challenging real-world problems. Here, these projects are interesting because they made the difference with respect to previous research, and tackled some of the difficulties related to real-world operational environment. In the following, we summarize the achievements of a few projects that brought technological innovations and proof-of-concept demonstrations. A common feature of these projects is that they handled large amounts of recordings collected in real-world conditions, and could have done much better if at that time off-the-shelf ubiquitous computing technology had been readily available.

In brief, the application scenarios covered in these projects can be summarized to acoustic species recognition for the purpose of biodiversity assessment and

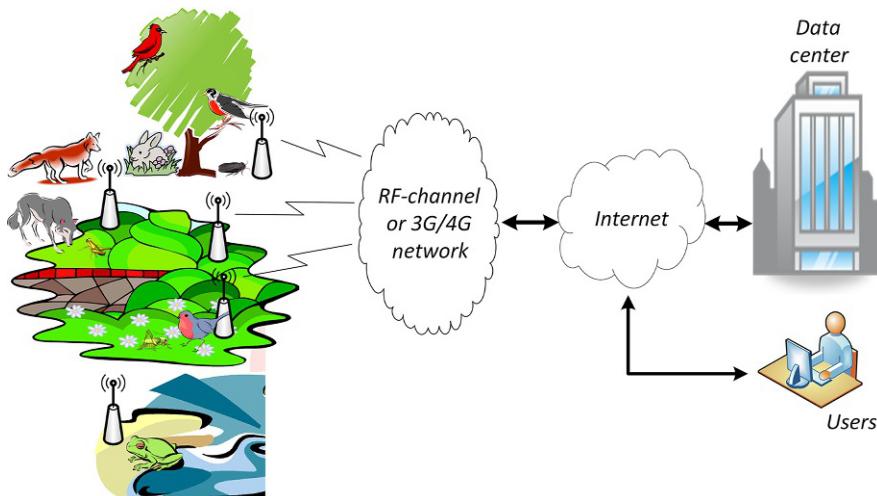


FIG. 8.1

Overall concept of the soundscape acquisition, transmission, and analysis.

^tThe National Council for Scientific and Technological Development (CNPq): <http://cnpq.br>.

^uThe CAPES Foundation, Ministry of Education, Brazil: <http://www.capes.gov.br>.

^vThe LifeWatch project: <https://www.lifewatch.eu>.

^wThe EU-Brazil OpenBio project: <http://www.eubrazilopenbio.eu>.

^xThe COOPEUS: Connecting Research Infrastructures project: <http://www.coopeus.eu>.

biodiversity monitoring studies, such as those implemented in the ARBIMON^y project in the United States and the AmiBio^z project in the EU. Both projects designed and implemented scalable automated systems that are capable of collecting environmental sounds, then transmitting data over large distance to a base station, where the actual data analysis takes place (cf. Fig. 8.1). However, the implementation of the technological framework differed and thus the capacity for data collection and distributed processing. In the following, we briefly mention some facts about these two indicative projects.

8.2.1 The ARBIMON acoustics project

The ARBIMON Acoustics project developed and deployed biodiversity monitoring infrastructure for permanent acoustic monitoring of various terrestrial habitats. This infrastructure consists of multiple remote stations that are capable to collect and transmit soundscape recordings, a base station for data aggregation, compression, and upload, and a server facility for extensive audio processing and permanent data storage (cf. Fig. 8.2). The main purpose of each remote station is periodically to capture soundscapes of short duration and transmit the recordings to the base station.

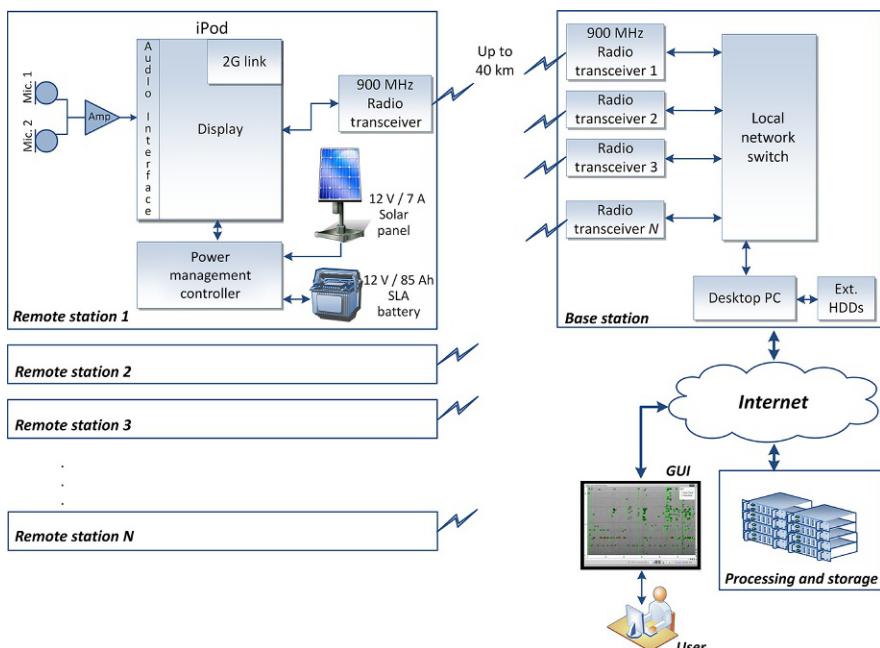


FIG. 8.2

The ARBIMON acoustics project: audio acquisition, transmission, storage, and processing.

^yThe ARBIMON Acoustics project: <https://arbimon.sieve-analytics.com>.

^zThe AmiBio project: <http://www.amibio-project.eu>.

A key point here is that the recordings are not stored locally but instead are transmitted wirelessly via a radio communication link to the base station. The selected communication channel (900 MHz RF-link), radio power, and antennae permitted transmission on a distance between 1.2 and 40 km, depending on the weather conditions and density of vegetation in the direction of radio communication link. The overall design of this network provides a certain degree of scalability and makes possible the simultaneous acoustic monitoring of sound emitting species at various locations, in their natural habitats, remotely, unobtrusively, and in near real-time.

The entire infrastructure was integrated based on commercial off-the-shelf components, which contributed to minimizing the development time and reduced the overall project cost. The remote stations that form the backbone of the audio collection network are based on a commercial iPod device [9]. The iPod was equipped with a stereo microphone and a custom-built application, which sets the iPod to serve as a reprogrammable audio recorder. A microphone preamplifier was used in order to improve sensitivity to weak sounds and thus increase the perceptive range of the remote station.

Each remote station is equipped with an autonomous power source—a car battery charged by a solar panel through a voltage controller. Such a power source facilitates extended autonomy of operation, which lowers the maintenance cost of the remote stations. All components of the remote station, except the microphones, solar panel, and radio antenna, are packed in a waterproof case for protection against humidity and mechanical damage. The limitations due to the specific design of the remote station are related to the range of feasible values for audio sampling frequency, the number of audio channels, and the relatively narrow bandwidth of the radio communication channel. In stereo mode and high sampling frequency, which are considered in the particular setup, the communication link permits only a few minutes of audio to be recorded and transmitted each hour. The commonly referred soundscape acquisition schedule implements the capturing of 1 min of audio once every 10 min, which totals 144 one-minute recordings per 24 h [9]. However, the software for audio recording that runs on the iPod device allows a flexible data acquisition schedule, depending on the project needs. The selected 1-min length of audio recordings is linked mostly to the limitations associated with the communication channel bandwidth, the energy efficiency of the remote station, and the capacity of the permanent data storage repository at the ARBIMON premises. Here we need to point out that the ARBIMON team made the implicit assumption that sampling 1 min per every 10 provides a fair representation of the acoustic activity near the remote station.

At the base station side, multiple radio receivers are attached to a local area network (LAN) switch, with one receiver^{aa} for each remote station. A desktop PC is also

^{aa}Taking advantage of a certain time-division scheme, one radio receiver could potentially collect data of several remote stations. This is feasible only when each remote station uses the communication channel periodically (e.g., only for a few minutes per hour) and if the communication protocol allows such a functionality. Alternatively, when time-division protocol is not desirable, or when continuous audio monitoring is a valid option, each remote station requires a separate receiver at the side of the base station. Such an approach would not favor scalability—for instance, consider that one base station has to serve as a data concentrator for more than a few dozen remote stations.

connected to the same LAN switch to collect the audio transmitted by many remote stations. The primary purpose of this PC is to detect the arrival of a new audio recording, convert the audio from stereo to mono, compress the audio files, make a local copy on an external hard disk drive (HDD), and upload the compressed data to the ARBIMON Processing and Storage server through a wideband Internet connection. Lossless audio compression (a compression technique that decompresses data back to its original form without any loss) based on the FLAC^{ab} encoding/decoding tools was applied on each audio file in order to reduce the requirements for long-term data storage and for a more economical use of the Internet connection bandwidth when data are sent to the server.

The project established a data storage repository with a relational database and certain tools for automated data processing, annotation, modeling, content mining, search of content on different levels of abstraction, etc. Once audio files are uploaded to the ARBIMON processing and storage server, these are archived in the permanent storage repository. The raw audio data, the audio spectrograms, or their processed versions are made available to registered users through a web-based interface. After a successful login, registered users can select a specific audio recording, by means of filters that allow specifying the particular subproject/location/date/time. In addition, a web-service running on the ARBIMON processing and storage server allows registered users to listen, annotate, and search in preselected dataset, either manually or through certain (semi)automated tools. In such a way, taking advantage of the ARBIMON infrastructure and Internet, biologists and citizens can have remote access to monitor acoustic environments at various locations. The ARBIMON data repository contains all recordings made from 2008 to the present.

Some more sophisticated tools and services for the automated detection of sound events and of regions of interest in the audio spectrogram became publicly available with the second phase of the project, referred to as ARBIMON II. The most significant advances in ARBIMON II are due to the Cloud-based services for data storage and processing, which provide convenient tools and flexibility of data access and analysis through a web-based user-friendly interface. The ARBIMON II platform^{ac} and the Sieve Analytics web-services,^{ad} launched in November 2014, facilitate tasks associated with the visualization of soundscape recordings and their audio spectrograms, annotation/tagging of soundscape recordings for presence/absence of certain species, and development of automated species-specific recognizers. Finally, yet importantly, the ARBIMON II platform permits registered users conveniently to share their projects design, data, audio recording, annotations, models, etc., and thus foster collaborations, and bring benefits to society. It is very important to emphasize that the ARBIMON team made a long-term investment of efforts and resources to establish and support a community of researchers which contributes to the

^{ab}The Free Lossless Audio Codec (FLAC): <https://xiph.org/flac>.

^{ac}The ARBIMON II platform: <https://arbimon.sieve-analytics.com>.

^{ad}The ARBIMON Acoustics web application: <http://www.sieve-analytics.com/#!arbimon/cjg9>.

enrichment of the ARBIMON II platform with data, methods, and tools. All these are of great value and assistance to the efforts toward the implementation of scalable studies in support of global biodiversity preservation [10].

In addition to the Cloud-based platform, the ARBIMON II project also modernized the design of the remote stations for audio acquisition and transmission. The improved design is based on a contemporary smart-phone with 16 GB, touch screen, and 3G and Wi-Fi connectivity. The pair of microphones was replaced with a single microphone, which slightly worsened the spatial sensitivity in the direction backwards to the microphone. In this new design, the radio communication link is with an extended range (reportedly up to 65 km). Furthermore, the radio communication link is now optional, and is considered essential only at locations where Wi-Fi and 3G wireless infrastructure are not readily available. Due to the use of low-power technology, the new remote stations make use of a smaller rechargeable battery (6.4 Ah) and a much smaller solar panel (14 W). The smaller dimensions of the rechargeable battery and other components made it possible for all modules to be placed in a compact waterproof case.^{ae,af} The latter renders the remote station smaller and lighter (weights approximately ~1 kg), and thus easier to deploy and maintain. Without the solar panel, the remote station can be used as a portable device capable of 20 days' autonomy, given that audio capturing schedule is set to recording 1 min every 10 min. The convenient user interface, the increased CPU power, and local storage in the new remote station provide extra flexibility and reduced risks of information loss due to transient connectivity failure of the radio communication channel.

As a final point, we would like to highlight that we credit the ARBIMON project as the first long-term project to promote the use of automated technology for soundscape collection, transmission, and processing, and the longest-lasting among all biodiversity-monitoring projects that made extensive use of automation. All this, together with the free data access, and the opportunities for collaboration it provides, makes it one of the most influential success stories of emerging technology for automated biodiversity monitoring. We are hopeful that the advances brought by ubiquitous computing and Internet of Things technology will help to improve the scope and scalability of biodiversity monitoring networks.

8.2.2 The AMIBIO project

AmiBio was a 4-year project, co-funded by the EC LIFE+ programme,^o which was implemented between February 2010 and July 2013 at the area of Hymettus Mountain in Greece. Among the expected outcomes were: biodiversity assessment and inventorying in the Hymettus Mountain area; estimation of the density of animals in the monitored areas; monitoring and sending out urgent notifications about the presence or absence of rare and threatened species in inaccessible areas as well as

^{ae}Sieve Analytics: <https://www.sieve-analytics.com/buy>.

^{af}Sieve Analytics on Twitter: <https://twitter.com/sieveAnalytics>.

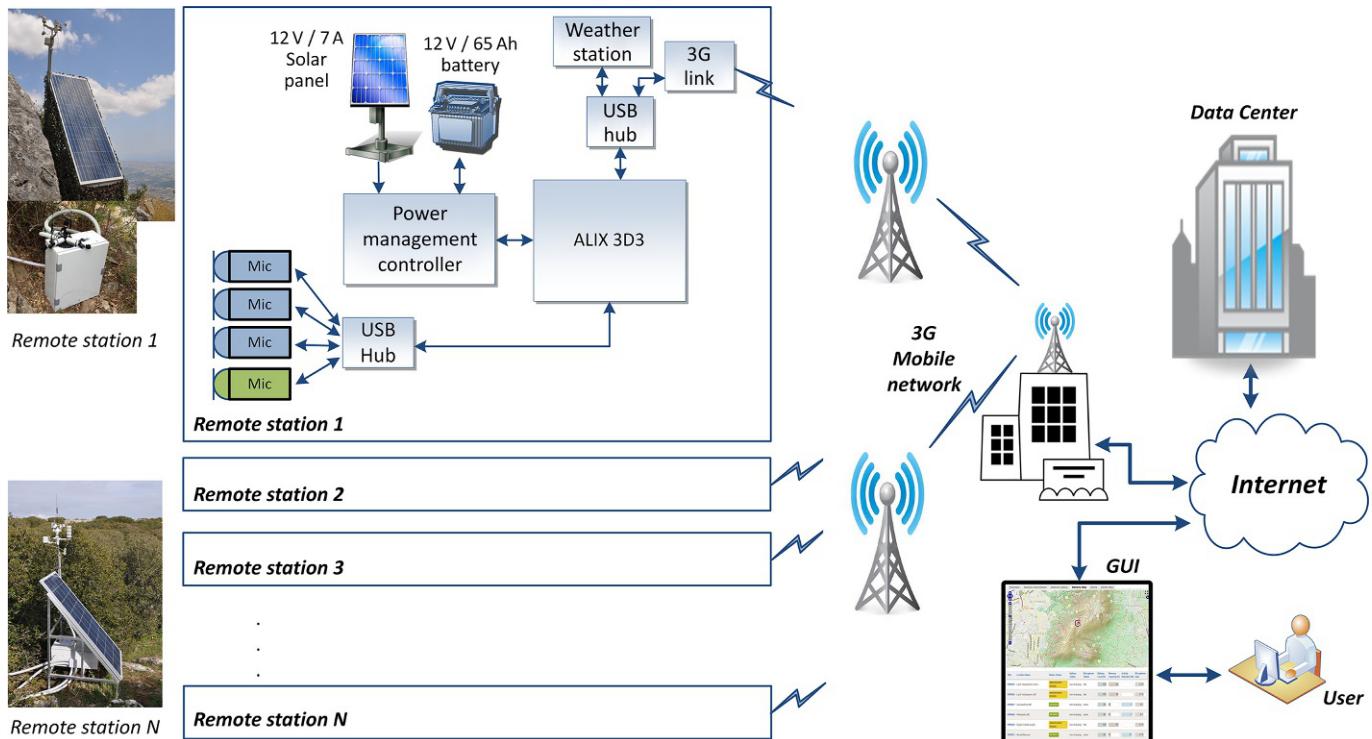
of night-migrating birds; estimation of the health status of certain species based on their vocalizations; monitoring and sending out urgent notifications of specific atypical sound events, such as those related to illegal or potentially hazardous human activities, such as gun shots, tree felling, illegal motorcycle races, etc.; and reporting to responsible authorities about an emergency situation, 24/7 monitoring for danger and crisis events, monitoring for natural calamity, and detection of human-induced disasters.

The accomplishment of these outcomes depended on the deployment of an automated system for acquisition, transmission, storage, detection, analysis, and visualization of acoustic and weather data, so that forest management organizations, NGOs engaged with the protection of the natural resources of the Hymettus Mountain, law enforcement authorities, and firefighters can take prompt action when needed. Therefore, the ultimate goal was to develop an automated system for continuous biodiversity monitoring, which is also capable of real-time detection of potentially dangerous events and of identifying evidence of ongoing illegal activity, such as tree felling, motocross, forest fires, etc. For that purpose, the AmiBio team designed and deployed an automated real-time monitoring network (cf. Fig. 8.3), which consisted of 17 remote stations. Each remote station is equipped with four microphones and a weather station, and some stations may have a fire sensor and illumination sensor. These stations allow users to register, record, compress, and transmit audio and weather data to the remote Data Center, where data streams are stored permanently and analyzed automatically or semiautomatically. The permanent data archival provides opportunities for long-term monitoring of biodiversity trends in the four main habitats at Hymettus.

The remote stations were deployed in the four major types of habitats at the Hymettus Mountain, such as mid- to low-forest canopy, semiopen scrublands, and around wetlands. Four remote stations were deployed in each habitat type, in order to provide data for statistical analysis. One additional remote station was deployed in the botanic garden at Hymettus. In fact, the project concept and technical design permitted a larger number of remote stations to be deployed afterwards (on the go), and therefore AmiBio offered straightforward scalability [11]. Such scalability provides flexibility as the network size can be adjusted to the needs of a particular plot study and deployed in subsequent increments according to the available budget and resources.

The remote stations were integrated taking advantage of off-the-shelf components, hardware modules, and software tools. Each remote station consisted of an audio acquisition unit with Wi-Fi and 3G connectivity, 4GB of local storage capacity, 4 USB microphones, a 65 Ah car battery, a 0.6 m² solar panel,^{ag} and a voltage controller. A number of recording stations were also equipped with a small weather

^{ag}Due to the free public access to the Hymettus Mountain and its statute of a recreational area, there were restrictions imposed on the use of solar panels in certain areas—authorities requested the project equipment to remain less visible to visitors. For that reason, solar panels were mounted only to approximately half of the remote stations, and for the rest the car batteries were replaced for recharge every 2 weeks. In few locations, power supply was available from nearby buildings or by the electricity network available in proximity to the remote stations.

**FIG. 8.3**

The AmiBio project: Audio acquisition, transmission, storage, and processing.

station, which registers the local micro-site conditions, such as wind direction and strength, humidity, temperature, rainfall, illumination, etc. The custom software developed by the AmiBio project along with some open-source signal processing applications were hosted on a small Linux-based single-board PC, which served as an audio acquisition unit, local processing unit, and communication unit.

The main CPU and RAM resources were provided by the commercial single-board computer ALIX 3D3.^{ah} An 802.11a/b/g DualBand miniPCI-extension board was used for implementing the Wi-Fi connectivity. The 3G connectivity was implemented via a 3G USB modem and 3G SIM card provided as is by the telecom operator. The 4 GB data storage card hosted the operating system, application software, and served as a temporary storage for the four audio streams and the compressed data before upload. Since the ALIX 3D3 board has only two USB ports, two four-port USB hubs were used to accommodate the interface to a total of seven USB devices, among which are the four USB microphones, the weather station, the 3G modem, and the illumination sensor. The AmiBio project relied on low-cost electret microphones combined with low-cost ICICLE XLR-to-USB converter^{ai} that supports sampling frequency of 44.1 kHz and resolution of 16 bits per sample.

Three of the microphones were arranged coplanar, placed horizontally and directed at 120 degrees one from another, in order to cover the full panorama of 360 degrees. The fourth microphone was mounted perpendicular to the three coplanar ones, and pointed upward in order to capture sound emissions from birds and bats flying over the remote stations. Some of the remote stations were equipped with an ultrasonic USB microphone,^{aj} capable of registering ultrasonic emissions with frequency bandwidth of 100 kHz and resolution 16 bits per sample. The sampling frequency of 200 kHz was presumed sufficient for capturing ultrasonic emissions of all bat species inhabiting the Hymettus Mountain. The ultrasonic microphone was always positioned to point upward, replacing the vertical microphone in the above-mentioned four-microphone setup. The remote stations with ultrasonic microphone were deployed at locations where bat and insect acoustic activity was expected.

The four audio channels captured by the microphones of each remote station, including the ultrasonic one, were provisionally stored locally in an audio acquisition unit—each audio stream in a separate .WAV file. The .WAV files were then compressed on the fly with the lossless FLAC audio codec.^{ak} The FLAC compression contributed to a reduction of the overall amount of data by more than 50%. The compressed .FLAC files were next transmitted via the 3G wireless network to the data center, located more than 250 km away from the Hymettus Mountain. In order to optimize the network traffic load over the 3G network, the telecom operator redirected through the Internet all traffic associated with the AmiBio remote stations. The

^{ah}The PC Engines website: <https://www.pcengines.ch/alix3d3.htm>.

^{ai}The Blue ICICLE XLR-to-USB converter: <https://www.bluedesigns.com/accessories>.

^{aj}The DODOTRONIC Ultramic200K: <https://www.dodotronic.com>.

^{ak}The Free Lossless Audio Codec (FLAC): <https://xiph.org/flac>.

last also made possible AmiBio data streams to be delivered to any other location with little extra effort.

The data center hosted an FTP server, a file server, and an application server. Each remote station possessed an individual account on the FTP server, as this facilitates the upload of recordings and organization of data, and provides flexibility of telemetry, maintenance, and opportunity for individual reconfiguration of the recording setup. For example, the independent FTP-account allowed each remote station periodically to retrieve its configuration settings and upload the compressed .FLAC files without interference with the operation of the other remote stations. Next, the .FLAC files were automatically moved to the file server, which served as the permanent storage repository. The application server carried out all tasks associated with audio processing and analysis. The GUI provided end-users with functionality and tools helping to retrieve statistics, information about the operation of the monitoring network, alerts, etc. The GUI was implemented as a web-based interface with multiple tabs.

The AmiBio project also developed a database and a convenient interface, which allows recordings of a certain location/date/time to be selected for visualization, annotation, processing, etc. In total, the AmiBio project developed 32 species-specific acoustic recognizers [12].

The most significant advance in the AmiBio project was that it first demonstrated in practice the concept of scalable real-time acoustic monitoring of species in a large area, along with the real-time detection of illegal activity. The deployment of additional remote stations is feasible at any time and does not require change of the overall design or the protocol of data collection, processing, or visualization. Such a scalability of real-time data collection is due to a great extent to the availability of 3G-communication infrastructure and the wideband Internet connection accessible in the region of the Hymettus Mountain. At the time of project implementation, 3G wireless communication infrastructure was available for over 95% of the territory of Europe; however, it might not be readily granted at other places. These achievements of the AmiBio project were praised by the EC LIFE+ program and AmiBio was distinguished as one of *The 4 “Best of the Best” LIFE Nature Projects 2013*.^{al}

Regrettably, the AmiBio project was discontinued in July 2013, shortly after the LIFE+ co-financing ended—the project consortium failed to raise subsequent funding or to secure the sustainability of the AmiBio monitoring system. From this perspective, the discontinuation of efforts for further advancement and elaboration of the AmiBio monitoring system (or at least securing budget for its long-term maintenance and operation) is perceived as a great opportunity lost. Consequently, the consortium failed to benefit from the AmiBio system and use it to carry out valuable research on biodiversity monitoring studies.

^{al}Best LIFE-Nature Projects 2013: <http://ec.europa.eu/environment/life/bestprojects/bestnat2013/index.htm>.

Despite the hardware and software limitations at the time of their implementation, the ARBIMON and AmiBio projects brought forward some important conceptual and technological innovations and facilitated the implementation of ambitious technological solutions and proof-of-concept prototypes that have the potential to transform the manner in which sound emitting species are monitored. Coping with the challenges of real-world operational conditions, these projects developed the overall strategy which could accommodate not only acoustic monitoring but also video cameras and other types of sensors.

8.3 Pest control

The ENTOMATIC project,^{am} entitled “Novel automatic and stand-alone integrated pest management tool for remote count and bioacoustic identification of the Olive Fly (*Bactrocera oleae*) in the field,” was implemented between September 2014 and December 2017 by a consortium including partners from Belgium, France, Germany, Greece, Italy, Portugal, Spain, and Turkey. The project was co-financed under the FP7 SME program BSG-SME-AG of the European Commission.

The ENTOMATIC project provided technological support to the present-day methodology for integrated pest management. The main goal was to develop an automated pest monitoring system, which provides timely alerts about the presence of *B. oleae* adults, and spatial information about their abundance in the areas with olive tree plantations. For that purpose, the project developed a purposely designed *smart trap* for *B. oleae*, data collection, and communication infrastructure, software tools for automated data processing, a geographical information system, a decision support system, and automated alert services, which facilitate the timely treatment of infested areas. It was expected that the introduction of automated technology will contribute to a significant decrease of insecticides use and in the same time will significantly improve the efficiency of control of the *B. oleae* populations, the larvae of which feed on olive fruits.

In an attempt to control the populations of *B. oleae*, olive producers in EU make extensive use of insecticides sprayed many times per year. Recent estimates cited by the ENTOMATIC project reported that the annual spending for pesticides in the eight Mediterranean countries that produce olive oil amounts to 5 billion Euros. However, despite these efforts, olive producers regularly report over 30% loss of production, and olive oil producers report up to 50% financial losses due to decreased quality of production and missed opportunities to deliver extra virgin olive oil—according to EU legislation, extra virgin olive oil quality supposes less than 10% of olive fruit damage. The main reason for these financial losses is that *B. oleae* infestations are not detected in time, and due to lack of coordination among regional and national authorities, some areas with infestation are overlooked while other are over-treated as a preventive measure. The extensive use of insecticides (i) brings health

^{am}The ENTOMATIC project website: <https://www.upf.edu/web/entomatic>.

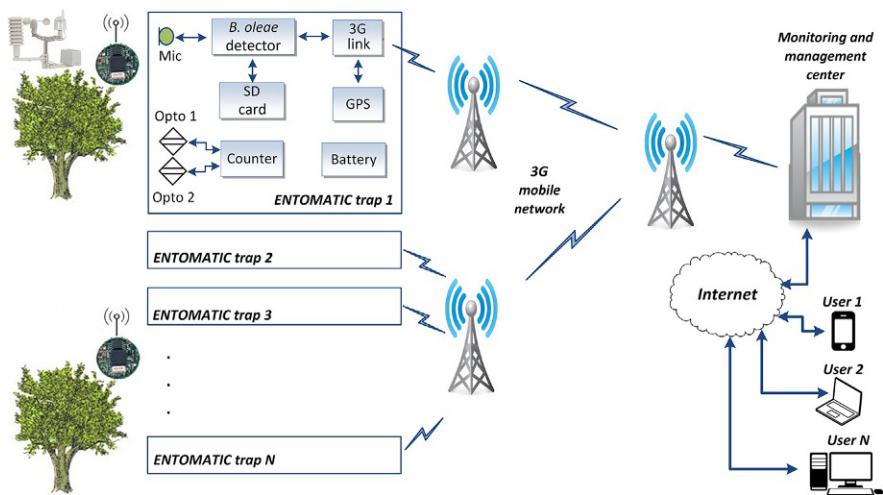


FIG. 8.4

Overall concept of the ENTOMATIC project.

concerns, (ii) has negative effects on domestic bees and wild animal species, and thus (iii) contributes to balance shift in the ecosystem, which results in long-term harm to wildlife and humans. At the same time, the under- and overuse of insecticides brings substantial financial losses to olive producers.

In brief, the ENTOMATIC pest monitoring system (cf. Fig. 8.4) made use of *smart traps* for *B. oleae*, which serve as sensor nodes integrated in a local wireless sensor network (WSN). The data collected by each trap are relayed over the WSN to a gateway, which serves as data concentrator and WSN supervisor. Each gateway is equipped with a small weather station, which logs the local weather conditions in the olive tree plantation. The gateway transmits the information aggregated from the individual *B. oleae* traps and weather data over GSM/GPRS networks to a monitoring and management center (MMC). The MMC implements the data processing and analysis, and with the help of a geographical information system (GIS) and a decision-support system, makes recommendations for pesticide application, dosage, scope of spraying, etc. These recommendations are sent as alerts to individual olive producers subscribed to the automated alert service and to olive producer associations.

The smart traps use baits and lures to attract adult *B. oleae* inside; however, other insect species are also attracted and enter the trap. Once an insect enters the trap, an optical counter registers its presence and an acoustic recognizer detects whether it is *B. oleae* or another species. The automated detection process is based on the specific pattern of *B. oleae* wing sounds during flight [13, 14].

The ENTOMATIC project demonstrated another success story on the application of computational bioacoustics—species-specific automated acoustic detection of insects was embedded in a real-world application for integrated management of pest populations.

8.4 Health and disease transmission

Mosquitoes are recognized as the most dangerous creatures with respect to disease spread and resultant human mortality. In the past decades, mosquito populations control was of great concern mainly due to malaria, yellow fever, and dengue. To this end, various methods for control of the mosquito populations were developed. These carry out strategic actions^{an} aiming at the reduction of mosquito habitats, or make use of specific procedures for spraying with insecticides, implementing measures aiming to impede the mosquito reproduction through the introduction of sterile insects, etc. However, to a great extent the success of these methods depends on the availability of accurate and up-to-date information about the spatial distribution of mosquito populations. In this regard, the REMOSIS project,^{ao} “Remote Mosquito Situation and Identification System,” facilitated the quick availability of data collected by means of smart traps, which are deployed at various locations along the expected paths of mosquito propagation.

The overall concept of the REMOSIS project is illustrated in Fig. 8.5. The data collected by the smart traps are uploaded wirelessly to a Cloud server. Internet of

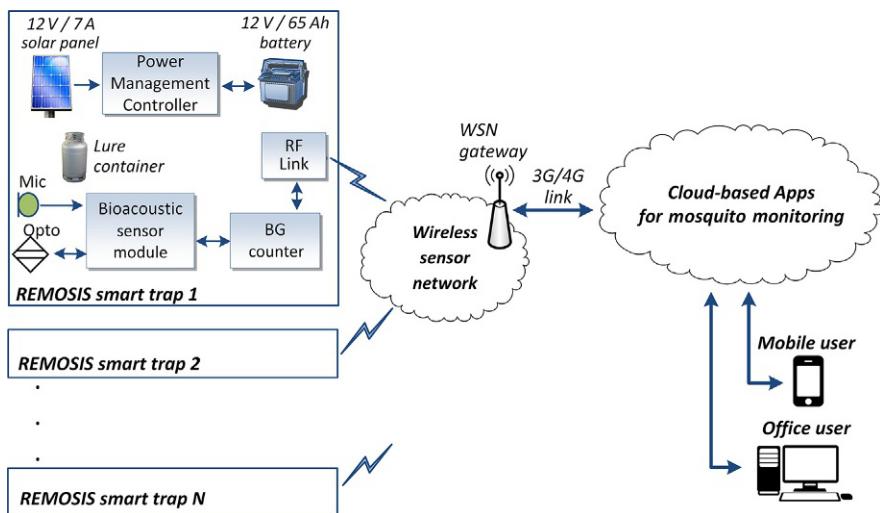


FIG. 8.5

Overall concept of the REMOSIS project.

^{an}The “African Vector Control: New Tools” (AvecNet) project aimed at malaria control. It was implemented during February 2011–December 2016, and co-financed under the EU FP7-HEALTH program: <https://cordis.europa.eu/project/rcn/98328/reporting/en>.

^{ao}The REMOSIS project (February 2016–January 2018) was co-financed by the EC H2020 program under topic FTIPilot-1-2015 “Fast Track to Innovation Pilot”: <https://remosis.bg-counter.com>.

Things (IoT) technology contributes to a wider availability of data collected in different contexts for a more precise risk assessments analysis.

In brief, the REMOSIS project upgraded an existing commercial line of mosquito traps^{ap} with new functionality for remote monitoring and optoacoustic identification of mosquito species [15]. This new smart trap is capable of counting and automatically identifying several dangerous Tiger mosquito species, such as *Aedes albopictus* and *Aedes aegypti*. These and other mosquito species are known to transmit yellow fever, dengue, the Zika virus, and various mosquito-borne pathogens, which pose a significant threat to human health. The project consortium reported on the successful development of a commercially available smart trap station with mosquito-counting module and remote management tools. Furthermore, a prototype technology for automated mosquito species identification and advanced wireless sensor network were evaluated in pre-production series. Thanks to the Cloud, application data are received and displayed with a delay of not more than 15 min after registration.

In summary, the REMOSIS project significantly contributed toward an improved control of mosquito-transmitted diseases and a reduction of the overall monitoring cost. The last is mainly due to the elimination of manual human-expert inspection practices, and the opportunity for focused insecticide use which contributes to reducing the negative effects on the environment. The smart trap and Cloud-based services support future planning and surveillance actions with improved accuracy and promptness. The earlier detection and more precise localization of specific mosquito species provides more opportunities for mosquito population control, which is expected to facilitate the risk reduction actions associated with preventing the outbreaks of mosquito-transmitted infectious diseases.

8.5 Monitoring of urban ecosystems

The human population is growing fast, and more than half of humankind lives in cities. According to estimates from the European Environment Agency, by 2020 almost 80% of the European population will live in big cities, towns, or in urban settlements between the two [16]. This undoubtedly comes at a cost. The large-scale analysis of interactions amid political context, economic growth, land use changes, ecosystem degradation, and economic activities improved public awareness about the heavy impact of urbanized territories on the quality of Earth's ecosystems [2]. The problems associated with the city territory enlargement, the increasing concentration of urban population, and the typical thrifless use of resources in cities, waste management issues, and other negative effects are already well understood.

^{ap}The BG-Counter (<https://www.bg-counter.com>) and BG-Sentinel (<https://www.bg-sentinel.com>) product lines are designed, manufactured, and marketed by BIOGENTS AG: <https://eu.biogents.com>.

Furthermore, the intensification of agricultural land use imposes significant pressure on the populations of nearly every species. Yet numerous animal species discovered ways to adapt and exploit the opportunities offered by urbanized territories and some of them successfully cohabit with humans in typical city environments. In the past decade, a half a dozen projects^{aq} aimed at the improvement of our understanding of the mechanisms of this adaptation have been implemented. These projects recognized the importance of urban ecosystems and in many cases developed strategies for resilient planning and management of large cities, such as New York, Berlin, etc. Furthermore, the URBIS,^{ar} URBES,^{as} GrowGreen,^{at} and other initiatives brought forward to the general public the current well-understood links between biodiversity and ecosystem services and human physical health and well-being.

Although research studies undoubtedly confirmed the importance of green infrastructure, where the design of interlinked green spaces was demonstrated to bring advantages in terms of temperature reduction and lessening the energy pollution (viz., light, noise, and small particles), so far, very little has been done on a global scale. The reason for this is that knowledge of green infrastructure and the interconnecting links is primarily considered in urban development plans in well-developed countries, while the vast number of less-developed countries where city growth is appreciably faster are left out of the equation.

The implementation of biodiversity monitoring in urban environments benefits a lot from the availability of infrastructure and wide-band data communication networks. These allow for taking advantage of the latest technological achievements—smart sensing in the context of IoT and Cloud computing, which is undoubtedly the long-awaited convergence of the ideas and concepts underlying ubiquitous computing.

Certainly, in the long term, ubiquitous computing is the most promising technology that could make feasible the large-scale continuous biodiversity monitoring—either in an urban environment or in the wild. However, at this time we need to emphasize the single most important advantage offered by cities striving for large-scale biodiversity monitoring: the great abundance of volunteers in the context of “Citizen Science.” This provides opportunities for the involvement of citizen volunteers at all stages of data collection, tagging and processing, and the validation and interpretation of results, etc., which is undoubtedly the most valuable resource in support of research and technology development efforts aimed at the creation of automated tools and services for biodiversity monitoring and assessment. Although it may sound surprising, it might turn out that ubiquitous computing will reach its height through the involvement of large masses of volunteer contributors who help

^{aq}List of recent projects co-funded EC LIFE+ Nature and Environment instruments: <http://ec.europa.eu/environment/life/project/Projects/index.cfm?fuseaction=home.getProjects&themeID=97&projectList>.

^{ar}The Urban Biosphere Initiative: <http://urbis.org>.

^{as}The Urban Biodiversity and Ecosystem Services (URBES) project: <https://www.biodiversa.org/121>.

^{at}The GrowGreen project—embedding Nature-based Solutions in cities: <http://growgreenproject.eu>.

in the culling of crucial materials and in the production of resources (audio/video recordings, tags, metadata, contextual information, domain knowledge, etc.), which are the necessary prerequisites for technology development.

8.6 Discussion and future trends

By terraforming activities, overexploitation of resources, and pollution, we alter the current ecological balance and risk forfeiting the future of humanity and many other species. At present, we still do not fully comprehend the significance of all factors influencing the Earth's ecosystem, and even for those factors influencing the Earth of which we are genuinely aware, we cannot accurately predict their effects, as the computational complexity involved is nothing less than prohibitive. In this regard, the development of better wireless sensor technology will help in the acquisition of knowledge and in better understanding of the underlying relations among component parts.

Decentralized data sensing, storage, processing, and interpretation, by means of currently available and in future technology, such as a worldwide network of intelligent agents, IoT, global-intelligence network, ubiquitous computing, etc., are the instruments in support of large-scale biodiversity monitoring and conservation actions.

Nevertheless, there is the danger that ubiquitous computing and all concomitant kinds of technological advancements may become the new instruments used by humanity in attempts to bypass the natural ecosystem regulation mechanisms, just to exploit more resources than the Earth can recuperate. As explained by Rees [17], the average world citizen has an eco-footprint of about 2.7 global average hectares, while there are only 2.1 global hectare of bio-productive land and water per capita on Earth. This means that humanity has already overshot^{au} global biocapacity by 30% and now lives unsustainably by depleting stocks of "natural capital" [17]. The ultimate way to deal with this unpleasant situation is to educate people so that we can constructively change the way in which we make use of resources and technological advancements.

References

- [1] A.R. Wallace, *The Geographical Distribution of Animals; With a Study of the Relations of Living and Extinct Faunas as Elucidating the Past Changes of the Earth's surface*, vol. 2, Macmillan and Co., London, 1876 (Chapter XXIII, pp. 553).
- [2] Millennium Ecosystem Assessment, *Ecosystems and Human Well-Being: Biodiversity Synthesis*, World Resources Institute, Washington, DC, 2005.
- [3] United Nations Convention on Biological Diversity (UN-CBD), *Handbook of the Convention on Biological Diversity*, third ed., (2005). <https://www.cbd.int/doc/handbook/cbd-hb-all-en.pdf>.

^{au}Earth Overshoot Day: https://en.wikipedia.org/wiki/Earth_Overshoot_Day.

- [4] D. Hill, M. Fasham, G. Tucker, M. Shewry, P. Shaw (Eds.), *Handbook of Biodiversity Methods: Survey, Evaluation and Monitoring*, Cambridge University Press, Cambridge, 2005. ISBN 978-0-511-12535-5.
- [5] T. Ganchev, Computational Bioacoustics: Biodiversity Monitoring and Assessment, first ed., De Gruyter, Berlin, 2017. ISBN 978-1-61451-729-0 June 2017. www.degruyter.com.
- [6] J. Patrício, S. Little, K. Mazik, N. Zampoukas, H. Teixeira, M.C. Uyarra, O. Solaun, N. Papadopoulou, A. Zenetos, G. Kaboglu, T. Churilova, O. Kryvenko, S. Moncheva, K. Stefanova, M. Bucăs, M. Elliott, Monitoring Networks Currently Used in European Seas, IMBER OSC 2014, June 24, Bergen. Available from: http://www.devotes-project.eu/wp-content/uploads/2014/07/8-Patricio-et-al_IMBER-2014_FINAL.pdf, 2014.
- [7] L. Mayer, M. Jakobsson, G. Allen, B. Dorschel, R. Falconer, V. Ferrini, G. Lamarche, H. Snaith, P. Weatherall, The Nippon foundation—GEBCO seabed 2030 project: the quest to see the World's oceans completely mapped by 2030, *Geosciences* 8 (2018) 63.
- [8] Global Biodiversity Outlook (GBO), Secretariat of the Convention on Biological Diversity, fourth ed., (2014). <https://www.cbd.int/gbo/gbo4/publication/gbo4-en.pdf>.
- [9] T.M. Aide, C. Corrada-Bravo, M. Campos-Cerdeira, C. Milan, G. Vega, R. Alvarez, Real-time bioacoustics monitoring and automated species identification. *PeerJ* 1 (2013). e103 <https://doi.org/10.7717/peerj.103>.
- [10] O. Jahn, K. Riede, Further progress in computational bioacoustics needs open access and registry of sound files, in: IBAC–2013, Brazil, 2013.
- [11] T. Ganchev, K. Riede, I. Potamitis, T. Stoyanova, S. Ntalampiras, V. Dimitriou, B. Nomikos, K. Birkos, O. Jahn, N. Fakotakis, AmiBio: automatic acoustic monitoring and inventorying of BIOdiversity, in: Poster at the IBAC2011, the XXIII Meeting of the International Bio-acoustics Council (IBAC), in La Rochelle, France, 12–16 September 2011, 2011.
- [12] O. Jahn, I. Mporas, I. Potamitis, I. Kotinas, C. Tsimpouris, V. Dimitrou, O. Kocsis, K. Riede, N. Fakotakis, The AmiBio project—automating the acoustic monitoring of biodiversity, in: Poster at the IBAC2013, the XXIV Meeting of the International Bio-acoustics Council, Brazil, 2013.
- [13] I. Potamitis, Automatic classification of a taxon-rich community recorded in the wild, *PLoS One* 9 (5) (2014) e96936.
- [14] I. Potamitis, I. Rigaklis, K. Fysarakis, Insect biometrics: optoacoustic signal processing and its applications to remote monitoring of McPhail type traps, *PLoS One* 10 (11) (2015) e0140474.
- [15] I. Potamitis, I. Rigaklis, Measuring the fundamental frequency and the harmonic properties of the wingbeat of a large number of mosquitoes in flight using 2D optoacoustic sensors, *Appl. Acoust.* 109 (August) (2016) 54–60.
- [16] EEA, EEA Signals 2009: Key Environmental Issues Facing Europe, 1831-2772European Environment Agency, 2009. ISBN 978-92-9167-381-0. <https://doi.org/10.2800/52418>. <https://www.eea.europa.eu/publications/signals-2009>.
- [17] W. Rees, The Human Nature of Unsustainability, <https://www.postcarbon.org/publications/human-nature-of-unsustainability>, 2011.

Further reading

- [18] K.H. Frommolt, R. Bardeli, M. Clausen (Eds.), Computational bioacoustics for assessing biodiversity, Proc. of the International Expert Meeting on IT-based Detection of

- Bioacoustical Patterns, Dec. 7–10, 2007, International Academy for Nature Conservation (INA), Isle of Vilm, Germany (Internationale Naturschutzakademie Insel Vilm), BfN-Schriften, 2008, 2007. <https://www.bfn.de/fileadmin/MDB/documents/service/skript234.pdf>.
- [19] T. Virtanen, M.D. Plumbley, D. Ellis (Eds.), Computational Analysis of Sound Scenes and Events, Springer International Publishing AG, 2018. https://doi.org/10.1007/978-3-319-63450-0_1.

This page intentionally left blank

The use of WSN (wireless sensor network) in the surveillance of endangered bird species

9

Amira Boulmaiz^a, Noureddine Doghmane^b, Saliha Harize^b, Nasreddine Kouadria^b, Djemil Messadeg^b

^a*Electronics Department, Faculty of Engineering Sciences, LERICA Laboratory, Badji Mokhtar University, Sidi Amar, Annaba, Algeria* ^b*Electronics Department, Faculty of Engineering Sciences, Laboratory of Automatic and Signals of Annaba (LASA), Badji Mokhtar University, Sidi Amar, Annaba, Algeria*

9.1 Introduction

Around the world, there are approximately 10,000 known bird species with on average several dozens to a few hundreds per country and up to several thousand per continent (Table 9.1) [1, 2]. However, according to new research led by the American Museum of Natural History [3, 4], the number of avian species is about 18,000. Birds react to possible changes in the environment and they are good indicators for biodiversity. They have also a major role in environmental safety, for example:

- the consumption of insects harmful to agriculture;
- seed dispersal for reforestation; and
- pollination of plants.

If some avian species begin to decline, this can highlight serious problems both in the food chain and in the environmental safety. For this purpose, monitoring wildlife in a general way and birds in particular can provide many precious insights into the environment, health, land productivity, etc.

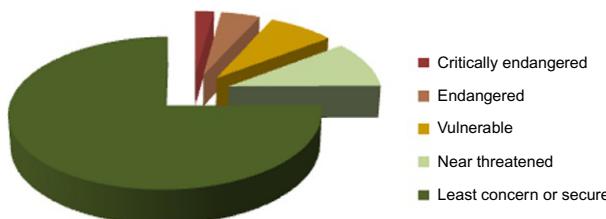
Bird monitoring is relevant to identify the species and to assess the progress in efforts to save biodiversity. This allowed, among other things, the definition of indicators, such as the IUCN Red List [5]. This list of threatened species, created in 1964, is the most comprehensive global inventory of the global conservation status of plant and animal species. It allows to monitor the evolution of endangered species in their natural habitat. The essential goal of this list is to propose an objective classification of very numerous animal species according to their extinction risk based on a set of relevant criteria. A series of Regional Red Lists are produced by

Table 9.1 Number of bird species per continent.

Continent	Number of bird species
South America	More than 3400
Asia	More than 2900
Africa	More than 2300
North America	More than 900
Australia	More than 820
Europe	More than 700
Antarctica	More than 65

countries or organizations, which assess the risk of extinction of species within a political management unit. According to Red List for birds by BirdLife International [5], these last years, the number of bird species listed as critically endangered has reached an all-time high. Indeed, 13% of the world's bird species are threatened, thus representing the third most threatened animal species after amphibians and mammals. Thus, 222, 461, and 786 species are classified as critically endangered, endangered, and vulnerable, respectively [5]. The total represents approximately an eighth of extant species. If we also take into account the 1017 species considered near-threatened, we will approach a quarter of all bird species in the world (Fig. 9.1).

The taxonomic distinction is one of the monitoring wildlife objectives. It is used for identification of IUCN Red List categories in the birds case (Table 9.2) [6, 7] and five quantitative criteria (A–E) are utilized. These measures and statistics are based on biological indicators of populations that are endangered or threatened with extinction, such as the rapid population reduction or its small size, the limitation or fragmentation of the occupancy geographic range, etc. This allows a regular measurement of species abundance distribution in time and space. Such information will not only help to ensure that threatened species are correctly recognized, but will also assist planning for their conservation. However, the biodiversity measurements are difficult, due to several constraints as the diversity of species, the diversity of environments, the variety of evaluation methods, the multiplicity of monitoring systems, etc.

**FIG. 9.1**

IUCN Red List status of all bird species in the world [5].

Table 9.2 First three criteria used for categories identification (IUCN Red List) [6, 7].

Criterion		Critical	Endangered	Vulnerable	Subcriterion
A	Population size reduction	>90% in 10 years	>70% in 10 years	>50% in 10 years	A1
		>80% in 10 years	>50% in 10 years	>30% in 10 years	A2, A3, A4, and A5
B	Occupancy geographic range	<100 km ²	<5000 km ²	<20,000 km ²	B1
		<10 km ²	<500 km ²	<2000 km ²	B2
C	Low population level and decline	<250 individuals	<2500 individuals	<10,000 individuals	C1
		<50 individuals	<250 individuals	<1000 individuals	C2

Criteria are as follows:

- A1: Population decline observed, estimated, inferred, or suspected in the past where the causes of the reduction are clearly reversible, understood, and have ceased.
- A2: Population decline observed, estimated, inferred, or suspected in the past where the causes of reduction may not have ceased, may not be understood or may not be reversible.
- A3: Population decline projected or suspected to be met in the future (up to a maximum of 100 years).
- A4: Population decline projected or suspected to be met in the future (up to a maximum of 100 years) where the time period must include both the past and the future, and where the causes of reduction may not have ceased or may not be understood or may not be reversible.
- B1: Extent of occurrence.
- B2: Area of occupancy.
- C1: An estimated continuing decline of at least 25% in 3 years, 20% in 5 years and 10% in 10 years corresponding to critically endangered, endangered, and vulnerable, respectively.
- C2: A continuing decline.

Criteria D and E, respectively, refer to:

- Population size estimated to numbers fewer than 50, 250, and 1000 mature individuals corresponding to critically endangered, endangered, and vulnerable, respectively; and
- quantitative analysis showing that the probability of extinction in the wild is at least 20% and up to a maximum of 100 years.

Table 9.3 IUCN Red List status of bird species in Algeria [8].

IUCN Red List status	Number of bird species
Extinct	0
Extinct in the wild	0
Globally threatened	15 (109th)
% threatened	5% (95th)
Critically endangered	3
Endangered	3
Vulnerable	9
Near-threatened	18
Least concern	284
Data deficient	0

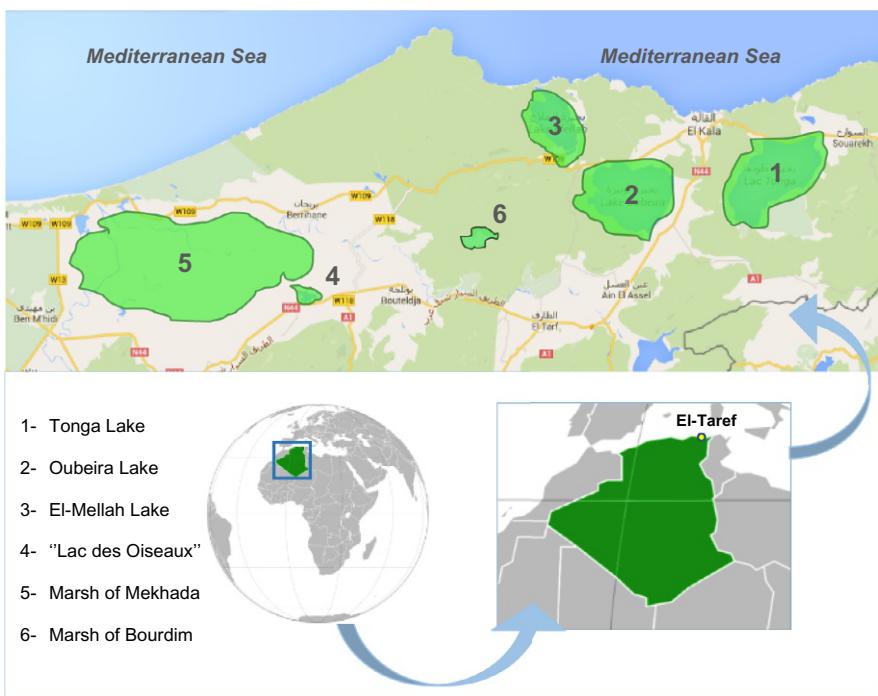
Algeria has more than 300 different bird species, sedentary and/or migratory birds, of which more than 100 are waterbirds especially in the humid region of El Kala (northeastern Algeria). At least 15% are threatened birds, critically endangered, endangered, vulnerable, or near-threatened [5, 8]. Table 9.3 presents the IUCN Red List status of bird species in Algeria. The numbers in brackets designate the rank of Algeria compared to other countries of the world.

Monitoring these birds in their natural habitats presents several points of interest. On the one hand, we have a reliable indicator to assess regularly the evolution of the number of endangered birds in a region. On the other hand, the birds can be considered as a tool in an environmental monitoring. This monitoring, which consists of regular observations and measurements on these birds, should allow for protection measures if unfavorable changes will be observed.

9.2 Study area

The El-Taref is among the most important wetlands in Algeria (Fig. 9.2). Located in the Western Palearctic area (south of the Mediterranean), crossed by two major flyways of the East Atlantic (East Atlantic Flyway) and Black Sea/Mediterranean, (Mediterranean/Black Sea Flyway) [9–12], they are special sites for tens of thousands of birds to overwinter or make a temporary stop. The wetlands of Annaba and El Taref are essential breeding and wintering sites for several rare, threatened, or restricted species according to IUCN, including the White-headed Duck (*Oxyura leucocephala*) and the Ferruginous Duck (*Aythya nyroca*).

The El Kala National Park (PNEK) ($36^{\circ} 52' N$, $8^{\circ} 27' E$), located in the extreme northeast of Algeria in the Wilaya of El Taref, was created in 1983 by the decree n° 83–462, classified as a biosphere reserve in 1990, and includes nine communes entirely contained in the Wilaya of El Taref (Fig. 9.2). This integral reserve covers

**FIG. 9.2**

Geographical location of the main wetlands of northeast Algeria (El-Taref).

an area of 76,438 ha, or 26% of the area of the Wilaya. The region is bounded on the east by the Algerian-Tunisian border, on the north by the sea, on the west by Cape Rosa and on the south by the foothills of the “Jebel El Ghorra” [13–15].

The PNEK is considered to be the most important site for breeding waterbirds in Algeria and one of the most important in the Mediterranean. Its mission is to ensure the conservation of the precious natural heritage and owes its fame to its wetlands, giving it the title of the main center of biodiversity in the Mediterranean. Heavily forested (more than 69% of its surface area), the PNEK extends over a 40 km coastal strip and runs along the Tunisian border for 98 km. More than 120,000 inhabitants live on this territory. The human pressure on fauna and flora makes them very vulnerable. Economic resources in the PNEK area show that agriculture, tourism, and fishing remain the main activities. However, their organization in space and time reveals inconsistencies that they are detrimental to the conservation of this natural environment: overgrazing, uncontrolled fishing, unregulated beach tourism, and many illegal activities.

The PNEK represents a reservoir of Mediterranean biodiversity, with 1264 plant species, or 32% of the Algerian flora and 878 animal species. It contains many rare and endangered species according to the IUCN lists. PNEK has several wetland sites

Table 9.4 Description of El-Taref main wetlands and their international status (Ramsar site and/or IBA).

Sites	Area (ha)	Current status	Altitude	Geographic coordinates	
				Longitude	Latitude
Tonga Lake	2400	Ramsar (1982), IBA (2001)	0–5 m	08°31'E	36°53'N
Oubeira Lake	3160	Ramsar (1982), IBA	10 m	08°23'E	36°50'N
El-Mellah Lake	875	Ramsar (2004)	0–1 m	08°19'E	36°54'N
Lac des Oiseaux	70	Ramsar (1999), IBA	0–5 m	08°07'E	36°46'N
Marsh Mekhada	15,000	Ramsar (2003), IBA	0–5 m	08°00'E	36°48'N
Marsh Bourdim	25	Ramsar (2009), IBA	10–16 m	08°15'E	36°48'N

classified by the 1971 Ramsar^a Convention and IBA,^b characterized by a great diversity of ecosystems with an invaluable biological wealth (Table 9.4).

9.3 Study bird species

We were interested in this study in two bird types: the White-Headed Duck and the Ferruginous Duck. About 173 White-headed Duck birds on average were identified during two breeding seasons in Tonga Lake [16]. More numerous, the breeding population of Ferruginous Duck was estimated at over 700, more than half of whom are males [17]. The choice of these two species is due to their status as threatened and near-threatened species, respectively, included in the Red List of the IUCN. These are also protected by the Algerian legislation as being threatened with extinction [18]. Indeed, the White-headed Duck especially, with the Marbled Teal, the Audouin's Gull, the Eleonora's Falcon, and the Algerian nuthatch (*Sitta ledanti*, which is the only bird species endemic to Algeria) are among the birds of highest conservation concern in Algeria [10].

9.3.1 The white-headed duck

The global white-headed duck (scientific name: *Oxyura leucocephala*) population is estimated at fewer than 10,000 mature individuals distributed over a fragmented range located mainly around the Mediterranean. In Algeria, this species is composed of two populations: the first sedentary and breeding one while the second is only wintering and more numerous [19, 20]. Currently, Tonga Lake and "Garaet Hadj Tahar" (Skikda, northeast of Algeria) host the highest numbers of this species in Algeria and North Africa.

^a Convention on Wetlands of International Importance especially as Waterfowl Habitat.

^b Important Bird Area.

The population is estimated to 2500 individuals in Spain and Morocco [21]; 400–600 individuals in Algeria and Tunisia [22]; 5000–10,000 individuals in the east Mediterranean and southwest Asia, and 10 individuals in south Asia. This totals 7900–13,100 individuals, equating to roughly 5300–8700 mature individuals.

Habitat degradation, environmental factors, climate change, and hybridization with its congeneric Ruddy Duck (*Oxyura jamaicensis*) are thought to be the most important threats [23, 24]. The White-headed Duck is known to have been resident in Algeria for a long time [25, 26]. At Tonga Lake, its numbers have increased substantially over the past four decades, from 40 individuals in 1976 to 1045 in 2010 [27]. This population growth is most likely due to the effective protection measures implemented at the site and a possible immigration of the nearby “Lac des Oiseaux” population, which has suffered significant degradation over the last decade [19].

9.3.1.1 General description and biometric feature (voice)

The White-headed Duck is a small duck about 45 cm long with a small stiff tail. We can differentiate the duck sex from the color of the plumage and the bill (Fig. 9.3). Indeed, the male has a white head with black crown, a blue bill, and a reddish-gray plumage, while the female has a dark bill and rather duller coloring [29]. Usually silent, the White-headed Duck is sometimes heard grunting and buzzing in collective parades. The female sometimes emits shrill cries.

9.3.1.2 Habitat and social behavior

Inside the country, it turns out that the attendance of wetlands depends on two inter-related factors: the degree of salinity and the presence of helophytes (mostly *Phragmites australis*, *Scirpus lacustris*, *Carex divided*) in the water's edge. According to the IUCN list, the status of the White-headed Duck is “Endangered” (Table 9.5).

Table 9.5 IUCN Red List status of White-headed Duck [30].

Least concern	Near-threatened	Vulnerable	Endangered	Critically endangered	Extinct in the wild	Extinct
LC	NT	VU	EN	CR	EW	EX

9.3.1.3 Annual statistics of individuals

In the Mediterranean basin, the White-headed Duck is mostly present in Spain, Algeria, and Tunisia. The effective conservation plan including habitat protection, hunting restriction, and captive breeding played a crucial role in the increase of the local population to 2500 individuals [31]. However, recent estimations of Algerian and Tunisian populations have shown smaller numbers with a total of a few hundred individuals for both countries. In Algeria, this population is restricted to small colonies distributed in northeastern wetlands (El Kala and Guerbes-Sanhadja complex) [27] and eastern high plains [20, 32].

Generally, the White-headed Duck is present in Tonga Lake throughout the year. The number of individuals increases significantly in January and can reach more than 356 ducks. Then, the number gradually decreases until April and slightly recovers by the end of the breeding season in July (Fig. 9.3).

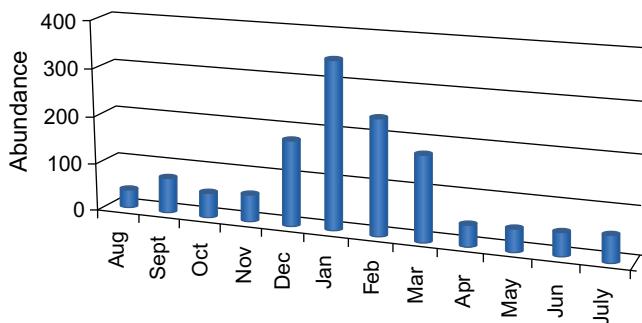


FIG. 9.3

Monthly counts of the White-headed Duck at Tonga Lake (August 2011–July 2012) [28].

9.3.2 The ferruginous duck

The Ferruginous Duck (scientific name: *Aythya nyroca*) is listed as “Near-threatened” on the IUCN Red List (Table 9.6) and is a priority species on three prominent international conservation treaties: the European Union Birds Directive, the Bern Convention and the African Eurasian Migratory Waterbird Agreement (AEWA) [34]. There have been international concerns about population declines and range contraction in the western Palearctic, and an international species action plan has been developed to help to conserve the species [35].

The Ferruginous Duck is sedentary in North Africa [36]. The species is significantly represented in the majority of the main wetlands of northeast Algeria. The key threat is the loss of its habitat in wetlands rich in vegetation and shallow after drainage and intensification of fish farming, development of dams, and construction of flood plains. Increasing drought due to global climate change has a negative impact on the species, especially on its distribution.

9.3.2.1 General description and biometric feature (voice)

Rather discreet, the females produce a “*Kerr Kerr Kerr...*” dry, rolled, humming, and resonant characteristically. Males emit “*vih-viu*” during courtship and a harsh and tinny staccato cry “*tik-tik-tik ...*” [37]. Generally, males produce a squeaky breathing while that of females is noisier and rougher [38].

The Ferruginous Duck breeding male is a rich chestnut color, with a darker back and a yellow eye. In flight, the white belly and underwing patch are visible helps to distinguish this species from the other similar ducks. The females are similar but browner and have a dark eye.

9.3.2.2 Habitat and social behavior

Ferruginous Ducks breeds and winters on shallow lakes, marshes and pools rich in submerged vegetation and floating, generally avoiding large open areas of water. Accustomed to relatively closed habitats, this species spends most of the time swimming, diving, and resting on vegetation.

Table 9.6 IUCN Red List status of Ferruginous Duck [33].

Least concern	Near-threatened	Vulnerable	Endangered	Critically endangered	Extinct in the wild	Extinct
LC	NT	VU	EN	CR	EW	EX

9.3.2.3 Annual statistics of individuals

The Ferruginous Duck is a poorly studied partial migrant widely distributed in Europe, Asia, and Africa. The total breeding population in Europe varies between 11,000 and 25,000 pairs [39]. This species winters in the majority of wetlands of northeast Algeria. This wintering duck forms homogeneous or mixed groups with other similar species observed at Tonga Lake. Ferruginous Ducks start coming to Tonga Lake in early September with an initial number of several hundred birds, followed by a progressive increase to reach, at the end of April, a total exceeding 1000 ducks (Fig. 9.4). Tonga Lake represents both a regular winter site and an excellent breeding area for this species and for several other waterbirds [40].

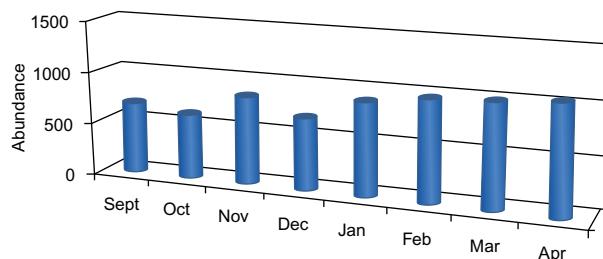


FIG. 9.4

Weekly counts of Ferruginous duck populations across Tonga Lake during wintering season from 2002 to 2009 [40].

9.4 Measurement and data collection

The first methods of monitoring bird populations were the manual collection of information on the behavior of individuals by observers in the field. For example, we can cite the work of [40], which studies the behavior of a population of Ferruginous Ducks in the wetlands of PNEK, or those of [20], who are interested in ecology and behavior throughout the day of the White-headed Duck in the lake “Hadj Tahar” (Skikda, northeast Algeria). The goal of these two works is to study the evolution of the number of these birds and their spatial distribution in the site, by observing the individuals in a discontinuous way (on average three times per month).

However, in some cases this technique is subject to the problem of differentiating individuals of the same species. It becomes difficult to guarantee whether two individuals observed at different times are identical or not. This is the pitfall facing specialists studying birds.

The sound emitted by birds is their most attractive factor. Indeed, in most cases (even in the presence of dense vegetation), we can hear a bird without seeing it. In fact, most bird vocalizations are specific to each species. Therefore, it is a natural and appropriate way automatically to study and identify bird species through their vocalizations. McIlraith and Card [41] were among the first to apply automatic classification to a large number of bird species using their sounds. In recent years, recognition approaches have been proposed for the automatic classification of bird sounds [42–44]. The most commonly used functions for bird sound recognition include spectral characteristics such as Linear Predictive Coding (LPC) coefficients [45], spectral density [46], wavelet coefficients [47], spectral bandwidth [48], and cepstral characteristics such as Mel Frequency Cepstral Coefficients. In the majority of these studies, the databases used in the experiments consist of preprocessed sound recordings of different bird species (e.g., a commercial CDROM) [49, 50]. However, in the real case of habitat monitoring, bird sounds are subject to innumerable environmental noises (such as the sounds of other animals, bad weather and means of transport), thus distorting the correct recognition of the individual.

Habitat and environmental monitoring is a class of applications of sensor networks offering innumerable potential benefits to scientific communities and civilization as a whole. Natural space equipment with many wireless sensor nodes can support long-term, large-scale data collection, which is difficult if not impossible to achieve otherwise. The intimate connection with its immediate physical environment allows each sensor to provide localized measurements and detailed information that is difficult to obtain using traditional equipment. The aggregation of many of these sensor nodes is called wireless sensor networks (WSNs). In this work, we propose implementing a system for automatic recognition of bird sounds on a WSN.

In the following, we discuss the problem of collecting data in real-life situations. We compare the most commonly used methods for habitat monitoring: first, the

manual observation method, which requires the continuous presence of observers in the field; and second, surveillance using mobile devices such as GPS^c-based portable sensors.

9.5 Habitat monitoring and wildlife information gathering

9.5.1 Conventional observation method

Retrieving information manually on the behavior of individuals through field observers was the first method of collecting data on animal populations. This technique is still widely used. For example, we can cite the work of Ref. [51], which studied the behavior of a dolphin population in New Zealand, or that of Ref. [52], which was interested in the behavior of apes. Both studied the relational structures of these animals using a social network built on the observation of social contacts between individuals. One of the major problems of that technique is the way of differentiation of individuals.

A solution proposed by Ref. [53] in a study conducted on giraffes is, for example, to differentiate individuals by observing the natural marks on their necks. However, in the case of birds, the visual identification of an individual is difficult for different reasons [54]:

- The dense vegetation, which is a characteristic feature of their habitat, makes it difficult to access the site and the location of the birds.
- Human presence can disrupt animals and modify their behavior, thus biasing the information collected.
- The duration of the observation campaigns is limited and depends on the availability of the teams. However, it is important to have enough data that is spread over long periods of time.

9.5.2 Mobile devices

Recent improvements in micro-technologies are at the origin of new types of devices that can provide spatiotemporal data continuously and in real time. Beginning in the mid-2000s, the increasing miniaturization of these systems enabled their use in the automatic collection of wilderness information. Indeed, research projects have exploited the new capabilities proposed by what is now known as “wearables sensors” [55]. For example, GPS technology has been used for several years to track and identify the position of animals. The problem of identifying individuals does not arise since the sensor is worn by an individual and only one. This device could be used in situations where large-scale observation was not possible—for example, to study marine animals [56] or in the mountains to follow elk [57].

^c Global Positioning System.

However, although the GPS system placed on animals is currently the most widely used method, it was not feasible in contexts such as waterbirds for the following reasons:

- It only works effectively in open areas and free from any object that may obstruct the viewer's fields of view. Operation under dense foliage, which is the environment of waterfowl, is not possible.
- It is particularly expensive because it is necessary to capture and equip all individuals.
- It consumes a lot of energy.
- It sometimes has a nonnegligible weight, which can handicap the movements of small birds.

These major drawbacks for the study of waterbirds have led to other collection devices that are fixed sensors.

9.5.3 Fixed devices

Fixed sensors, such as wireless sensor nodes, have already been used in some studies for social data collection. The study in Ref. [58] has, for example, implemented sensors, equipped with sound and video recorders to detect and analyze social interactions within a retirement home. Depending on the peripherals associated with the sensors, this type of device makes it possible to detect a varied number of interactions provided that they can be identified. Unlike GPS, these devices are fixed and do not need to be installed on the individuals studied. They are generally installed on the ground or on equipment fixed on the ground.

However, the use of this type of devices requires the ability to identify individuals and interactions accurately. Typically, in the case of a video recorder, image sequence analysis algorithms must be put in place to allow recognition of the individuals as well as the type of interaction. In addition, the data collected on individuals are not regular over time. Individuals are detected only when they are in the field of action of the sensors. For example, the work of Ref. [58] remains limited to a small space and is facilitated by the diversity of the recorders with which the sensors are equipped. However, on much larger spaces, it is necessary to place a large number of sensors in order to detect the maximum of sounds and vocalizations of birds. These devices must also be able to identify, filter, and communicate effectively the information they record to centralized systems that collect and process the data.

9.5.4 Comparison

In the context of information gathering and audio recognition of birds, an ideal collection method must be able to guarantee certain major functional constraints such as the respect for birds' behavior and their habitat, and therefore the ability to function under dense foliage. It must also be able to cover an area of relatively large size, of the order of several hundred square meters, and be able to detect individuals and their interactions effectively. In Table 2.1, we compare three families of techniques

against a number of functional criteria defined for the study. The first six criteria, unavoidable for the current project, preclude the use of mobile devices. Indeed, although mobile devices make it possible to carry out the collection over a long period and do not contribute to the deterioration of the habitat, they require the capture and the equipment of these birds, which can cause high implementation costs and also handicap animals in their movements. In addition, the operation under dense foliage is not guaranteed, which can affect the quality of the data collected in the context of studies conducted on waterfowl.

Table 9.7 gives a concise comparison between these three methods of data collection according to different criteria [54].

Table 9.7 Comparison of the three data collection techniques [54].

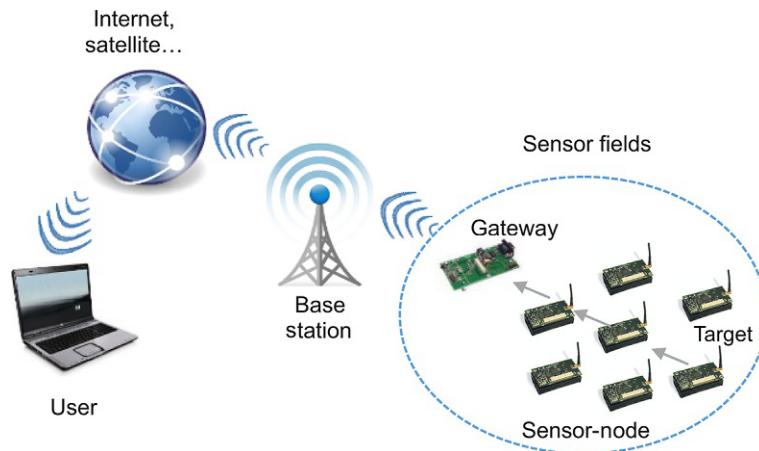
Functional criteria	Manual method	Mobile equipment	Fixed equipment
Study duration (time)	Short	Long	Long
Habitat deterioration	Yes	No	No
Capture obligation	No	Yes	No
Financial cost	Average	High	Average
Behavior alteration	Yes	Yes	No
Operation under dense foliage	No	No	Yes
Identification of individuals	Difficult	Easy	Average
Regular data	No	Yes	No
Study area	Limited	Large	Variable

Thus, the major difficulties related to the use of mobile devices in the habitat of birds, have led us to the employment of fixed sensors. In addition to their compliance with the major functional requirements of the project (the first six), one of the advantages of sensors is that of being able to collect several sources of data (sounds, images or videos), thus making it possible to consider various types of methods for the recognition of individuals and their interactions. However, the sensors have the disadvantage of not providing data at regular time intervals since the individuals are only detected when they are in the detection field of a sensor. Thus, the amount of data depends essentially on the surface of the study area covered by the devices.

9.6 Wireless sensor networks

9.6.1 WSNs principles

During the last two decades, WSNs have increasingly intrigued the scientific and professional communities [59]. Indeed, the application perspectives and use domains are in perpetual increase [59]: military, environment, healthcare, and security.

**FIG. 9.5**

Wireless sensor networks topology.

A WSN is a network of sensor nodes that cooperatively sense, control, and transmit relevant information of a given environment (Fig. 9.5). WSNs offer many advantages, for example:

- ease of deployment;
- installation in hostile and difficult to access environments;
- scalability; and
- use of several types of wireless communication technologies.

WSN also have disadvantages including, for example:

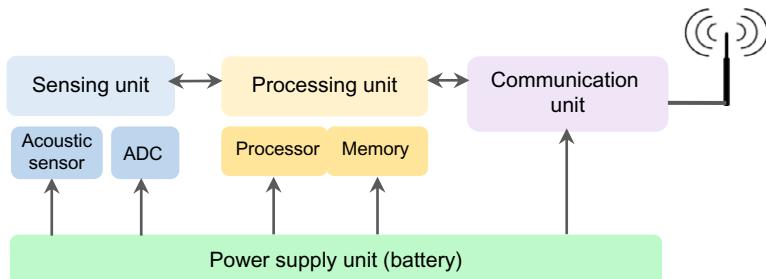
- limited lifetime due to battery power;
- limited storage and computation capabilities;
- limited bandwidth; and
- vulnerability against security attacks.

Despite these disadvantages, WSNs are used widely in many domains and today they represent among others the spearhead of Internet of Things (IoT) [60].

9.6.2 Wireless sensor node structure

A sensor node, called also a mote, is the basic element in a WSN. It senses the environment, processes the sensed information, and communicates with the other nodes within the network. This is a particular example of a telecommunication embedded system. Table 9.8 gives certain characteristics of some popular wireless sensor nodes.

Generally, a sensor node is composed of a processing unit, a communication unit (transceiver), a power supply, and one or more sensors (Fig. 9.6).

**FIG. 9.6**

Example of wireless sensor node structure.

Processing unit: The processing unit, which is often a microcontroller, assures the digital signal processing and manages the other node units. For some nodes, the processing unit may be of a developed technology such as a digital signal processor or even an FPGA target [61]. The controller is usually accompanied by a storage memory. Flash memory is the most used because of its relatively high storage capacity and its low cost.

Communication unit (transceiver): The transmission used is often radio frequency (RF). However, some sensor nodes have optical communication (laser) or infrared units. Infrared just like the lasers, has limited broadcasting capacity. In RF, the WSN often make use of the industrial, scientific, and medical (ISM) radio bands such as frequencies 173, 433, 868, 915 MHz or the 802.15.4 norm of 2.4 GHz [59].

Power supply unit: The power supply of a wireless sensor node, which is often in a hard-to-reach place, can only be ensured by a battery. Nevertheless, frequent change of the battery can be difficult and costly. This aspect represents the major disadvantage of the WSN. Among the known developments in the WSN is the extension of their lifetime using multiple strategies, such as:

- reducing the energy consumption of the controller and the transmission unit;
- shutting down parts of the sensor node which are not currently used;
- adapting the power supply levels within the sensor node according to the workload; or
- improving battery electrical storage technology using others electrochemical material. Today, the electrochemical materials used are NiCd (nickel-cadmium), NiZn (nickel-zinc), NiMH (nickel-metal hydride), and lithium-ion [62].

Sensing unit: This may generally include several sensor units and each sensor produces a measurable response to a change in a physical condition such as temperature, humidity, or light. This unit is usually composed of two subunits: sensors and analog-to-digital converters (ADCs). The ADC converts the raw signals delivered by the sensors into digital pulses, which can be retrieved by the controller via a specific link [63].

Table 9.8 Characteristics of some popular wireless sensor nodes.

Node	Manufacturer	Controller	RF	RAM (bytes)	Operating system
MICA2	MEMSIC, United States	ATmega128L—16 bit 8MHz	433/315 or 868/916MHz 2.4 GHz IEEE 802.15.4	4k	TinyOS
Telos	University of California, Berkeley/Sentilla (Moteiv)	TI MSP430—16 bit 8MHz		10k	TinyOS
MICAz	MEMSIC, United States	ATmega128L—16 bit 8MHz	2.4 GHz IEEE 802.15.4	4k	TinyOS
ez430-RF2500	Texas Instrument, United States	TI MSP430—16 bit 16MHz	2.4 GHz SimpliciTI	1k	TinyOS
WSN430	FIT IoT-LAB	TI MSP430—16 bit 8MHz	ISM band (315–915MHz) IEEE 802.15.4 2.4 GHz	10k	FreeRTOS, Contiki, Riot, TinyOS, OpenWSN
VirtualSense	/	TI MSP430—16 bit 25MHz	2.4 GHz IEEE 802.15.4	16	JAVA
Wasp mote	Libenium	AT Atmega 1281	868/900/2400MHz	8k	Libelium OTAP ^a
LOTUS	MEMSIC, United States	NXP LPC1758 32 bit ARM Cortex-M3	2.4 GHz IEEE 802.15.4	64k	RTOS, Mote Runner, TinyOS
Shimmer	TI MSP430F1611	TI MSP430F1611	2.4 GHz IEEE 802.15.4	10k	TinyOS

^aOver the air programming.

9.6.3 WSNs for wildlife monitoring

Great Duck Island [64] was one of the first WSNs implemented for wildlife monitoring. The Berkeley sensor nodes Mica [65] were used to measure temperature, humidity, and atmospheric pressure, and to detect the presence of birds. The recordings, made by the sensor nodes, are periodically sampled and transmitted from the local sink node to the base station on the island. The base station sends the data using a satellite link to a server connected to the Internet. In the same context, other works followed as [66–68] and propose an algorithm using wireless sensors fitted with microphone for automatic counting of singing birds in their habitat. Sensors are used to record audio samples of bird songs, which will then be analyzed to extract fingerprints that permit identification of the bird. Since then, numerous works have been proposed to monitor wildlife (including birds monitoring) using WSNs [69–71].

9.7 Methodology

Fig. 9.7 gives an overview of our approach to an automatic bird sound recognition system using WSNs. Nevertheless, several limitations were faced when setting up our system, namely: environmental noise, attenuation of the sound signal (bird sounds), and energy capacity of the wireless sensor nodes.

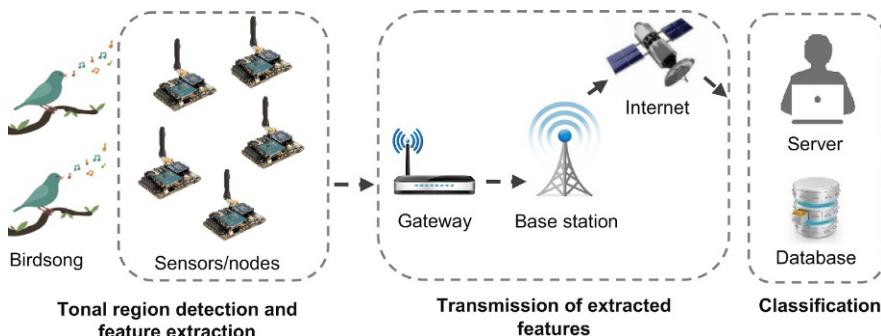


FIG. 9.7

Automatic recognition system for bird sounds using WSN.

9.7.1 Environmental noise

Automatic recognition of bird sounds in their natural habitats is subject to many constraints and difficulties, such as the following:

- The field recordings are strongly corrupted with environmental noises,
- The birds are usually far apart from the recorder (or sensor node), so the captured calls/songs will be faint and weak.
- A large number of bird sounds of different species can be simultaneously captured by a same recorder (sensor node).

Among all these constraints, environmental noise may be the most degrading. It is therefore necessary to reduce its intensity or even remove it during preprocessing in order to suggest a robust automated recognition methodology. The environmental noises are mainly due to human activity, to other biophony species, and to geophonic and/or anthropophony sources. It should also be noted that environmental noises can disrupt birds and thus influence the sounds they generate. For example, studies for some types of birds have shown the impact of noise on birds sounds, singing, particularly [72]. In most cases, these noises are low frequencies, which affect the recognition system efficiency by hiding the low-frequency sounds of some birds or by leading to false detections [73, 74]. Some other works, on the other hand, present the capture of several sounds of birds of different species simultaneously by the same recorder, as the main obstacle to implement an efficient automatic recognition system [74, 75].

There are different types of approaches for denoising recorded bird sounds. Some of these techniques propose the use of filters [42, 76–78], others recommend the spectral subtraction [79] inspired by Boll's works [80], while recent works use packet wavelet transform [81, 82] or even combinations of several methods such as minimum mean square error short-time spectral amplitude estimator (MMSE STSA) [83, 84].

We can divide the environmental noise according to its temporal evolution in three categories—continuous noise, intermittent noise, and impulsive noise:

- Continuous noise has a globally constant sound pressure level over time (e.g., rain, wind).
- Intermittent noise is characterized by a succession of noise phases each having a different level and spectrum of frequencies (e.g., the passage of an airplane or a vehicle, the cry of an animal).
- Impulse noise is a brief sound event characterized by a punctual high peak sound pressure value (e.g., thunder).

Fig. 9.8 shows signal examples, sampled at 44.1 kHz, representing the Ferruginous Duck cry corrupted by a continuous noise, an intermittent noise, and an impulse

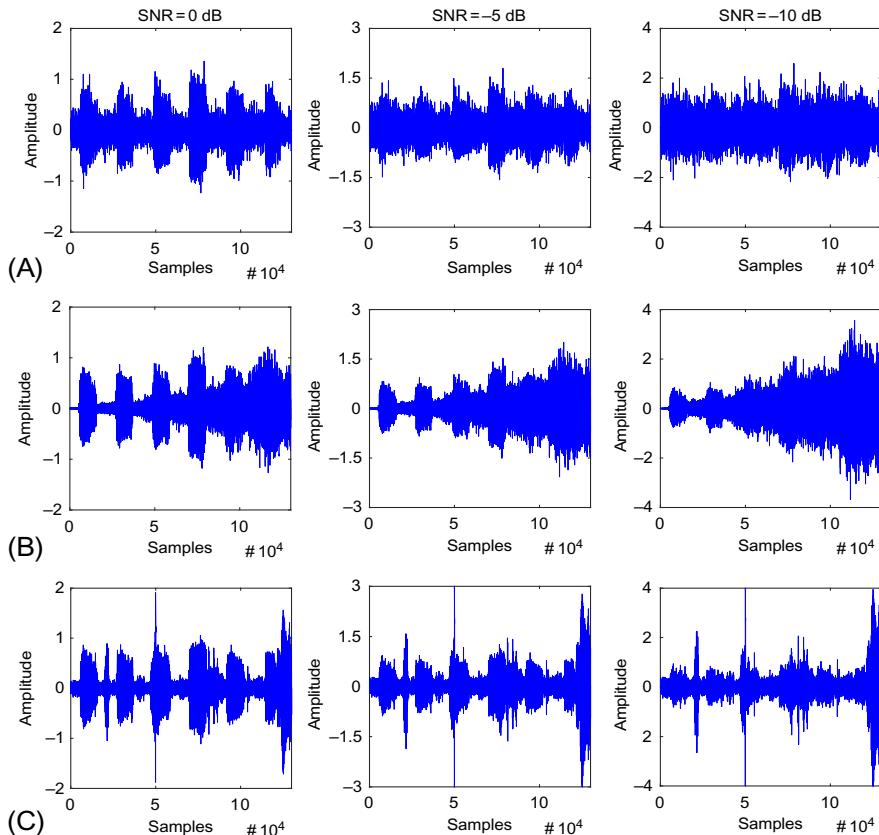


FIG. 9.8

Ferruginous Duck cry waveform, disturbed by (A) continuous noise, (B) intermittent noise, and (C) impulse noise, with a global SNR between 0 and -10dB with a step of 5dB .

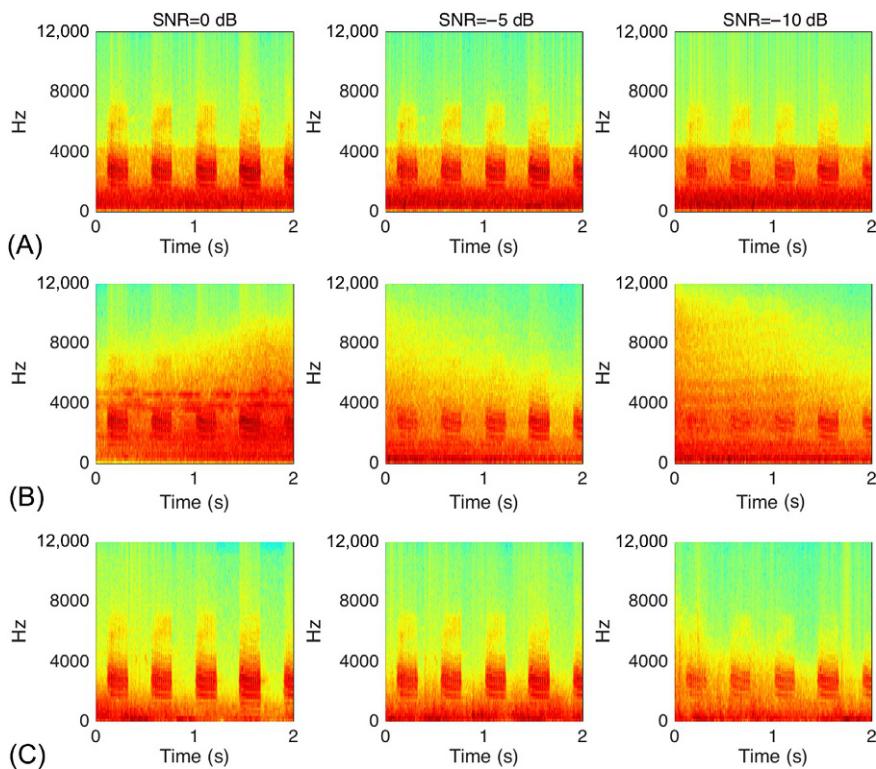


FIG. 9.9

Spectrogram of Ferruginous Duck cry, disturbed by (A) continuous noise, (B) intermittent noise, and (C) impulse noise, with a global SNR between 0 and -10dB with a step of 5dB .

noise, respectively, considering a global SNR between 0 and -10dB with a step of 5dB . Fig. 9.9 shows the spectrograms of these same signals.

9.7.2 Limited computational capacity of wireless sensor nodes

Usually a WSN consists of hundreds to thousands of low-power multifunctional sensor nodes, operating in an unattended environment, and having sensing, computation and communication capabilities. As shown in Fig. 9.6, the basic components of a node are a capture unit, a central processing unit (CPU), a power unit, and a communication unit. The sensor nodes are usually powered by batteries. However, because a sensor network contains hundreds to thousands of nodes, and because WSNs are often deployed in remote or hostile environments, it is difficult to replace or recharge batteries. The power is used for various operations in each node, such as running the sensors, processing the information gathered and data communication. The communication step between sensor nodes consumes most

of the available power, much more than sensing and computation. Computational capacity is linked with the available amount of power. Since there is a limited amount of power, computational capacity is also constrained. Although it is acknowledged that sensors are not expected to have the computing power of workstations or even mobile handheld devices, researchers and developers are greatly concerned with the issue.

9.8 Bird species recognition systems

Generally, in an automatic bird-recognition system, the sound signal is first preprocessed (denoising), windowed, and segmented, then features are extracted and finally a statistical model is identified [76]. The most widely used modeling techniques for acoustic bird identification are based on dynamic time warping (DTW) [85], on support vector machine (SVM) [86], on artificial neural networks (ANN) [87], on the hidden Markov model (HMM) [74, 88], or on the Gaussian mixture models (GMMs) [88, 89]. Nevertheless, the effectiveness of the identification depends essentially on the relevance of the parameterization step [85]. Indeed, an effective parameterization consists of transforming the bird sound signal into a set of relevant acoustic features, highly discriminating and nonredundant. The most used parameters in an automatic bird-recognition system are the linear Predictive Coding (LPC), the linear prediction cepstrum coefficients (LPCC) and the Mel frequency cepstral coefficients (MFCC) [85, 90].

9.8.1 Acoustic bird recognition system: Proposition 1

In Ref. [91], a new real-time approach for audio recognition of waterbird species in noisy environments, based on an embedded system, was proposed. The authors were interested in monitoring and recognition of waterbird species of Tonga Lake using embedded systems. The automatic bird sound recognition system was based on two main phases: the capture and preprocessing phase, and the recognition phase.

The recognition phase consists of the following steps. First, the tonal region detection (TRD) step is introduced to separate natural acoustic noise from bird sound. Second is the feature extraction step, which consists of the voiceprints extraction from segments of foreground bird sound using Mel Frequency Cepstral Coefficients with Spectral Subtraction (MFCC-SS) integration algorithm. Finally, the extracted voiceprints are compared with a database for classification using Support Vector Machine (SVM). This study focuses only on the first and second steps.

9.8.1.1 Preprocessing and segmentation: TRD

Bird vocalization is commonly considered to be composed of calls and songs, which comprise a unique syllable or a series of syllables. Sounds produced by birds may be of a various nature. Some birds produce sounds of a noisy broadband nature, but most produce a tonal sound, which may consist of a pure tone frequency, several

harmonics of the fundamental frequency, or several nonharmonically related frequencies [92].

Consider the noisy signal of the bird sound in the discrete time domain, given by

$$y(i) = b(i) + n(i) \quad (9.1)$$

where $y(i)$ is the observed noisy bird sound, $b(i)$ is the uncontaminated bird sound, and $n(i)$ is the background noise.

In order to develop the technique of tonal region detector in bird sound, we have to work under the following assumption:

$$E(|Y|^2) = E(|B|^2) + E(|N|^2) \quad (9.2)$$

where E denotes the statistical expectation operator.

However, the noise signal $n(i)$ can be correlated to $b(i)$, but the noise signal $n(i)$ can be correlated to $b(i)$. Indeed, the background noises can be of various types such as white noise, impulse noise, and even sounds caused by human activity, other biophony species, wind, rain, and anthropogenic interference. However, noises caused by the natural environment (human activity, other biophony species, wind, rain, etc.), unlike other perturbations such as white noise, cannot be considered completely uncorrelated with the bird vocalizations. Those are audible sounds with characteristics, especially spectral, relatively close. For this purpose, the assumption of Eq. (9.2) may not be valid.

To show that this assumption from Eq. (9.2) nevertheless remains valid regardless of the type of noise, we have

$$\Gamma_{yy}(i) = \Gamma_{bb}(i) + \Gamma_{nn}(i) + \Gamma_{bn}(i) + \Gamma_{nb}(i) \quad (9.3)$$

Γ_{yy} , Γ_{bb} , Γ_{nn} , Γ_{bn} , and Γ_{nb} are, respectively, the autocorrelation of the noisy bird sound $y(i)$, the autocorrelation of the uncontaminated bird sound $b(i)$, the cross-correlation of $b(i)$ against $n(i)$, and the cross-correlation of $n(i)$ against $b(i)$.

Therefore, the power spectrum of the distorted speech can then be obtained as

$$|Y(k)|^2 = |B(k)|^2 + |N(k)|^2 + 2|B(k)||N(k)|\cos(\theta_b + \theta_n) \quad (9.4)$$

where $B(k)$ and $N(k)$ represent, respectively, the spectrum coefficients of the clean bird sound $b(i)$ and the additive natural noise $n(i)$. θ_b and θ_n denote the (random) angles between the two complex variables $B(k)$ and $N(k)$, respectively. So, if $\cos(\theta_b + \theta_n)$ is set as 0, in the case where $b(i)$ and $n(i)$ are completely independent, Eq. (9.4) will become

$$|Y(k)|^2 = |B(k)|^2 + |N(k)|^2 \quad (9.5)$$

In order to prove this hypothesis, sounds of different bird species are interfered with various types of environmental noise, which are wind noise for the continuous type, train noise for the intermittent type, and thunder noise for the impulsive type. For each case, cross-correlations $\Gamma_{bn}(i)$ and $\Gamma_{nb}(i)$ are relatively negligible compared to autocorrelations $\Gamma_{bb}(i)$ and $\Gamma_{nn}(i)$ for different levels of SNR comprised between -10 and 20 dB. Now, the assumption of Eq. (9.3) can be considered legitimate.

Let us define the a priori SNR by $\xi = \sigma_b^2 / \sigma_n^2$ and a posteriori SNR by $\gamma = |y|^2 / \sigma_n^2$, where $E(|B|^2) = \sigma_b^2$ and $E(|N|^2) = \sigma_n^2$ are the spectral bird sound and noise power, respectively. The quantities estimation are designated by a hat symbol, e.g., $\hat{\sigma}_n$ is an estimate of σ_n .

Now, let us define H_0 and H_1 as the “tonal region absence” and the “tonal region presence” hypothesis, respectively. Under the assumption that the spectrum coefficients of both sound and noise are complex Gaussian distributed, as given by [93], the conditional probability density functions (PDFs) of the observation are

$$P(Y|H_0) = \frac{1}{\pi\sigma_n} \exp\left(-\frac{|Y|^2}{\sigma_n^2}\right) \quad (9.6)$$

$$P(Y|H_1) = \frac{1}{\pi(\sigma_b + \sigma_n)} \exp\left(-\frac{|Y|^2}{\sigma_b^2 + \sigma_n^2}\right) \quad (9.7)$$

By applying Bayes’ theorem, the a posteriori tonal region presence probability (TRPP) is

$$P(H_1|y) = \left\{ 1 + \frac{P(H_0)}{P(H_1)} (1 + \xi) \exp\left(-\hat{\gamma} \frac{\xi}{1 + \xi}\right) \right\}^{-1} \quad (9.8)$$

where the a priori probabilities for informative tonal region absence and informative sound presence are denoted $P(H_0)$ and $P(H_1)$, respectively. The noise power spectrum σ_n has to be estimated because it is practically unknown. A direct way of obtaining the estimate is by applying a temporal recursive smoothing to the noisy observation during sound absence periods only, such that

$$\begin{aligned} H_0 : \hat{\sigma}_n(k) &= \tau \hat{\sigma}_n(i-1) + (1-\tau) |Y|^2 \\ H_1 : \hat{\sigma}_n(k) &= \hat{\sigma}_n(i-1) \end{aligned} \quad (9.9)$$

where τ denotes the smoothing factor for the noise PSD estimate. Its range is $0 \leq \tau \leq 1$. By applying the a posteriori tonal region presence probability from Eq. (9.8) to Eq. (9.9), the recursive averaging becomes [93]

$$\hat{\sigma}_n(k) = P(H_1|y) \hat{\sigma}_n(k-1) + (1 - P(H_1|y)) \left[\tau \hat{\sigma}_n(k-1) + (1-\tau) |Y|^2 \right] \quad (9.10)$$

In this case, the main factor that updates the noise power estimate lies in the a priori estimation of ξ . $P(H_0)$ and $P(H_1)$ in Eq. (9.8), where $P(H_1) = 1 - P(H_0)$.

The proposed a posteriori tonal region presence probability in Eq. (9.9) requires an estimate of the a priori SNR, ξ , which tends to zero in sound absence, and is gradually augmenting with sound power spectrum. However, when $\xi = 0$, i. e., $\sigma_b = 0$, the likelihoods of tonal region presence and tonal region absence become identical [94].

In this paper, it is demonstrated that an adaptive sigmoid function can be exploited as a tonal region presence probability (TRPP) algorithm. This function offers the possibility to adjust the slope and the mean of the TRPP independently to accomplish a desired trade-off between noise overvaluation and undervaluation.

This sigmoid function, also called the sigmoidal curve or logistic function, is defined as [95].

$$S = \frac{1}{1 + \exp(-\alpha_{sgm}(\hat{\gamma} - \beta_{sgm})} \quad (9.11)$$

where α_{sgm} and β_{sgm} are the slope and the mean of the sigmoid function, respectively, given by

$$\alpha_{sgm} = \frac{\xi_{H_1}}{1 + \xi_{H_1}}, \beta_{sgm} = \log\left(\frac{P(H_0)}{P(H_1)}(1 + \xi_{H_1})\right) \frac{1 + \xi_{H_1}}{\xi_{H_1}} \quad (9.12)$$

ξ_{H_1} can be any value of $12 \text{ dB} \leq 10\log_{10}\xi_{H_1}$ when $P(H_0) = P(H_1)$. By substituting these values into the slope and the mean of the sigmoid function, one obtains $0.94 \leq \alpha_{sgm} \leq 0.98$ and $3 \leq \beta_{sgm} \leq 4.23$. This reflects that while the range of α_{sgm} and β_{sgm} is satisfied, any value of ξ_{H_1} , $P(H_0)$, and $P(H_1)$ can be used, provided that $10\log_{10}(\xi_{H_1}) \in [12 \text{ dB}, 18 \text{ dB}]$ and $P(H_0) = 1 - P(H_1)$. The slope and the mean can be controlled independently, from the moment that different values of ξ_{H_1} can be used such that ξ_α for α_{sgm} and ξ_β for β_{sgm} . As such, the a posteriori TRPP can be fully controlled to reach a requisite trade-off between noise overestimation and underestimation.

Rather than having a soft transition between the absolute tonal region absence and tonal region presence, we propose classifying the a posteriori TRPP into three zones, such as

$$\tilde{S} = \begin{cases} \text{less likely tonal region presence,} & S \leq s_1 \\ \text{more likely tonal region presence,} & s_1 < S \leq s_2 \\ \text{most likely tonal region presence,} & S > s_3 \end{cases} \quad (9.13)$$

where $0 < s_1 < s_2 < s_3 \leq 1$ with s_1, s_2, s_3 being diverse values of the sigmoid function. For the zone where the tonal region of the bird sound is less likely to be present, i.e., when $\hat{\gamma} \approx 0$, in case sound is active, S should be prevented from getting too close to zero. By doing so, the noise power spectral density (PSD) estimate yields an included weight of sum between the estimated noise PSD at earlier frames and the instance noisy observations. The result is an even smoothed estimate compared to the original noise PSD estimate when $\hat{\gamma}$ is small, which decreases the likelihood of noise being overestimated and under estimated locally. While for the areas where tonal region of bird sound is either more likely or most likely to be present, the soft transitions of S might not be sufficient for that the estimated noise power density (PSD) changes, based on the previous estimate of noise PSD, and tracks the current noisy observations and vice versa. Accordingly, to avoid these pitfalls, multilevel decisions are imposed on S to realize an improved a posteriori TRPP estimate. A solution for this is proposed as follows:

$$\tilde{S} = \begin{cases} \lambda_1, & S \leq 0.3 \\ \lambda_2, & 0.3 < S \leq 0.6 \\ \min\{\lambda_3, S\}, & S > 0.6 \end{cases} \quad (9.14)$$

where $\lambda_m = \exp(-2.2R)/(t_m F_s)$ indicates the exponential smoothing constant, with $m = [1, 2, 3]$. While R indicates the STFT frame rate, t_m represents the averaging time constant, with $t_1 < t_2 \ll t_3$. Note that rather than having a fixed smoothing constant, \tilde{S} is allocated with S for $S > 0.6$ to keep the noise PSD estimate as robust to sound onsets as possible. While the noise power estimate may cease to update when the noise level would make a sudden step from one sample to another, such that $\tilde{S} = 1$, the upper bound λ_3 is employed to prevent such stagnation in the estimate.

9.8.1.2 Audio parameterization and noise reduction: MFCC-SS integration

Feature extraction is a key element for any audio recognition system such as birdsong recognition. Indeed, this step produces the parameters on which the recognition system is based. To describe birdsong, generating a voiceprint of the audio signal, represented by a series of coefficients, is necessary. In sound processing, the Mel frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear Mel-scale of frequency. Due to computational efficiency and good performance in clean conditions [96], MFCCs are the most popular acoustic features extraction methods for their several advantages, including that the feature values are approximately decorrelated from each other, and the amount of information needed to describe a signal is reduced, for both periodic and aperiodic signals. Dimension reduction is advantageous for manual inspection of data and for use in systems that cannot cope with high-dimensional data. Perceptual systems of birds are not the same as in humans, but exhibit similar characteristics. Eq. (9.15) describes the mathematical relationship between the Mel scale and the linear frequency scale where f_{lin} is the linear frequency in Hz:

$$f_{mel} = 2595 \cdot \log_{10} \left(1 + \frac{f_{lin}}{700} \right) \quad (9.15)$$

The steps for computing the MFCC coefficients are as follows:

Step 1: Divide the bird sound signal into segments of 20 ms using overlapping smooth windows. A Hamming window is used:

$$ham(i) = 0.54 - 0.46 \cos \frac{2\pi i}{I} \quad (9.16)$$

where I is the length of the window.

Step 2: Take the Discrete Fourier Transform (DFT) $X(k)$ of the windowed time domain signal $x(i)$.

$$X(k) = \sum_{i=0}^{I-1} x(i) \times ham(i) \times \exp \left(\frac{-2j\pi ik}{I} \right), k = 0, 1, \dots, I-1, \quad (9.17)$$

Step 3: Calculate the square of the DFT of the windowed signal.

Step 4: The outputs of the fourth step are the Mel-scaled filter bank energies.

$$H_m(k) = \begin{cases} 0 & k < f_{m-1} \\ \frac{(k-f_{m-1})}{(f_m-f_{m-1})} & f_{m-1} \leq k \leq f_m \\ \frac{(f_{m+1}-k)}{(f_{m+1}-f_m)} & f_m \leq k \leq f_{m+1} \\ 0 & k > f_{m+1} \end{cases} \quad (9.18)$$

where m is the desired number of filters, and f is the list of $m+2$ Mel-spaced frequencies.

Step 5: Calculate the logarithm of the Mel-scaled filter bank energies.

Step 6: Take the Discrete Cosine Transform (DCT) of the Mel-scaled log-filter bank energies to determine MFCC.

As the MFCC algorithm has low noise robustness [43], a similar method able to decrease the effect of noise on voiceprints of waterbirds' sounds is proposed. It is entitled MFCC-SS for Mel frequency cepstral coefficients with spectral subtraction integration. By integrating the method of noise reduction (spectral subtraction) to that for characterization (MFCC), this will reduce the computational complexity and the execution time of our system, which makes this technique suitable for implementation in an embedded system (DSP). Fig. 9.10 shows the block diagram of the proposed algorithm.

Spectral subtraction (SS) [80] is a method for the magnitude spectrum restoration of a signal corrupted by an additive noise. This is achieved through the subtraction of an estimate of the average noise spectrum from the noisy signal spectrum. The noise spectrum is usually estimated, and updated, from the periods when the signal is absent and only the noise is present. Eq. (9.19) defines the result of spectral subtraction, where $X(k)$ and $N(k)$ are the magnitude spectrum of the detected noisy tonal region and the noise, respectively.

$$|\hat{D}(k)| = \begin{cases} |X(k)| - |N(k)| & \text{if } |X(k)| > |N(k)| \\ 0 & \text{otherwise} \end{cases} \quad (9.19)$$

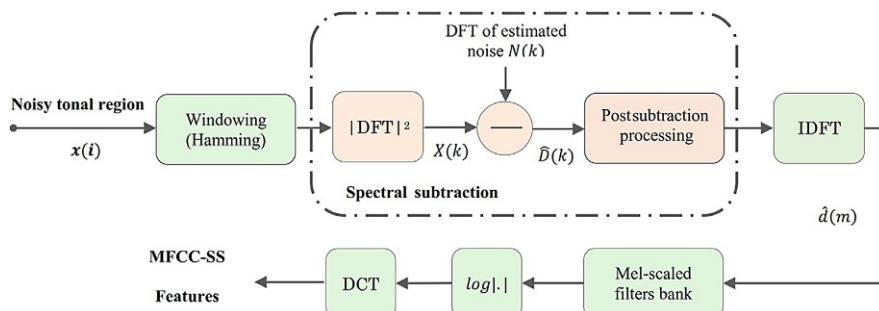


FIG. 9.10

Block diagram of MFCC with spectral subtraction integration (MFCC-SS) algorithm.

For restoration of time-domain signals, an estimate of the instantaneous magnitude spectrum $\hat{D}(k)$ is combined with the phase of the noisy signal $\theta_X(k)$, and then transformed via an inverse DFT to the time domain, as shown in Eq. (9.20). In terms of computational complexity, spectral subtraction is relatively inexpensive [80]:

$$\hat{d}(\omega) = \frac{1}{I} \sum_{k=0}^{I-1} |\hat{D}(k)| e^{j\theta_X(k)} e^{j\frac{2\pi}{I} k \omega}, \quad \omega = 0 \dots I-1 \quad (9.20)$$

9.8.1.3 Classification: SVM

In this step, voiceprints can be stored in a database or be employed for classification that aims to determine if the generated voiceprint belongs to a known class. Many classifier used in speech recognition were exploited in birdsong recognition. In our study, SVM [86] is exploited for classification. SVM is based on Vapnik-Chervonenkis (VC) dimension theory and structural risk minimization of the statistical learning theory. It is a very efficient classifier and shows unique advantages in solving the problems of the small samples, nonlinear and high dimensional pattern recognition. The basic principle is to isolate properly samples of two classes by finding the optimal separating hyperplane, and to maximize the minimum distance between the plus or minus class samples and the isolating hyperplane.

9.8.2 Acoustic bird recognition system: Proposition 2

In Ref. [97], the authors were also interested in monitoring and recognition bird species of Tonga Lake, using a WSNs. The WSN are supposed to capture, then process and transmit bird sound features. Generally, automatic recognition of bird sounds in WSN is based on three main phases: (1) TRD and feature extraction, (2) transmission of extracted features, and (3) classification. The first and second phases are performed at sensor nodes and the final phase at the server.

9.8.2.1 Audio parameterization: Gammatone Teager energy cepstral coefficients

In Ref. [97], the authors introduced a robust front-end that is motivated from auditory perception and uses a frequency dense bank of Gammatone filters (GTFs). The filter bandwidths are proportional to the auditory Equivalent Rectangular Bandwidth (ERB) function. The short-time average of the signal squared is commonly exploited as an ad hoc approximation of the energy of the signal's source. For resonance signals, the Teager-Kaiser Energy and the nonlinear energy operator Ψ provide a good estimation of the “real” source energy. In the following, we describe a front-end that combines the Gammatone auditory filter bank with the Teager energy estimation method.

a. Teager's energy operator

Newton's law of motion for an oscillator with mass m and spring constant k states that

$$\frac{d^2x}{dt^2} + \frac{k}{m}x = 0 \quad (9.21)$$

This second order differential equation describes an object with mass m suspended by a string with constant k . It is considered as a simple, but incomplete, model of a mechanical-acoustical system.

The solution to Eq. (9.21) is a signal given by $x(t) = a \cos(\phi(t))$. The system's total energy ERG is the sum of the kinetic and potential energy and is given by $ERG = 0.5 kx^2 + 0.5 m\dot{x}^2$. By substituting $\dot{x} = dx/dt$, and $x = a \cos(\omega t + \phi)$, we get

$$ERG = \frac{1}{2}ma\omega^2a^2 \quad (9.22)$$

where $\omega = d\phi(t)/dt$. Taking this analysis under consideration, Teager, and then [98], proposed the Teager-Kaiser Operator Ψ :

$$\Psi[x(t)] = \dot{x}^2(t) - x(t)\ddot{x}(t) \quad (9.23)$$

When applied to an AM-FM signal $x(t) = a(t) \cos(\phi(t))$, the Ψ operator yields

$$\Psi[x(t)] \cong a^2(t)\phi^2(t) \quad (9.24)$$

Herein, instead of using the conventional signal energy approximation of x^2 (which only takes into account the kinetic energy of the signal's source), we will use the Teager-Kaiser energy operator for computing the "true" source energy. The short-time average of the output of the energy operator will be used for feature estimation. The Teager-Kaiser estimated energy includes both amplitude and frequency information; the hope is that the additional information in the estimated energy can be translated into enhancement in bird sound recognition accuracy.

b. Gammatone auditory filter bank

In auditory modeling, the digital filter bank is one of the most fundamental notions that resemble the characteristics of the basilar membrane. In the inner ear's cochlea, each band-pass filter modeled response of part of the basilar membrane to some localized frequency information of the sound signals. Human auditory processing is based on a set of density frequency asymmetric filters used to estimate the activity of each frequency band. The bandwidth of asymmetrical filters is quantified using the notion of the Equivalent Rectangular Bandwidth (ERB). The Gammatone function that represents the impulse response of each filter has the following temporal form [99]:

$$g(t) = A t^{n-1} \exp(-2\pi r ERB(f_c)t) \cos(2\pi f_c t) \quad (9.25)$$

where the GTF design parameters are A , r , n and the center frequency of the filter is f_c . In the GTF bank, the bandwidth of each filter is established to the auditory critical band related to its center frequency. In particular, the filter's ERB is defined in "Hz" as in [100], and this is when we specified the magnitude of a filter's frequency response $|H(f)|$ and the filter's maximum gain $|H(f_{max})|$ at the frequency f_{max} :

$$ERB = \frac{\int |H(f)|^2}{|H(f_{max})|^2} \quad (9.26)$$

The ERB is the equivalent bandwidth of an orthogonal filter with constant gain $|H(f_{max})|$ and energy equal to the original filter's energy. The filter's energy is defined as the integral of the filter's frequency response squared. Based on the human physiology states, it is revealed in the current research [101] that the auditory filter bandwidths are given by the following $ERB(f)$ function:

$$ERB(f) = 6.23 \left(\frac{f}{1000} \right)^2 + 93.39 \left(\frac{f}{1000} \right) + 28.52 \quad (9.27)$$

where f is the filter center frequency expressed in “Hz.” Using the critical Bark frequency scale, the filter insertion is equidistant as follows:

$$Bark(f) = \frac{26.81f}{f + 3920} - 0.53 \quad (9.28)$$

where $0 \leq f \leq F_s/2$ and F_s is the sampling frequency of the signal.

Davis and Mermelste [102] have proposed that the auditory filters should have $r = 1.019$ and $n = 4$. Thus, the filter frequency response $G(\omega)$ is given by

$$G(\omega) = \frac{A}{2} \frac{6}{(2\pi r ERB(f_c) + j(\omega - \omega_c))^4} + \frac{A}{2} \frac{6}{(2\pi r ERB(f_c) + j(\omega + \omega_c))^4} \quad (9.29)$$

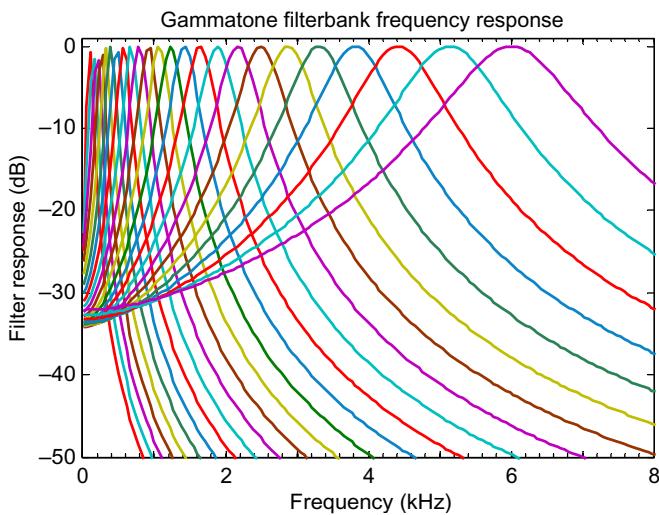
Moreover, the filter gain A is set taking under consideration that $|G(\omega_c)| = 1$ and is equal to

$$A = \frac{1}{\sum_{k=1}^N t^{n-1} \exp(-2\pi r ERB(f_c)t)} \quad (9.30)$$

where N is the length of the discretized impulse response in samples.

The GTF bank presented above, with filters placed according to the bark scale and with bandwidths given by the $ERB(f)$, is a good approximation of the human auditory system [103]. Fig. 9.11 shows an example of the GTF bank with 24 filters. Center frequencies are uniformly spaced on ERB scale in [100–6000] Hz range. The sample-rate is set at 22,050 Hz.

It can be seen that one of the major differences between Gammatone Teager energy cepstral coefficient (GTECC) and MFCC is the set of filters used in the extraction. In traditional MFCC, triangular filters equally spaced in the Mel scale frequency axis are used, while in TECC, Gammatone filters are used. Another main difference with GTECC is that the sound energy is estimated through Teager-Kaiser energy operator (TEO) on GTF output, instead of using DFT.

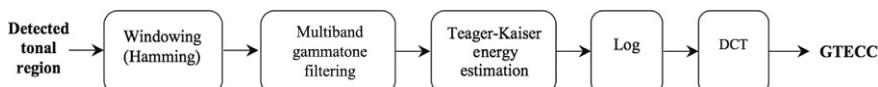
**FIG. 9.11**

Frequency response of the 24-channel Gammatone filter bank.

c. Steps for GTECCs extraction

The processing steps for the GTECCs front-end are as follows (Fig. 9.12):

- The bird sound is first preaccented with a preemphasis filter equal to 1 to increase the amplitude of high frequencies in birdsong in the time domain. The equation of preemphasis filter applied is $H(z) = 1 - 0.97z^{-1}$.
- The preemphasized bird sound signal is then transformed into the short-time Fourier transform (STFT) domain using a Hamming window of 25.6 ms duration with a skip rate of 10 ms.
- Next, as defined in Eqs. (9.25), (9.29), a family of GTF bank with 24 filters is used to band pass the bird sound signal. The filter spacing is linear in the Bark scale.
- Then, the logarithm of the short-time average of the Teager-Kaiser energy operator for each one of the band-passed signals is evaluated. The short-time averaging window period and window shift are the same as for the classic MFCC front-end.
- Lastly, the discrete cosine transform (DCT) is applied to obtain the cepstral coefficients of the short-time average Teager energy. Finally, the cepstral coefficients are truncated to keep the first 13 coefficients (including the zeroth coefficient) to obtain the GTECC feature vectors.

**FIG. 9.12**

Schematic block diagram of the GTECC front-end.

d. Quantile-based cepstral dynamics normalization for noise reduction

The presence of additive noise directly affects cepstral coefficients extracted from audio signal, and may cause a severe mismatch between cepstral distributions of the processed sound signal and those captured in acoustic models for audio recognition. Recently, advanced techniques normalizing the fine contours of cepstral histograms have been developed, utilizing either a rather extensive adaptation data sets matching the test conditions [104], or quantile-based online normalization applying two-pass search and continuity criteria [105]. In contrast to these complex methods, the goal of the cepstral compensation proposed here is exclusively to address the dynamic range mismatch in cepstral samples introduced by background noise and channel, extending the concepts of the popular and computationally inexpensive normalizations of spectral subtraction [105], cepstral mean and variance (CMVN) [106], and recently introduced cepstral gain normalization (CGN) [107].

Quantile-based cepstral dynamics normalization (QCN) [108] extends the concept of cepstral mean-variance normalization (CMVN) and cepstral gain normalization (CGN). The dynamic range of cepstral sample occurrence is valued from histogram quantiles, and subsequently the samples are normalized to a zero interquantile mean and unit interquantile distance. This method is motivated by the observation that mean and variance normalization may not be as efficient in aligning training and test distributions of different skewness (caused by signal distortions), while normalizing distributions by their selected low and high quantiles can assure good dynamic range alignment for two distributions of any shape. Unlike CMVN, QCN does not make any assumptions about the distribution properties, and instead performs an alignment of the sample dynamic ranges estimated from distribution quantiles. In previous studies, QCN provided superior performance gains in speech recognition under noise and Lombard effect [109] and reverberation [110] compared to other popular normalizations.

9.8.2.2 Classification

A deep neural network (DNN) is a multilayer perceptron with many hidden layers with weights initialized through the pretraining algorithm [111, 112]. DNNs were implemented by treating each pair of layers in DNNs as a restricted Boltzmann machine (RBM) [113] before performing a joint optimization of all the layers. A bipartite graph with a visible layer and a hidden layer can be used to represent an RBM. The stochastic units in the visible layer only connect to the stochastic units in the hidden layer. The units in the visible layer are typically represented by Bernoulli or Gaussian distributions, and the units in the hidden layer are commonly represented with Bernoulli distributions. Gaussian-Bernoulli RBMs (GBRBM) can convert real-valued stochastic variables (such as short-term spectral features) to binary stochastic variables that can then be further processed using the Bernoulli-Bernoulli RBMs (BBRBMs).

9.8.3 Alternative acoustic features used in birdsong/speech recognition systems

9.8.3.1 Perceptual MVDR-based cepstral coefficients

Numerous studies have considered combining the merits of Minimum Variance Distortion less Response (MVDR) spectrum into the speech recognition framework [114–116]. The first use of MVDR in speech parameterization was for power spectrum estimation [116]. In [116], the FFT spectrum in the MFCC computation method was simply replaced by a high-order MVDR spectrum. The remainder of the feature extraction algorithm was the same as the MFCC front-end, therefore these features are called MVDR-based MFCCs (MVDR-MFCCs). In addition, Dharanipragada and Rao [114] incorporated a second step for cepstral smoothing to reduce the variances of the feature vectors, which was also shown to be useful. One obvious disadvantage is the high computational burden imposed by the high-order MVDR spectrum computation method and cepstral averaging.

A second study employing the MVDR methodology for feature extraction in Refs. [115, 116]. The features developed in Refs. [115, 116] are called Perceptual MVDR-based Cepstral Coefficients (PMCCs). In the PMCC front-end (Fig. 9.13), the MVDR methodology is used for spectral envelope extraction rather than for spectrum estimation. It was shown that using the MVDR for spectral envelope extraction is more successful and yields better accuracy on the IBM in-car speech recognition task [115] and the Wall Street Journal (WSJ) task [116]. The implementation of PMCCs is very similar to PLP in that they both represent the spectral envelope using an all-pole model. However, the use of the MVDR-based and not the LP-based envelope in the all-pole modeling stage provides a measurable difference in performance for real car noise conditions [115].

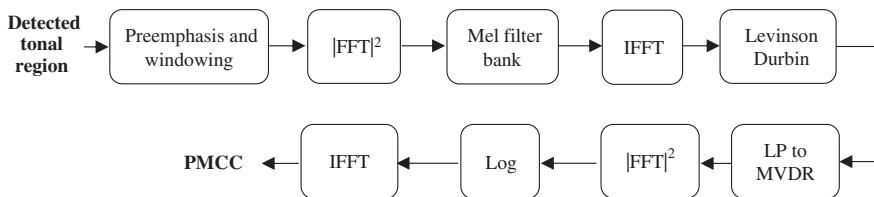


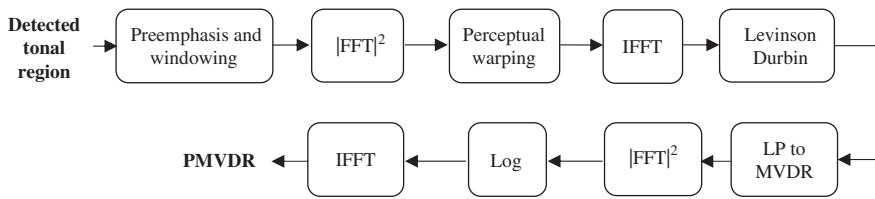
FIG. 9.13

Schematic block diagram of the PMCC front-end.

9.8.3.2 Perceptual-MVDR

Yapanen and Hansen [117] proposed the Perceptual-MVDR front-end. It is obtained by incorporating perceptual warping along with the MVDR (Fig. 9.14). The Perceptual-MVDR (PMVDR) feature [117] has been described as a better alternative to MFCC for ASR under noisy conditions.

The preemphasized sound signal is first transformed into the short-time Fourier transform domain using Hamming windows. Next, the power spectrum is calculated

**FIG. 9.14**

Schematic block diagram of the PMVDR front-end.

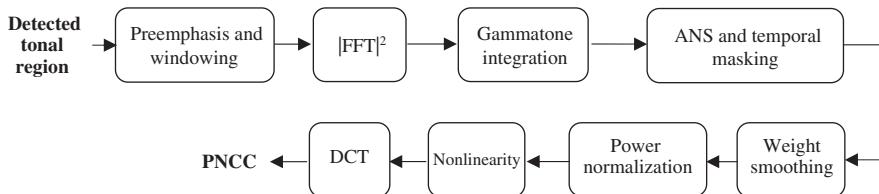
and directly warped towards a perceptual scale through a first order all-pass filter. The phase response of the first order filter is given as

$$\omega = \tan^{-1} \frac{(1 - \sigma^2) \sin(\hat{\omega})}{(1 + \sigma^2) \cos(\hat{\omega}) + 2\alpha} \quad (9.31)$$

where $\hat{\omega}$ represents the linear frequency, while ω represents the warped frequency. The value of σ controls the degree of warping, which is set to 0.31 to approximate the Mel scale with sampling rate $F = 8 \text{ kHz}$.

9.8.3.3 Power-normalized cepstral coefficients

The power-normalized cepstral coefficients (PNCCs) algorithm, proposed by Kim and Stern [118], is based on auditory processing (Fig. 9.15). Major new features of PNCC processing include the use of a power-law nonlinearity that replaces the traditional log nonlinearity used for MFCC coefficients, and a novel algorithm that suppresses background excitation by estimating SNR based on the ratio of the arithmetic to geometric mean power, and subtracts the inferred background power. This normalization makes use of the ratio of the arithmetic mean to the geometric mean, which has proved to be a useful measure in determining the extent to which sound is corrupted by noise [119]. The computational cost of PNCC is only slightly greater than that of conventional MFCC processing. In addition, PNCC uses frequency weighting based on the Gammatone filter shape [120] rather than the triangular frequency weighting associated with the MFCC computation.

**FIG. 9.15**

Schematic block diagram of the PNCC front-end.

9.8.3.4 Mel-scaled wavelet packet decomposition subband cepstral coefficient

In the process of MFCC extraction, the Fourier transform is used to convert the sound signal in time domain into magnitude spectrum, extracting just the statistical information of signal through the integral of the time domain, which cannot well reflect the local changes of nonstationary signal. Wavelet transform can better reflect the local changes of signal information, because its sampling step length for different frequency in time domain is adaptive, low resolution for low frequency, high resolution for high frequency, which accords with the characteristic of slow change at low frequency and rapid change at high frequency. It is suitable for the analysis of bird sound signal, which is famous for its randomness, diversity, and nonstationarity. In Ref. [121], the feature of Mel-scaled Wavelet packet decomposition Subband Cepstral Coefficient (MWSCC) was extracted from the bird sound signals. The feature MWSCC used wavelet packet decomposition (WPD) spaced in Mel scale to divide frequency band, instead of short-time Fourier transform, so as to improve the recognition performance of bird sound signal. The extraction procedure of MWSCC is similar to MFCC, except for several steps (Fig. 9.16).

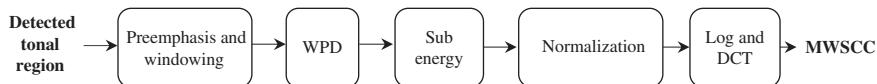


FIG. 9.16

Schematic block diagram of the MWSCC front-end.

9.9 Experimental evaluations

9.9.1 Data description (database) and experimental setup

For experimental results, the audio data were downloaded from the Xeno-canto archive (www.xeno-canto.org). We made use of a subset consisting of 36 bird species that are present in Tonga Lake (northeast Algeria). Note that these are field recordings and each file potentially contains vocalizations of several animal species and competing noise caused by wind, rain, and anthropogenic interference.

Bird species with multiple call and song examples recorded in Europe and Africa were used. The audio for each example were downloaded and changed to a monophonic wave file. The sampling frequencies were all converted to 44,100Hz with 16 bits for each sample. Seventy-two audio recordings with an average length of 28.9 s were available for each of the 36 species. We used 50 recordings for the model creation and 22 recordings for the purpose of performance evaluation. In total, there were 1800 files for model creation and 792 files for performance evaluation. For clarity reasons, the configuration parameters used for the feature extraction methods considered in this study are resumed in Table 9.9.

Table 9.9 Configuration parameters for GTECC as well as other parameterization strategies considered in our study.

Parameters	Features extraction methods					
	GTECC	MFCC	PMVDR	PMCC	PNCC	MWSCC
Preemphasis	Yes	Yes	Yes	Yes	Yes	Yes
Windowing	Hamming	Hamming	Hamming	Hamming	Hamming	Hamming
Frame length	25.6ms	25.6ms	25.6ms	25.6ms	25.6ms	25.6ms
Frame shift	10ms	10ms	10ms	10ms	10ms	10ms
WPD	N/A	N/A	N/A	N/A	N/A	Yes
Filter bank	Gammatone	Triangular	N/A	Triangular	Gammatone	N/A
Freq. scale	ERB	Mel	Perceptual Warping	Mel	ERB	Mel
No. bands	24	24	N/A	24	24	24
Coefficients	$13 + \Delta + \Delta\Delta$					
Compression	Logarithm	Logarithm	Logarithm	Logarithm	Power-law nonlinearity	Logarithm
Norm.	SS, CMVN, CGN, QCN					

9.9.2 Results and discussion

In the following, we analyze the performance of the bird recognition system proposed in this chapter on the basis of three main criteria, which are:

- the identification rate of bird species;
- the robustness to environmental noise; and
- the computational complexity.

For the purpose of verifying the effectiveness of our approach GTECC, we compared it to five popular strategies. We made several experiments at different signal-to-noise ratios (SNRs) under diverse noisy environments.

9.9.2.1 Segmentation and recognition performance

To examine the performance and efficiency of the proposed segmentation method, namely TRD, we took the sound of the Ferruginous Duck with a duration just over 6s with a sampling rate of 44.1 kHz, which gives a total number of samples of 265,250. Its time domain and spectrogram graphs are shown in Fig. 9.17A. By adding an environment noise, which leads to a SNR = 10 dB, we can get the time domain and spectrogram graphs of noisy sound shown in Fig. 9.17B. Fig. 9.17C represents the time domain graph and spectrogram of the noisy bird sound after applying the proposed tonal region detector (TRD). The samples are reducing from 265,250 down to 44,109, which represent approximately one-quarter of the total number of samples.

The recognition performance for different front-end/normalization setups found per each feature extraction strategy (GTECC, MFCC, PMVDR, PNCC, and MWSCC) is

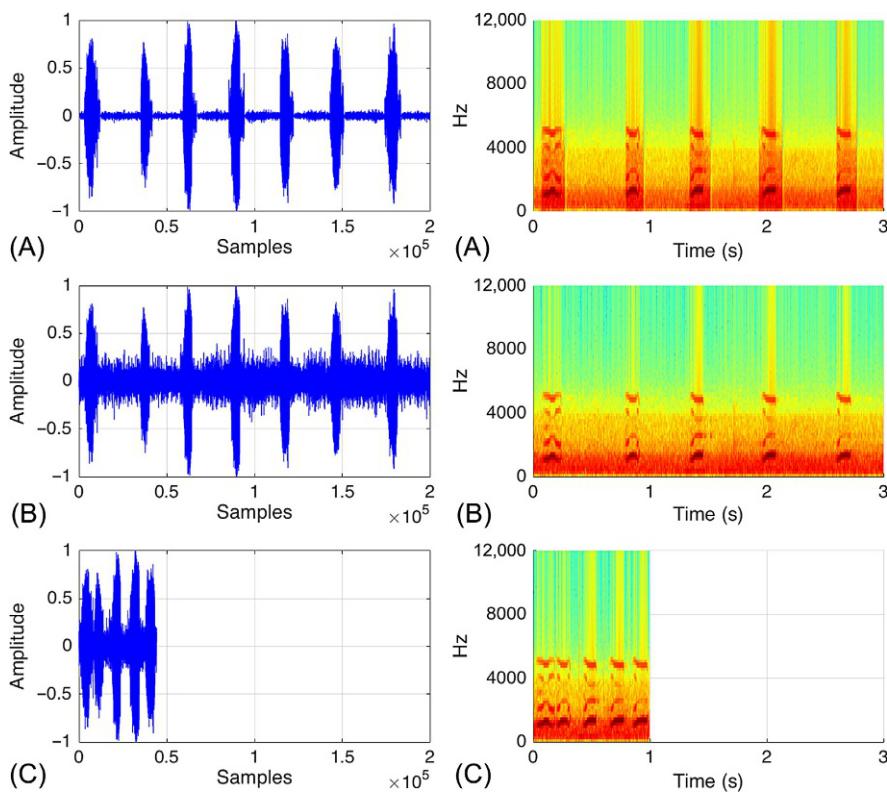
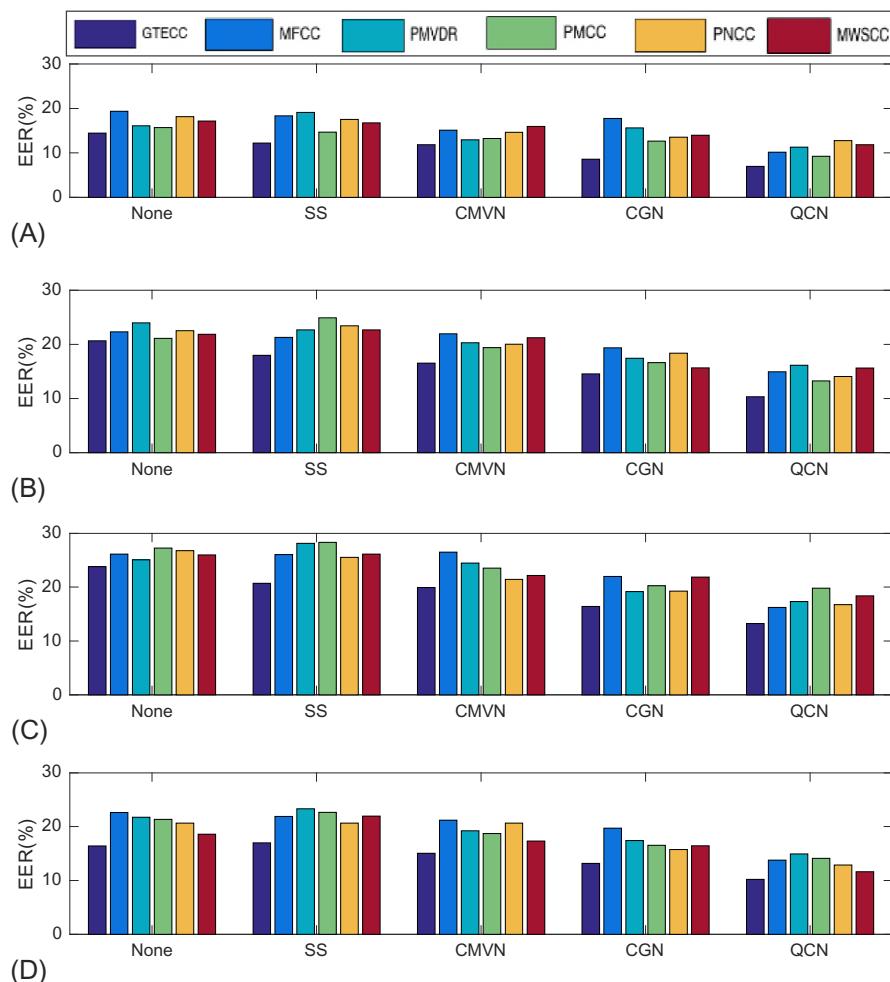


FIG. 9.17

The time domain graph of selected Ferruginous Duck sound in the left and his spectrogram in the right (A) in clean condition, (B) perturbed by environment noise at $\text{SNR} = 10\text{dB}$, and (C) after applying the tonal region detector (TRD).

given in Fig. 9.18. The results are reported in terms of EER (equal error rates). The EER refers to that point in a DET (Detection Error Tradeoff) curve where the FAR (False Acceptance rate) equals the FRR (False Rejection Rate). A perfect scoring model would yield an EER of zero, so a lower EER value indicates a better performance.

The training sounds data were corrupted by an additive white Gaussian noise (AWGN) with a SNR equal to 50dB. However, the testing data were corrupted by an AWGN with a SNR that varies from -10 to 30dB in 5-dB steps, in order to simulate the real-world complex noisy environments. From Fig. 9.18, we can see that the proposed GTECC front-end is almost in all cases (whether in clean or noisy conditions) the most efficient in terms of EER, with or without application of a method of normalization. Moreover, GTECC postprocessed by Quantile-based cepstral dynamics normalization (GTECC-QCN) gives an average EER equal to 10.18%, which is the lowest EER compared to the other studied methods, followed

**FIG. 9.18**

Performance comparison of alternative popular features considered in this study against GTECC in terms of EERs (%) for different values of SNR: (A) SNR = 50dB, (B) SNR varies from 30 to 0dB, (C) SNR varies from 0 to -10dB, and (D) the average values.

by MWSCC-QCN with an average EER of 11.61% and PNCC-QCN with an average EER equal to 12.87%, then MFCC-QCN with an average EER equal to 13.78%, then PMCC-QCN with an average EER equal to 14.11%, and finally PMVDR-QCN with an average EER equal to 14.91%.

Let us consider now the performance of normalization methods used in our study for the reduction of noise in the detected segments as tonal regions in the sound of a bird. From Fig. 9.18, we can observe that spectral subtraction (SS) is

less effective for reducing white Gaussian noise, especially with a SNR <0dB. Normalizations CMVN and CGN have relatively similar results on the Gaussian white noise with a significantly superior performance of the normalization CGN, when applied to PMVDR, in the presence of AWGN with a SNR between 0 and -10dB. Finally, QCN has the best performance when applied to the six methods of characterization studied namely GTECC, MFCC, PMVDR, PMCC, PNCC, and MWSCC.

9.9.2.2 Noise robustness analysis

Fig. 9.19 displays the average recognition rate of bird species disturbed by three different real-word noises (wind as continuous noise, takeoff of an airplane as intermittent noise, and thunder as impulsive noise) at global SNRs of “30 to -10” dB in steps of 5 dB. Strategies GTECC, MFCC, PMCC, PMVDR, PNCC, and MWSCC for feature extraction (with QCN for normalization) (a) before and

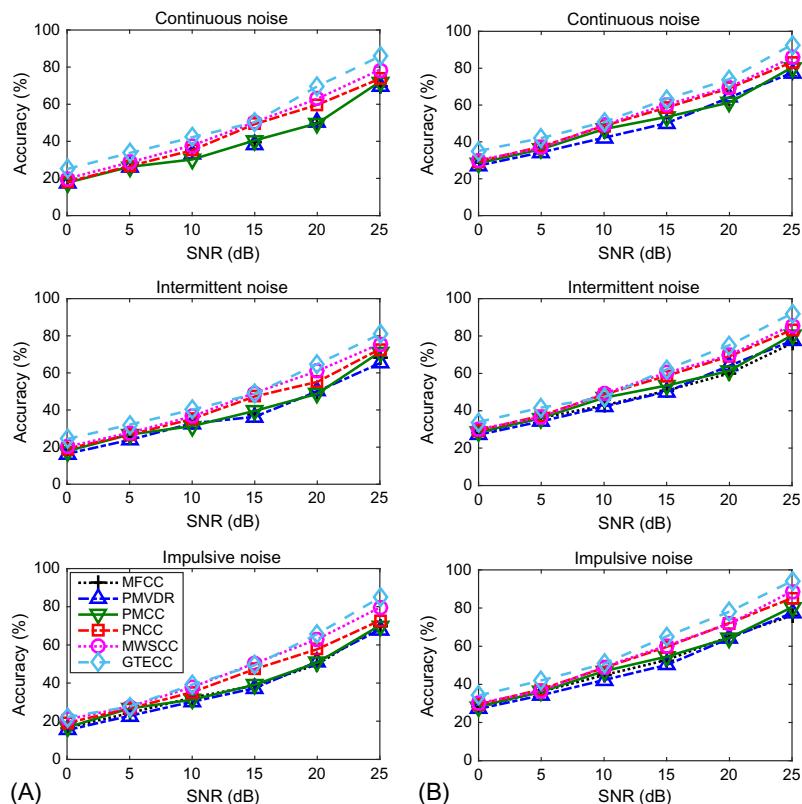


FIG. 9.19

Robustness of the feature extraction strategies to environmental noises (A) the accuracy evolution of studied strategies before applying TRD approach and (B) after applying TRD.

(b) after applying the proposed TRD approach are compared. As seen from Fig. 9.19, as the SNR decreases, the overall average identification accuracies of the six features decrease under environment noises, but the dropping trend is obviously different. The accuracy rate trend in Fig. 9.19A falls sharply, while the trend of Fig. 9.19B falls slowly.

All the accuracies reach higher than 80% at high SNR of 30 dB, but when SNR is low to -10 dB, the all recognition accuracies fall to the lowest. The average lowest accuracy of front-end falls to 12.2%, while when applying TRD it remains 24.89%, superior to classic front-end. From the above, the proposed TRD approach has good antinoise function, especially a better robustness for the low SNR, and is suitable for the bird sound identification under complex environments.

9.9.2.3 Computational performance

Table 9.9 shows estimates of the computational demands of GTECC, MFCC, PMVDR, PMCC, PNCC, and MWSCC feature extraction front-ends. We use the standard open source from auditory toolbox [122] for the implementation of MFCC and the implementation for PNCC (Open Source MATLAB can be found at Ref. [123]). The used parameters are set as follows:

- The window length is 25.6 ms.
- The interval between successive windows is 10 ms.
- The sampling rate is assumed to be 16 kHz.
- FFT with 512-pt for each analysis frame is used.

Specifically, the triangular weighting used in the MFCC and PMVDR calculation encompasses a narrower range of frequencies, which is considerably narrower than the GTF shapes, and the amount of computation needed for spectral integration is directly proportional to the effective bandwidth of the channels.

From Table 9.10, we notice that the PMVDR is the most demanding method in terms of computational cost with $2 \times (512 \times \log_2 512 + 512) = 2 \times 5120$ operations for FFT and IFFT steps only. This constitutes about 70% of the total number of operations required for the calculation of the PMVDR coefficients and almost twice the number of operations required for the calculation of the proposed GTECC coefficients. Moreover, the calculation of perceptual warping step requires $4 \times 257 = 1028$ operations. PNCC front-end comes in second place, followed by MWSCC then PMCC, MFCC, and finally GTECC, which has the lowest computational complexity.

The reason that GTECC front-end is less complex compared with the other studied methods is related to the use of the energy estimate with 256 operations instead of the FFT with 5120 operations that is an essential step for the three other methods. Concerning the proposed TRD approach, it exhibits a computational complexity lower than the computational complexity of the minimum statistics (MS) estimator [124] as the exponential function is the only special function that has to be computed.

Table 9.10 Computational complexity for different front-ends based on a 400-sample window at 16 kHz and a 50% overlap between consecutive frames.

Step	# Operations					
	GTECC	MFCC	PMVDR	PMCC	PNCC	MWSCC
Windowing	400	400	400	400	400	400
Energy estimation	256	–	–	–	–	–
FFT, $n = 512$	–	5120	5120	5120	5120	5120
Perceptual warping	–	–	1028	–	–	–
IFFT, $n = 512$	–	–	5120	–	–	–
Filter bank, $b = 24$	4984	514	–	514	4984	–
Medium time power norm.	–	–	–	–	40	–
ANS with temporal masking	–	–	–	–	320	–
Weight averaging	–	–	–	–	120	–
MVDR, $m = 22$	–	–	968	968	–	–
Sub energy +WPD	–	–	–	–	–	4984
FFT, $n = 128$	–	–	896	896	–	–
DCT	312	312	–	–	312	312
IFFT, $n = 128$	–	–	896	896	–	–
Total Ops.	5952	6346	14,428	8794	11,296	10,816

9.10 Conclusion

In this chapter, we proposed a methodology for acoustic monitoring of birds in natural environments. The main objective is to develop a system, using a WSN, for monitoring and identifying different bird species of the northeastern Algeria wetlands. This system must be able to gather as much information as possible to help put in place a protection plan for the most vulnerable bird species such as the White-headed Duck and the Nyroca Duck. The implementation of this surveillance system raises several challenges. Firstly, recordings collected in a natural environment are subject to different disturbances due mainly to environmental noises and/or possible capture of several bird

sounds, of the same species or not, by the same recorder. Secondly, the WSNs have many limitations, such as the relative weakness of their processing unit, the small size of the memory, communication problems, and especially the low lifetime of their batteries. Nevertheless, other constraints must also be solved, such as the distance between the bird position and the sensor node, which can attenuate the captured signal power, and consequently affects the performance of the recognition system.

In general, the robustness of an automatic bird recognition technique depends essentially on the step of extracting parameters. For this purpose, the effectiveness of the TRD-GTECC feature was confirmed, and it was revealed to outperform the feature extraction methods MFCCs as well as MWSCC, PMCC, PMVDR, and PNCC front-ends, especially in complex environments. Moreover, the computational complexity of the TRD-GTECC is less costly than the other studied audio parameterization methods, which makes it suitable for implementation on wireless sensor nodes.

The proposed systems for automatic recognition of birds in their natural habitat based on WSN can be generalized to other bird species and even to other types of animals.

References

- [1] J.F. Clements, The Clements Checklist of Birds of the World, sixth ed., Cornell University Press, 2007, p. 2007.
- [2] J.F. Clements, T.S. Schulenberg, M.J. Iliff, S.M. Billerman, T.A. Fredericks, B.L. Sullivan, C.L. Wood, The eBird/Clements Checklist of Birds of the World: v2019, Downloaded from:<https://www.birds.cornell.edu/clementschecklist/download/>, 2019.
- [3] G.F. Barrowclough, J. Cracraft, J. Klicka, R.M. Zink, How many kinds of birds are there and why does it matter? *PLoS One* 11 (11) (2016) e0166307. Editor Andy J Green, Consejo Superior de Investigaciones Científicas, Spain. November 23.
- [4] K. Snyder, New Study Doubles the Estimate of Bird Species in the World, Department of Communications, American Museum of Natural History, 2016.
- [5] http://www.iucnredlist.org/static/categories_criteria_3_1.
- [6] C. Bibby, M. Jones, S. Marsden, *Expedition Field Techniques: Bird Surveys*, BirdLife International, 2000. March.
- [7] F. Archaux, On methods of biodiversity data collection and monitoring, *Revue Science Eaux & Territoires* (2011). Public policy and biodiversity, numéro 03bis, pp. 70–75, 15/03/2011, <http://www.set-revue.fr/methods-biodiversity-data-collection-and-monitoring>.
- [8] <http://datazone.birdlife.org/country/algeria>.
- [9] M. Boulkhssaim, M. Houhandi, B. Samraoui, Status and diurnal behaviour of the Shelduck *Tadorna tadorna* in the Hauts Plateaux, Northeast Algeria, *Wildfowl* 56 (56) (2013) 65–78.
- [10] B. Samraoui, F. Samraoui, An ornithological survey of Algerian wetlands: Important bird areas, Ramsar sites and threatened species', *Wildfowl* 58 (2008) 71–96.
- [11] http://datazone.birdlife.org/userfiles/file/IBAs/Ramsar/IBAs_Ramsar_Africa/Algeria.pdf.
- [12] <http://datazone.birdlife.org/userfiles/file/IBAs/AfricaCntryPDFs/Algeria.pdf>.

- [13] <http://www.unesco.org/new/en/natural-sciences/environment/ecological-sciences/biosphere-reserves/africa/algeria/el-kala/>.
- [14] <http://www.unesco.org/mabdb/br/brdir/directory/biores.asp?mode=all&code=ALG+02>.
- [15] S. Djamel, D. Yamna, A. Djamel, Biological diversity of the National Park of El-Kala (Algeria), valorization and protection, *Biodivers. J.* 5 (4) (2014) 525–532.
- [16] A. LazLi, I. Nouari, N. Chater, A. MoaLi, Diurnal behaviour of breeding White-headed Duck *Oxyura leucocephala* at Lake Tonga, North-East Algeria, *Rev. Ecol.* 69 (2) (2014) 131–141.
- [17] A. Boumezbeur, Ecologie et biologie de la reproduction de l'Érismature à tête blanche *Oxyura leucocephala* et du Fuligule nyroca *Aythya nyroca* sur lac Tonga et le lac des oiseaux (Est Algérien) Mesures de protection et de gestion du lac Tonga, (Ph.D. thesis), EPHE, Montpellier, 1993.
- [18] H. Chenchouni, Statuts de protection et de conservation des oiseaux recensés dans les Aurès et ses alentours (nord-est Algérien), (2010).
- [19] P. Isenmann, A. Moali, Oiseaux d'Algérie, SEOF, Paris, 2000 (336 p).
- [20] S. Metallaoui, et al., Hivernage de l'Érismature à tête blanche (*Oxyura leucocephala*) dans Garaet Hadj-Tahar (Skikda, nord-est de l'Algérie), *AVES* 46 (3) (2009) 136–140.
- [21] J.A.T. Esquivias, La Malvasía cabeciblanca (*Oxyura leucocephala*) durante los primeros años del siglo XXI, *Oxyura: Revista sobre las zonas húmedas* 12 (1) (2009) 87–116.
- [22] H. Azafzaf, The Ferruginous Duck in Tunisia, in: Ferruginous Duck: From Research to Conservation, International Meeting Proceedings, 2002, pp. 11–14.
- [23] J.S. Kirby, A.J. Stattersfield, S.H. Butchart, M.I. Evans, R.F. Grimmett, V.R. Jones, I. Newton, Key conservation issues for migratory land-and waterbird species on the world's major flyways, *Bird Conserv. Int.* 18 (S1) (2008) S49–S73.
- [24] M.I. Sánchez, A.J. Green, J.C. Dolz, The diets of the white-headed duck *Oxyura leucocephala*, Ruddy Duck *O. jamaicensis* and their hybrids from Spain, *Bird Study* 47 (3) (2000) 275–284.
- [25] H. Heim De Balsac, N. Mayaud, Les oiseaux du Nord-Ouest de l'Afrique, Lechevalier, Paris, 1962.
- [26] G. van Dijk, J.P. Ledant, La Valeur Ornithologique des zones humides de l'est Algérien, *Biol. Conserv.* 26 (3) (1983) 215–226.
- [27] A. Lazli, A. Boumezbeur, N. Moali-Grine, A. Moali, Évolution de la population nicheuse de l'Érismature à tête blanche *Oxyura leucocephala* sur le lac Tonga (Algérie), (2011).
- [28] F. Chettibi, et al., Diurnal activity budget and breeding ecology of the white-headed duck *Oxyura leucocephala* at Lake Tonga (north-east Algeria), *Zool. Ecol.* 23 (3) (2013) 183–190.
- [29] F. Chettibi, Ecologie de l'Érismature à tête blanche *Oxyura leucocephala* dans les zones humides de la Numidie algérienne (du Littoral Est de l'Algérie) (Ph.D. thesis), Annaba University, 2014.
- [30] <http://www.iucnredlist.org/details/22679814/0>.
- [31] J.A. Robinson, B. Hughes, Ferruginous Duck Action plan published. WWT, TWSG news. The bulletin of the threatened waterfowl specialist Groupe.N0.15, 2006.
- [32] N. Baaziz, B. Mayache, M. Saheb, E. Bensaci, M. Ounissi, S. Metallaoui, M. Houhamdi, Statut phénologique et reproduction des peuplements d'oiseaux d'eau dans l'éco-complexe de zones humides de Sétif (Hauts plateaux, Est de l'Algérie), *Bulletin de l'Institut Scientifique, Rabat, section Sciences de la Vie* 33 (2) (2011) 77–87.
- [33] <http://www.iucnredlist.org/details/22680373/0>.

- [34] S.E. Merzoug, W. Amor Abda, M. Belhamra, M. Houhamdi, Eco-ethology of the wintering ferruginous duck *Aythya nyroca* (Anatidae) in Garaet Hadj Tahar (Guerbes-Sanhadja, northeast of Algeria), Zool. Ecol. 24 (4) (2014) 297–304.
- [35] J.A. Robinson, B. Hughes, Ferruginous Duck action plan published. WWT, TWSG news. The bulletin of the threatened waterfowl specialist Groupe.N°.15, 2006.
- [36] M. Houhamdi, B. Samraoui, Diurnal and nocturnal behaviour of ferruginous duck *Aythya nyroca* at lac des Oiseaux, northern Algeria, Ardeola 55 (1) (2008) 59–69.
- [37] K. Mullarney, L. Svensson, D. Zetterström, J. Peter, J. Grant, Guide ornitho, les 848 espèces d'Europe en 4000 dessins, Delachaux et Nestlé, 2007 (399 p).
- [38] S. Cramp, K.E.L. Simmons, The Birds of the Western Palearctic, vol. 1, Oxford University Press, Oxford, 1977.
- [39] V.J. Lysenko, Fauna of the Ukraine: Birds, 5: Anseriformes, Naukova Dumaka (en russe), Kiev, 1992.
- [40] R. Aissaoui, et al., Diurnal behaviour of Ferruginous Duck *Aythya nyroca* wintering at the El-Kala wetlands (northeast Algeria), Bulletin de l'Institut Scientifique, Rabat, section Sciences de la Vie 33 (2) (2011) 67–75.
- [41] A.L. McIlraith, H.C. Card, Bird song identification using artificial neural networks and statistical analysis, in: IEEE 1997 Canadian Conference on Electrical and Computer Engineering, 1997. Engineering Innovation: Voyage of Discovery. Vol. 1, IEEE, 1997, pp. 63–66.
- [42] R. Bardeli, D. Wolff, F. Kurth, M. Koch, K.H. Tauchert, K.H. Frommolt, Detecting bird sounds in a complex acoustic environment and application to bioacoustic monitoring, Pattern Recogn. Lett. 31 (12) (2010) 1524–1534.
- [43] P. Jančovič, M. Köküer, Automatic detection and recognition of tonal bird sounds in noisy environments, EURASIP J. Adv. Signal Process. 2011 (1) (2011) 982936.
- [44] A. Patti, G. Williamson, Methods for classification of nocturnal migratory bird vocalizations using Pseudo Wigner-Ville Transform, in: Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, IEEE, 2013, pp. 758–762.
- [45] C.F. Juang, T.M. Chen, Birdsong recognition using prediction-based recurrent neural fuzzy networks, Neurocomputing 71 (1–3) (2007) 121–130.
- [46] F. Briggs, R. Raich, X.Z. Fern, Audio classification of bird species: a statistical manifold approach, in: 2009 Ninth IEEE International Conference on Data Mining, IEEE, 2009, pp. 51–60.
- [47] A. Selin, J. Turunen, J.T. Tanttu, Wavelets in recognition of bird sounds, EURASIP J. Adv. Signal Process. 2007 (1) (2006) 051806.
- [48] P. Somervuo, A. Harma, Bird song recognition based on syllable pair histograms, in: Acoustics, Speech, and Signal Processing, 2004, Proceedings. (ICASSP'04). IEEE International Conference on, vol. 5, IEEE, 2004. pp. V–825.
- [49] D. Stowell, M.D. Plumley, Automatic large-scale classification of bird sounds is strongly improved by unsuper- vised feature learning, PeerJ 2 (2014) e488.
- [50] L. Ptacek, L. Machlina, P. Linhart, P. Jaska, L. Muller, Automatic recognition of bird individuals on an open set using as-is recordings, Bioacoustics 25 (1) (2015) 1–19.
- [51] D. Lusseau, Evidence for social role in a dolphin social network, Evol. Ecol. 21 (3) (2007) 357–366.
- [52] S. Wasserman, Advances in Social Network Analysis: Research in the Social and Behavioral Sciences, Sage, 1994.
- [53] B. Shorrocks, D.P. Croft, Necks and networks: a preliminary study of population structure in the reticulated giraffe (*Giraffa camelopardalis reticulata de Winston*), Afr. J. Ecol. 47 (3) (2009) 374–381.

- [54] E. Stattner, Contributions à l'étude des réseaux sociaux: propagation, fouille, collecte de données (Doctoral dissertation), Université des Antilles-Guyane, 2012.
- [55] M. Laibowitz, J. Gips, R. Aylward, A. Pentland, J.A. Paradiso, A sensor network for social dynamics, in: Proceedings of the 5th international conference on Information processing in sensor networks, ACM, 2006, pp. 483–491.
- [56] P.G. Ryan, S.L. Petersen, G. Peters, D. Grémillet, GPS tracking a marine predator: the effects of precision, resolution and sampling rate on foraging tracks of African penguins, *Mar. Biol.* 145 (2) (2004) 215–223.
- [57] M.A. Rumble, L. Benkobi, F. Lindzey, R.S. Gamo, Evaluating elk habitat interactions with GPS collars, in: Tracking animals with GPS: an international conference held at the Macaulay Land Use Research Institute, Aberdeen, March 12–13, 2001, Macaulay Institute, Aberdeen, 2001, pp. 11–17.
- [58] D. Chen, J. Yang, R. Malkin, H.D. Wactlar, Detecting social interactions of the elderly in a nursing home environment, *ACM Trans. Multimedia Comput. Commun. Appl.* 3 (1) (2007) 6.
- [59] K. Yang, Wireless Sensor Networks. Principles, Design and Applications, Signals and Communication Technology, Springer-Verlag, London, 2014.
- [60] L. Da Xu, W. He, S. Li, Internet of things in industries: a survey, *IEEE Trans. Ind. Inform.* 10 (4) (2014) 2233–2243.
- [61] B. Bengherbia, M.O. Zmirli, A. Toubal, A. Guessoum, FPGA-based wireless sensor nodes for vibration monitoring system and fault diagnosis, *Measurement* 101 (2017) 81–92.
- [62] M. Prauzek, J. Konecny, M. Borova, K. Janosova, J. Hlavica, P. Musilek, Energy harvesting sources, storage devices and system topologies for environmental wireless sensor networks: a review, *Sensors* 18 (8) (2018) 2446.
- [63] Q. Chi, H. Yan, C. Zhang, Z. Pang, L. Da Xu, A reconfigurable smart sensor interface for industrial WSN in IoT environment, *IEEE Trans. Ind. Inform.* 10 (2) (2014) 1417–1425.
- [64] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, J. Anderson, Wireless sensor networks for habitat monitoring, in: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, ACM, 2002, pp. 88–97.
- [65] J. Hill, D. Culler, A Wireless Embedded Sensor Architecture for System-Level Optimization, (2002). UC Berkeley Technical Report.
- [66] H. Wang, D. Estrin, L. Girod, Preprocessing in a tiered sensor network for habitat monitoring, *EURASIP J. Appl. Signal Process.* 4 (2003) 392–401.
- [67] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, D. Estrin, Habitat monitoring with sensor networks, *Commun. ACM* 47 (6) (2004) 34–40.
- [68] V. Trifa, L. Girod, T.C. Collier, D. Blumstein, C.E. Taylor, Automated Wildlife Monitoring Using Self-Configuring Sensor Networks Deployed in Natural Habitats, 2007. Center for Embedded Network Sensing.
- [69] J.P. Dominguez-Morales, A. Rios-Navarro, M. Dominguez-Morales, R. Tapiador-Morales, D. Gutierrez-Galan, D. Cascado-Caballero, A. Jimenez-Fernandez, A. Linares-Barranco, Wireless sensor network for wildlife tracking and behavior classification of animals in Doñana, *IEEE Commun. Lett.* 20 (12) (2016).
- [70] S.P. Pubudu Aravinda, S. Gunawardene, a. Kottege, An acoustic Wireless Sensor Network for remote monitoring of bird calls, in: IEEE International Conference on Information and Automation for Sustainability (ICIAfs), 2016.
- [71] V. Bapat, P. Kale, V. Shinde, N. Deshpande, A. Shaligram, WSN application for crop protection to divert animal intrusions in the agricultural land, *Comput. Electron. Agric.* 133 (C) (2017) 88–96.

- [72] H. Brumm, The impact of environmental noise on song amplitude in a territorial bird, *J. Anim. Ecol.* 73 (3) (2004) 434–440.
- [73] N. Priyadarshani, S. Marsland, I. Castro, Automated birdsong recognition in complex acoustic environments: a review, *J. Avian Biol.* (2018).
- [74] I. Potamitis, S. Ntalampiras, O. Jahn, K. Riede, Automatic bird sound detection in long real-field recordings: applications and tools, *Appl. Acoust.* 80 (2014) 1–9.
- [75] K. Wolf, *Bird Song Recognition Through Spectrogram Processing and Labelling*, Univ. of Minnesota, 2009.
- [76] T.M. Ventura, A.G. de Oliveira, T.D. Ganchev, J.M. de Figueiredo, O. Jahn, M. I. Marques, et al., Audio parameterization with robust frame selection for improved bird identification, *Expert Syst. Appl.* 42 (22) (2015) 8463–8471.
- [77] L. Neal, F. Briggs, R. Raich, X.Z. Fern, Time-frequency segmentation of bird song in noisy acoustic environments, in: Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), May, 2011, pp. 2012–2015.
- [78] M.C. Baker, D.M. Logue, Population differentiation in a complex bird sound: a comparison of three bioacoustical analysis procedures, *Ethology* 109 (3) (2003) 223–242.
- [79] B.C. Pijanowski, et al., Soundscape ecology: the science of sound in the landscape, *BioScience* 61 (3) (2011) 203–216.
- [80] S.F. Boll, Suppression of acoustic noise in speech using spectral subtraction, *IEEE Trans. Acoust. Speech Signal Process.* 27 (2) (1979) 113–120.
- [81] N. Priyadarshani, S. Marsland, I. Castro, A. Punchihewa, Birdsong denoising using wavelets, *PLoS One* 11 (2016) e0146790.
- [82] Priyadarshani, Wavelet-Based Birdsong Recognition for Conservation (Ph.D. thesis), Massey Univ., Palmerston North, New Zealand, 2017.
- [83] A. Brown, S. Garg, J. Montgomery, Automatic and efficient denoising of bioacoustics recordings using MMSE STSA, *IEEE Access* 6 (2018) 5010–5022.
- [84] J.B. Alonso, J. Cabrera, R. Shyamnani, C.M. Travieso, F. Bolaños, A. García, A. Villegas, M. Wainwright, Automatic anuran identification using noise removal and audio activity detection, *Expert Syst. Appl.* 72 (2017) 83–92.
- [85] L. Muda, M. Begam, I. Elamvazuthi, Voice recognition algorithms using mel frequency cepstral coefficient (MFCC) and dynamic time warping (DTW) techniques, (2010). arXiv preprint arXiv:1003.4083.
- [86] J. Manikandan, B. Venkataramani, Study and evaluation of a multi-class SVM classifier using diminishing learning technique, *Neurocomputing* 73 (10) (2010) 1676–1685, <https://doi.org/10.1016/j.neucom.2009.11.042>.
- [87] B. Yegnanarayana, *Artificial Neural Networks*, PHI Learning Pvt. Ltd., 2009.
- [88] W. Chu, D.T. Blumstein, Noise robust bird song detection using syllable pattern-based hidden Markov models, in: IEEE International Conference on Acoustics, Speech and Signal Processing, 2011, pp. 345–348.
- [89] T. Ganchev, O. Jahn, M.I. Marques, J.M. de Figueiredo, K.-L. Schuchmann, Automated acoustic detection of *Vanellus chilensis lampronotus*, *Expert Syst. Appl.* 42 (15–16) (2015) 6098–6111.
- [90] C.-H. Lee, C.-H. Chou, C.-C. Han, R.-Z. Huang, Automatic recognition of animal vocalizations using averaged {MFCC} and linear discriminant analysis, *Pattern Recogn. Lett.* 27 (2) (2006) 93–101.
- [91] A. Boulmaiz, D. Messadeg, N. Doghmane, A. Taleb-Ahmed, Design and implementation of a robust acoustic recognition system for waterbird species using TMS320C6713 DSK, *Int. J. Ambient Comput. Intell.* 8 (1) (2017) 98–118.

- [92] N.H. Fletcher, A class of chaotic bird calls, *J. Acoust. Soc. Am.* 108 (2) (2000) 821–826, <https://doi.org/10.1121/1.429615>. 10955649.
- [93] I. Cohen, Noise spectrum estimation in adverse environments: Improved minima controlled recursive averaging, *IEEE Trans. Speech Audio Process.* 11 (5) (2003) 466–475.
- [94] R. Yu, A low-complexity noise estimation algorithm based on smoothing of noise power estimation and estimation bias correction, in: *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, IEEE, 2009, pp. 4421–4424.
- [95] P.C. Yong, S. Nordholm, H.H. Dam, Noise estimation based on soft decisions and conditional smoothing for speech enhancement, in: *Acoustic Signal Enhancement; Proceedings of IWAENC 2012; International Workshop on*, VDE, 2012, pp. 1–4.
- [96] F. Weninger, B. Schuller, Audio recognition in the wild: Static and dynamic classification on a real-world database of animal vocalizations, in: *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, IEEE, 2011, pp. 337–340.
- [97] A. Boulmaiz, D. Messadeg, N. Doghmane, A. Taleb-Ahmed, Robust acoustic bird recognition for habitat monitoring with wireless sensor networks, *Int. J. Speech Technol.* 19 (3) (2016) 631–645.
- [98] J.F. Kaiser, On a simple algorithm to calculate the energy'of a signal, in: *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88, 1988 International Conference on*, 1990, pp. 381–384.
- [99] T. Irino, R.D. Patterson, A time-domain, level-dependent auditory filter: the gamma-chirp, *J. Acoust. Soc. Am.* 101 (1) (1997) 412–419.
- [100] J. Kortelainen, K. Noponen, *Neural Networks, Intelligent Systems*, Addison-Wesley Publishing Co., Reading, MA, 2005.
- [101] H. Patil, K.K. Parhi, Novel variable length Teager energy based features for person recognition from their hum, in: *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, IEEE, 2010, pp. 4526–4529.
- [102] S.B. Davis, P. Mermelstein, Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences, *IEEE Trans. Acoust. Speech Signal Process.* 28 (4) (1980) 357–366.
- [103] O. Ghita, Auditory models and human performance in tasks related to speech coding and speech recognition, *IEEE Trans. Speech Audio Process.* 2 (1) (1994) 115–132.
- [104] S. Dharanipragada, M. Padmanabhan, A nonlinear unsupervised adaptation technique for speech recognition, in: *Sixth International Conference on Spoken Language Processing*, 2000.
- [105] F. Hilger, H. Ney, Quantile based histogram equalization for noise robust large vocabulary speech recognition, *IEEE Trans. Audio Speech Lang. Process.* 14 (3) (2006) 845–854.
- [106] N.V. Prasad, S. Umesh, Improved cepstral mean and variance normalization using Bayesian framework, in: *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, IEEE, 2013, pp. 156–161.
- [107] S. Yoshizawa, N. Hayasaka, N. Wada, Y. Miyanaga, Cepstral gain normalization for noise robust speech recognition, in: *Acoustics, Speech, and Signal Processing, 2004 (ICASSP'04). IEEE International Conference on*, vol. 1, IEEE, 2004, pp. I–209.
- [108] H. Boril, J.H. Hansen, Unsupervised equalization of Lombard effect for speech recognition in noisy adverse environments, *IEEE Trans. Audio Speech Lang. Process.* 18 (6) (2010) 1379–1393.
- [109] H. Boril, J.H. Hansen, UT-Scope: Towards LVCSR under Lombard effect induced by varying types and levels of noisy background, in: *Acoustics, Speech and Signal*

- Processing (ICASSP), 2011 IEEE International Conference on, IEEE, 2011, May, pp. 4472–4475.
- [110] S.O. Sadjadi, H. Boril, J.H.L. Hansen, A comparison of front-end compensation strategies for robust LVCSR under room reverberation and increased vocal effort, in: Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on, IEEE, 2012, pp. 4701–4704.
- [111] G. Hinton, L. Deng, D. Yu, G.E. Dahl, A.R. Mohamed, N. Jaitly, et al., Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups, *IEEE Signal Process. Mag.* 29 (6) (2012) 82–97.
- [112] L. Deng, D. Yu, J. Platt, Scalable stacking and learning for building deep architectures, in: Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on, IEEE, 2012, March, pp. 2133–2136.
- [113] S.M. Siniscalchi, D. Yu, L. Deng, C.H. Lee, Exploiting deep neural networks for detection-based speech recognition, *Neurocomputing* 106 (2013) 148–157.
- [114] S. Dharanipragada, B.D. Rao, U.S. Patent No. 7,016,839. U.S. Patent and Trademark Office, Washington, DC, 2006.
- [115] U.H. Yapanel, S. Dharanipragada, Perceptual MVDR-based cepstral coefficients (PMCCs) for robust speech recognition, in: Acoustics, Speech, and Signal Processing, 2003, Proceedings (ICASSP'03). 2003 IEEE International Conference on, vol. 1, IEEE, 2003, pp. I-I.
- [116] U.H. Yapanel, J.H. Hansen, A new perspective on feature extraction for robust in-vehicle speech recognition, in: Eighth European Conference on Speech Communication and Technology, 2003.
- [117] U.H. Yapanel, J.H. Hansen, A new perceptually motivated MVDR-based acoustic front-end (PMVDR) for robust automatic speech recognition, *Speech Comm.* 50 (2) (2008) 142–152.
- [118] C. Kim, R.M. Stern, Power-normalized cepstral coefficients (PNCC) for robust speech recognition, in: Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on, IEEE, 2012, pp. 4101–4104.
- [119] C. Kim, R.M. Stern, Robust signal-to-noise ratio estimation based on waveform amplitude distribution analysis, in: INTERSPEECH, 2008, pp. 2598–2601.
- [120] R.D. Patterson, K. Robinson, J. Holdsworth, D. McKeown, C. Zhang, M. Allerhand, Complex sounds and auditory images, in: Auditory Physiology and Perception, vol. 83, 1992, pp. 429–446.
- [121] X. Zhang, Y. Li, Adaptive energy detection for bird sound detection in complex environments, *Neurocomputing* 155 (2015) 108–116.
- [122] M. Slaney, Auditory toolbox, Interval Research Corporation, 1998. Technical Report (Vol. 10).
- [123] http://www.cs.cmu.edu/~robust/archive/algorithms/PNCC_IEEETran/.
- [124] S. Rangachari, P.C. Loizou, A noise-estimation algorithm for highly non-stationary environments, *Speech Comm.* 48 (2) (2006) 220–231.

Further reading

- [125] A. Henríquez, J.B. Alonso, C.M. Travieso, B.R. Herrera, F. Bolaños, P. Alpízar, et al., An automatic acoustic bat identification system based on the audible spectrum, *Expert Syst. Appl.* 41 (11) (2014) 5451–5465.

Index

Note: Page numbers followed by *f* indicate figures, *t* indicate tables, and *b* indicate boxes.

A

- Abstracted nodes, 94–96
- Accelerometers, 64
- Acoustic bird recognition system
 - alternative acoustic features
 - Mel-scaled wavelet packet decomposition subband cepstral coefficient (MWSCC), 293, 293*f*
 - perceptual-minimum variance distortion less response (MVDR), 291–292
 - perceptual MVDR-based cepstral coefficients, 291
 - power-normalized cepstral coefficients (PNCCs) algorithm, 292
- audio parameterization
 - Gammatone Teager energy cepstral coefficients (GTECC), 286–290, 289*f*
 - Mel frequency cepstral coefficients (MFCC)-spectral subtraction (SS) integration, 284–286, 285*f*
- classification
 - deep neural network (DNN), 290
 - support vector machine (SVM), 286
- noise reduction, 284–286
- preprocessing and segmentation, 280–284
- tonal region detection (TRD), 280–284
- Action Executor module, 206
- Actuators, 187
- Additive white Gaussian noise (AWGN), 145–146, 295–296
- Addressability, 100–103
- ADV messages, 10–11
- Alamouti coding, bit error rate analysis using, 144–146
- Alamouti space-time encoder, 144, 145*f*
- Alexa, 191
- Algeria, IUCN Red List status of bird species in, 264, 264*t*
- Ambient Assisted Living, 200
- AMIBIO project, 247–252, 249*f*
- Amplitude modulation (AM), 18–19
- AMR (ITU-T G.722.2), 171–172
- Analog-to-digital converters (ADCs), 275
- Answer Construction module, 206
- Anti-spoofing, xvi

ARBIMON acoustics project, 244–247, 244*f*, 249*f*

ARBIMON II project, 246–247

Assistants, 192–194

onboarding, 210*f*

Audio sensors, 65

Auditory filterbank learning approach, 18–19

Automatic speaker verification (ASV), xvi, 17

B

- Bactrocera oleae*, 252–253
- Base station (BS), 3, 8–9, 11–12
- Bayes' theorem, 282
- Bernoulli-Bernoulli RBMs (BBRBM), 290
- Binary Phase Shift Keying (BPSK) modulated system, xviii
 - bit error rate (BER) analysis, 148
 - vs. SNR, 152–156, 154*f*
- Bioacoustic sampling, 66
- Biodiversity monitoring
 - health and disease transmission, 254–255
 - marine, 240–242
 - pest control, 252–253
 - terrestrial, 242–252, 243*f*
 - AMIBIO project, 247–252
 - ARBIMON acoustics project, 244–247, 244*f*, 249*f*
 - urban ecosystems, 255–257
- Biomedical ecosystem, xvi–xvii, 52
- Bird species
 - biodiversity measurements, 262–263
 - ferruginous duck, 268–269
 - IUCN Red List, 261–262, 262*f*, 263–264*t*
 - monitoring, 261–262
 - comparison, of data collection techniques, 272, 273*t*
 - conventional observation method, 271
 - fixed devices, 272
 - habitat, 271–273
 - measurement and data collection, 270–271
 - mobile devices, 271–272
 - wildlife information gathering, 271–273
 - per continent, 261, 262*t*
 - recognition systems, 280–293
 - computational performance, 276*t*, 298, 299*t*
 - database, 293

- Bird species (*Continued*)
 experimental setup, 293
 noise robustness analysis, 297–298, 297f
 segmentation and recognition performance, 294–297, 295–296f
 vocalizations, 270, 280–281
 white-headed duck, 266–268, 268f
- Bit error rate (BER) analysis, xviii, 140
 of BPSK modulated system, 148
 of QAM modulated system, 149–150
 using Alamouti coding, 144–146
 decoding of speech symbols, 145–146, 145f
 encoding of speech symbols, 144–145, 145f
- Bluetooth low energy (BLE), 187
- C**
- C++, 47
 Carrier transfers, 113–117
 Casalendar, 191
 Cascaded star topology. *See* Tree topology
 Category theory, 94
 CELP (FS-1016), 171–172
 Cepstral distance (CSD), 29–30
 Cepstral gain normalization (CGN), 290
 Cepstral mean-variance normalization (CMVN), 290
 Chain-based hierarchical routing protocol, 8
 Channel Algebra, 95–97, 118, 121, 124
 Channel Expression Language (CXL), 97, 116
 Channel Group library, 116–117
 Channel groups, 115–117
 Channelized hypergraphs (CHs), 82, 85–93
 dependent types
 and coconstructors, 104–111
 simulating with preconstructors, 108–111
 and typestate, 105–108
 function-typed values, initialization of, 98–103
 addressability and implementation, 100–103
 modeling procedures via, 98–111
 Channelized type systems, 96–97
 Channels and carriers, 111–125
 carrier transfers, 113–117
 channel groups and code graphs, 115–117
 channelized-type interpretation, 118–125
 code blocks, as typed values, 122–125
 statements, blocks and control flow, 119–122
 Channel State Information (CSI), 143–144
 Clinical looking glass (CLG), xvii, 53–54
 Cluster-based hierarchical routing protocol, 8
 Coconstructors, 101, 103–111
 Code libraries, 47
 Communication topology, 7
 Communication unit, 4, 275
 Compressed integrated linear prediction residual (CILPR), 18–19
 Computable Number, 79
 Computing machines, 79
 Conference on Natural Language Learning (CONLL), 72–73
 CONLL-U, 75
 Constant-Q transform (CQT), 27–28
 Constrained Application Protocol (CoAP), 188
 Continuous noise, 278, 278–279f
 Control Home Easily, 191
 Conventional observation method, 271
 Conversational assistants, 186, 192–193, 194t
 Conversational interfaces, 192–193
 Convolutional neural networks (CNN), 18–19
 Convolutional Restricted Boltzmann Machine (ConvRBM), 18–19
 Core language *vs.* external tools, 60–63
 Cortana, 191
 CXL. *See* Channel Expression Language (CXL)
 Cyber-physical case studies, 68
 Cyber-physical ecosystems, 70
 Cyber-physical sensors, 70
 Cyber-physical software, xvi–xvii
 Cyber-physical systems, xv, 45–46, 50
 Cyber safety, 50, 54–55
 Cyber security, 50
 Cyclic prefix (CP), 141
 Cyclic-prefix Orthogonal Frequency Division Multiplexing (CP-OFDM) approach, xviii
- D**
- Data
 design for, 195–196
 models, 195–196
 Data-centric ontologies, 71
 Data-centric routing protocol
 directed diffusion, 11
 sensor protocol for information via negotiation (SPIN), 10–11
 DATA messages, 10–11
 Data modeling priorities, internet of things
 interoperability in, 69–72
 Data profiles, 63, 64t
 Dataset applications, 47
 Declarative knowledge, 91
 Deep neural networks (DNNs), 18–19, 165–166, 168t, 223–224, 290
 Detection error tradeoff (DET), 33–34, 40f, 294–295
 Device, registering, 207–208

Devices, diversity of, 189, 189*f*
 Directed diffusion, 11
 Directed hypergraphs, 63, 77–97, 111–112
 channel abstractions, 81–85
 Discrete cosine transform (DCT), 23–24, 285, 289
 Discrete Fourier transform (DFT), 23–24, 284
 Diversity, 140
 DNNs. *See* Deep neural networks (DNNs)
 DNN-WA model, 169–171

E

Ecological monitoring, xv
 El Kala National Park (PNEK), 264–266
 El-Taref, wetlands, 264, 265*f*, 266*t*
 Emotional database, 225
 Emotions and speech parameters, 228–230*t*, 229
 Empirical mode decomposition cepstral coefficient (EMDCC), 18–19
 EMR standards, 53
 Energy of excitation (EoE), 228
 Energy separation algorithm (ESA), 18–19
 ENTOMATIC project, 252–253, 253*f*
 Environmental noise, 277–279, 278–279*f*
 EoE. *See* Energy of excitation (EoE)
 Epoch feature (EF), 18–19
 Equal error rate (EER), 18–19, 33–34
 Equivalent rectangular bandwidth (ERB), 286–288
 Error classification, 75
 European Environment Agency, 255
 European Telecommunications Standards Institute, 171–172
 External hard disk drive (HDD), 245–246

F

Facial recognition, 67
 False acceptance rate (FAR), 33
 False rejection rate (FRR), 33
 Fault tolerance, 3, 6
 Ferruginous duck (*Aythya nyroca*), 268–269
 annual statistics of individuals, 269
 biometric feature, 268–269
 cry waveform, disturbance, 278–279, 278–279*f*
 general description, 268–269
 habitat and social behavior, 269
 IUCN Red List status of, 268, 269*t*
 Fifth-generation (5G) networks, 139–140
 Filter-bank multicarrier (FBMC), 140
 First-order difference operator (FOGD), 175
 5G Networks
 power-efficient speech transmission, signaling techniques, xvii–xviii
 UFMC modulation for, xviii

Fixed sensors, 272
 FLAC files, 250–251
 Fragile code, 57, 59–60
 Frequency modulation (FM), 18–19
 Full-spectrum dependent types, 107*b*
 Function-typed values, initialization of, 98–103

G

Gammatone filters (GTFs), 286–288, 289*f*
 Gammatone Teager energy cepstral coefficients (GTECC), 286–290, 289*f*
 Gatekeeper code, 48–63
 Gaussian-Bernoulli RBMs (GBRBM), 290
 Gaussian mixture models (GMM), 165–166, 225
 Gaussian mixture models with a universal background model (GMM-UBM), 165–166
 Generalized frequency-division multiplexing (GFDM), 140
 General semantic interpretation, 89
 Geographic adaptive fidelity (GAF), 12
 Geographical and energy-aware routing (GEAR), 12
 Geographical information system (GIS), 253
 German emotional speech database (EMO-DB), 224
 Gideon, 191
 Glottal closure instants (GCIs), 18–20, 25–26, 172–173, 175
 Glottal flow derivative cepstral coefficient (GFDCC), xvi, 20–21, 26–27
 GMM. *See* Gaussian mixture models (GMM)
 GMM-UBM system, 225, 226*f*, 227, 232
 Google Assistant, 191
 Graphical output modality, 201
 GSM full rate (ETSI 06.10), 171–172

H

Habitat preservation, xv
 Haskell's type system, 61*b*
 Health and disease transmission, 254–255
 Heart-rate monitor, 64
 Hierarchical routing protocol
 chain-based hierarchical routing protocol, 8
 cluster-based hierarchical routing protocol, 8
 hybrid energy efficient distributed clustering (HEED), 10
 low energy adaptive clustering hierarchy (LEACH), 8–9
 power efficient gathering in sensor information systems (PEGASIS), 9
 threshold sensitive energy efficient sensor network (TEEN) protocol, 9–10

- HL-7data, xvii, 52–53
 Hub applications, 46, 48–63
 Hub library, 46–47, 77
 roles, 47
 Human building interaction (HBI), 189–190
 Human-home interaction, 196–197, 197*f*
 ubiquitous, 203–208
 Human-human interaction, 196
 Hybrid energy efficient distributed clustering (HEED), 10
 Hypernodes, 81, 84, 86*b*, 92, 94–95
- I**
 Idris language, 104–107, 107*b*
 IFTTT, 191
 ITKGP-SESC, 224
 Impersonation technique, 17–18
 Impulse noise, 278, 278–279*f*
 Indian Institute of Technology Guwahati Multi-Variability (IITG-MV), 28–30, 36–37
 Insecticides, 252–253
 Instantaneous amplitude (IA) cosine coefficients, 18–19
 Instantaneous frequency (IF) cosine coefficients, 18–19
 Interaction manager (IM), 200
 Interactions, 188–189
 human building, 189–190
 smart homes, 190–191
 Intermittent noise, 278, 278–279*f*
 Internet of Things (IoT), 185, 195
 applications, 192*r*
 data modeling priorities, interoperability in, 69–72
 design for, 195
 Inter Symbol Interference (ISI), 141
 iPod, 245
 IPSO Smart Object Guidelines, 188
 Iterative adaptive inverse filtering (IAIF), 20–21
 Ivory language, 61
 Ivory’s type system, 61*b*
- J**
 JSON, 199, 199*f*
- L**
 Lambda calculus, 77–97, 118
 Lambda channel, 95–96, 114, 116, 118–119
 Language design, 60
 Language engineers, 62
 Language identification (LID) techniques, smart cities
- approaches, in mismatched conditions, 178*t*
 CL strategies, 179–180
 speech signals, 177–178
 vowel region based front-end system, 172–177
 baseline LID system
 attention network, DNN, 169–171, 169*f*
 DNN, 167–169
 equal error rate (EER), 167
 i-vector-based LID system, 167
 shifted delta cepstral (SDC), 167
 curriculum learning (CL)-based training strategies, 165
 database, 166–167
 explicit LID systems, 165
 implicit approaches, 165
 i-vector representation, 165–166, 168*t*
 language models, 165
 in mismatched environments
 mobile environment, 171–172
 noisy conditions, 171
 noisy and coded speech environments, 164
 phone-level speech recognition system, 165
 preprocessing level noise enhancement methods, 164
 self-attention mechanism, 165–166
 spoken language identification systems, 163–164
 training and operating environments, 164
 Linear frequency modified group delay cepstral coefficient (LFMGDCC), 18–19
 Linear prediction (LP), xvi
 Linear prediction cepstral coefficients, 223–224
 Linear Predictive Coding (LPC), 270
 Linguistic case, 72–75
 Local area network (LAN) switch, 245–246
 Location-based protocol
 geographic adaptive fidelity (GAF), 12
 geographical and energy-aware routing (GEAR), 12
 minimum energy communication network (MECN), 12–13
 Logistic function, 282–283
 Low energy adaptive clustering hierarchy (LEACH), 8–9
 LP coefficients (LPCs), 22–23
 LP residual Hilbert envelope mel frequency cepstral coefficient (LPRHEMFCC), xvi, 19–20, 24, 39
- M**
 Marine biodiversity monitoring, 240–242
 Mars Climate Orbiter crash, 50

Maximum Ratio Combining (MRC) techniques, xviii, 140–141, 147–148
 Medical whole slide imaging (MWSI), 65
 Mel-cepstral analysis, 19–20, 27
 Mel-frequency cepstral coefficients (MFCCs), 223–224, 226, 284–285
 based emotion recognition, 233–234
 emotional conditions using, 227*t*
 Mel frequency cepstrum (MFC), 284
 MELP (TI 2.4 kbps), 171–172
 Mel-scaled wavelet packet decomposition subband cepstral coefficient (MWSCC), 293, 293*f*
 Mel-scale relative phase (Mel-RP), 18–19
 MFCCs. *See* Mel-frequency cepstral coefficients (MFCCs)
 Migratory multimodal interfaces, 189
 Minimum energy communication network (MECN), 12–13
 Minimum mean square error short-time spectral amplitude estimator (MMSE STSA), 277
 Minimum variance distortion less response (MVDR), 291
 Mobile devices, 271–272
 Modulation centroid frequency cepstral coefficient (MCFCC), 18–19, 27, 39
 Modulation static energy cepstral coefficient (MSECC), 18–19
 Monad-subblock formation, 121–122
 Monitoring and management center (MMC), 253
 Mosquitoes, 254
 Multidevice contexts, 188–189
 Multilayer perceptron (MLP), 170–171
 Multimodal interaction (MMI), 185–186, 203–204, 204*f*
 across rooms, 214–215, 214–216*f*
 Multiple Input Single Output (MISO), 144
 MWSI. *See* Medical whole slide imaging (MWSI)

N

Native applications, 48
 Noise reduction
 acoustic bird recognition system, 284–286
 quantile-based cepstral dynamics normalization (QDN) for, 290
 Noise robustness analysis, 297–298, 297*f*
 NOISEX dataset, 171
 Notation3 (N3), 85–87
 Numerical Python (NPY), 66–67
 Nyquist frequency, 21

O

Olive fly. *See* *Bactrocera oleae*
 Oriental Language Challenge, 72–73

Oriental language recognition challenge (AP17-OLR) database, 166–167
 Orthogonal Frequency Division Multiplexing (OFDM), 149
 Orthogonal Frequency Division Multiplexing (OFDM), 139–140
 PAPR comparison of, 154*t*
 spectrum analysis of speech signal, 152, 153*f*
 Orthogonal Space-Time Block Code (OSTBC), 146–147
 BER vs. SNR, 152–156, 155*f*
 encoder, 146, 146*f*
 speech symbols
 decoding of, 147, 147*f*
 encoding of, 146–147, 146*f*

P

Patient-centered query methodology, 53
 Peak-to-Average Power Ratio (PAPR), 150–152
 Peak-to-side lobe ratio mean and skewness (PSRMS), 18–19
 Pest control, 252–253
 Phase-based source-filter vocal tract (PBSFVT), 18–19
 Philips Hue app, 191
 Pitch contour, xvi, 26, 26*f*, 40
 Pointer, 201
 Polymorphism, 100
 Power efficient gathering in sensor information systems (PEGASIS), 9
 Power-normalized cepstral coefficients (PNCCs), 292, 292*f*
 Power spectral density (PSD), 283–284
 Power supply unit, 4, 275
 Proactive design, 75–77
 Probability density functions (PDFs), 282
 of Gaussian noise, 149*f*
 Procedural alignment, 70
 Procedural input/output protocols
 abstraction, kinds of, 94–96
 channelized type systems, 96–97
 via type theory, 93–97
 Procedural knowledge, 91–92
 Processing unit, 4, 275

Q

Quadrature Amplitude Modulation (OQAM), xviii
 Quadrature Amplitude Modulation (QAM)
 modulated systems, xviii
 bit error rate (BER) analysis, 149–150
 vs. SNR for, 152–156, 155–156*f*

- Quantile-based cepstral dynamics normalization (QCN), 290
- Question Answering module, 205
- R**
- Radio Access Technology (RAT), xvii–xviii
- RBM. *See* Restricted Boltzmann machine (RBM)
- RDF. *See* Resource Description Framework (RDF)
- Receiving diversity, 140
- Rectified linear unit (ReLU), 167–169
- Recurrent neural networks (RNN), 18–19
- REMOSIS (Remote Mosquito Situation and Identification System) project, 254–255, 254f
- Remote medical diagnosis, 65
- Replay attack false acceptance rate (RFAR), 33–34, 39
- Replay detection
- amplitude and frequency modulation (AM and FM), 18–19
 - ASV spoof 2017 database, 17–18, 18f, 30, 35t, 37–38
 - auditory filterbank learning approach, 18–19
 - compressed integrated linear prediction residual (CILPR), 18–19
 - convolutional neural networks (CNN), 18–19
 - Convolutional Restricted Boltzmann Machine (ConvRBM), 18–19
 - deep neural network (DNN), 18–19
 - empirical mode decomposition cepstral coefficient (EMDCC), 18–19
 - energy separation algorithm (ESA), 18–19
 - epoch feature (EF), 18–19
 - equal error rate (EER), 18–19
 - evaluation process
 - detection error tradeoff, 33–34, 40f
 - equal error rate, 33–34
 - tandem-detection cost function (t-DCF), 34 - explicit source features
 - GFDCC feature, 26–27
 - pitch contour feature, 26, 26f, 40 - Gaussian mixtures model-universal background model (GMM-UBM), 19–20
 - glottal closure instants (GCIs), 18–19
 - IITG-MV database, 19–20, 28–30, 36–37
 - impersonation, 17
 - instantaneous amplitude (IA) cosine coefficients, 18–19
 - instantaneous frequency (IF) cosine coefficients, 18–19
 - iterative adaptive inverse filtering (IAIF), 20–21
- linear frequency modified group delay cepstral coefficient (LFMGDCC), 18–19
- mel-scale relative phase (Mel-RP), 18–19
- peak-to-side lobe ratio mean and skewness (PSRMS), 18–19
- phase-based source-filter vocal tract (PBSFVT), 18–19
- prerecorded target speaker’s speech samples, 17
- recurrent neural networks (RNN), 18–19
- replay speech signal, 18–19, 21–22
- residual-based implicit source features
- Hilbert envelope, 23
 - LP residual features, 25–26
 - LPRHEMFCC feature, 24
 - RMFCC feature, 23–24
 - RPCC features, 24
- signal-to-noise ratio (SNR), 18–19
- single frequency filtering cepstral coefficients (SFFCC), 18–19
- speech synthesis (SS), 17–18
- SV and spoof detection experiments, 32
- classifiers, 31
 - features, 31
- system features
- CQCC feature, 27–28
 - MFCC feature, 27
 - voice conversion (VC), 17–18
- zero band filtering (ZBF) approach, 20–21
- Replay speech signal, 18–19, 21–22
- REQ messages, 10–11
- Research and development cycle, 72
- Residual mel frequency cepstral coefficient (RMFCC), xvi, 19–20, 39
- Residual phase cepstral coefficient (RPCC), xvi, 19–20, 39
- Resource-block filtered orthogonal frequency-division multiplexing (RB-F-OFDM), 140
- Resource Description Framework (RDF), 85–93
- Restricted Boltzmann machine (RBM), 290
- Return channel, 114, 116, 118–119
- RNNLM, 165
- Robust code, 59
- Routing protocol, WSNs, xvi
- data-centric routing protocol
 - directed diffusion, 11
 - sensor protocol for information via negotiation (SPIN), 10–11
- hierarchical routing protocol
- chain-based hierarchical routing protocol, 8
 - cluster-based hierarchical routing protocol, 8
 - hybrid energy efficient distributed clustering (HEED), 10

- low energy adaptive clustering hierarchy (LEACH), 8–9
 - power efficient gathering in sensor information systems (PEGASIS), 9
 - threshold sensitive energy efficient sensor network (TEEN) protocol, 9–10
 - location-based protocol
 - geographic adaptive fidelity (GAF), 12
 - geographical and energy-aware routing (GEAR), 12
 - minimum energy communication network (MECN), 12–13
 - Ruddy Duck (*Oxyura jamaicensis*), 267
- S**
- SCA. *See* Source Code Algebra (SCA)
 - Semantic Sensor Network (SSN), 188
 - Sensing unit, 3, 275
 - Sensor networks, high level handling of, 187–188
 - Sensor protocol for information via negotiation (SPIN), 10–11
 - Sensors, 187
 - Short-time Fourier transform (STFT), 289
 - Sigma channel, 114, 116, 118–119
 - Sigmoidal curve, 282–283
 - Signal-to-noise ratio (SNR), 18–19
 - Single frequency filtering cepstral coefficients (SFFCC), 18–19
 - Siri, 191
 - Skype, 208–209, 209 f
 - Smart appliances, 188
 - Smart Cities, xv
 - Smart homes, 185
 - assistant onboarding, 209, 210 f
 - challenges, 190–191
 - chatting with, 208–212, 209–212 f
 - home control, 200
 - information structuring, 198–199
 - interaction with, 190–191
 - multimodal interaction support, 200–201
 - ontology used for home information, 198–199, 198 f
 - platform, 69–70
 - sending email to home, 212, 212 f
 - sensors and actuators, 187
 - test bench, 201–202, 202–203 f
 - users and context, 199
 - SoE. *See* Strength of excitation (SoE)
 - Software malfunctions, Mars Climate Orbiter crash, 50
 - Soundscape
 - acquisition, transmission, and analysis, 227, 243 f
 - Source Code Algebra (SCA), 61, 62 b , 77–78
 - Source Code Ontology, 98
 - SOX toolkit, 171–172
 - Space-time codes, bit error rate analysis
 - using Alamouti coding, 144–146
 - SPARQL query, 205–206
 - Speaker-based variability, 223–224
 - Speaker recognition (SR) system, 223–225
 - baseline emotional, 225–227
 - emotional conditions, performance analysis in, 227–230, 227–228 t
 - duration, 228–230, 229–230 t
 - energy of excitation, 228
 - fundamental frequency, 228
 - strength of excitation, 228
 - with emotional data, 231–232, 231 t
 - emotional database, 225
 - with emotional UBM, 230–231
 - emotion recognition, speaker selection model
 - based on, 232, 232 f , 233 t
 - proposed strategies in emotional conditions, 230–232
 - Speaker verification (SV) technology, 17, 201
 - Special semantic interpretation, 89
 - Spectral subtraction (SS), 285, 285 f , 296–297
 - Spectrum analysis
 - of speech signal using OFDM, 152, 153 f
 - of speech signal using UFMC, 152, 154 f
 - Speech, 200
 - input, 200
 - and language technology, 72
 - output, 200
 - sampling, 65
 - signal, 153 f , 223–224
 - Speech symbols
 - decoding of
 - bit error rate analysis using Alamouti coding, 145–146, 145 f
 - Orthogonal Space-Time Block Code (OSTBC), 147
 - encoding of
 - bit error rate analysis using Alamouti coding, 144–145, 145 f
 - Orthogonal Space-Time Block Code (OSTBC), 146–147, 146 f
 - Speech synthesis (SS), 17–18
 - SRILM, 165
 - Strength of excitation (SoE), 228
 - Support vector machine (SVM), 286

T

Tandem-detection cost function (t-DCF), 34
 Teager-Kaiser energy operator (TEO), 288–289
 Teager’s energy operator, 286
 Terrestrial biodiversity monitoring, 242–252, 243f
 AMIBIO project, 247–252
 ARBIMON acoustics project, 244–247, 244f, 249f
 Text-dependent mode, 223
 Text-independent speaker, 223
 Threshold sensitive energy efficient sensor network (TEEN) protocol, 9–10
 Tonal region detection (TRD), 280–284
 Tonal region presence probability (TRPP), 282–284
 Touch, 201
 gestures, 201
 Transmitting diversity, 140
 Tree topology, 7
 TURTLE, 87–88
 Two-tiered directed hypergraphs, 85
 TXL. *See* Type Expression Language (TXL)
 Type Expression Language (TXL), 97, 100
 Type theory, procedural input/output protocols via, 93–97

U

Ubiquitous computing, xv
 power-efficient speech transmission, 5G
 Networks, xvii–xviii
 wetlands, xv–xvi
 WSNs, xv–xvi
 Ubiquitous human-home interaction
 conversational capability, 204–208, 205f
 device registering, 207–208, 208f
 Watson training, 206–207
 multimodal interaction, 203–204, 204f
 Ubiquitous multidevice interactions, 189
 Ubiquitous sensing for healthcare (USH),
 principles, 51b
 UBM. *See* Universal Background Model (UBM)
 U-healthcare systems, 51b
 Ultra-Filter Bank Multicarrier modulation (UFMC), xvii–xviii, 141–143
 equalized symbols, 143f
 PAPR comparison of, 154t
 preequalization symbols, 142f
 receive processing, 143, 143f
 receiver, 143, 143f
 spectrum analysis of speech signal, 152, 154f
 transmit-end processing, 142, 142–143f
 transmitter, 142f

Universal background model (UBM), 167, 225–226

SR system, with emotional, 230–231, 231t

Universal-filtered multicarrier (UFMC), 140–141

Urban ecosystems, monitoring, 255–257

User face identification, 201

User input analysis, 204

User interface networking, with smart refrigerators, 48–49

USH. *See* Ubiquitous sensing for healthcare (USH)

V

Valletto, 193, 194t
 Vapnik Chervonenkis (VC) dimension theory, 286
 Voice activity detection (VAD), 23–24
 algorithm, 200
 Voice conversion (VC), 17–18
 Voice user interfaces (VUI), 192
 Vowel end-points (VEPs), 172–175
 Vowel onset point (VOP), 172–175

W

Watson training, 206–207
 Wavelet packet decomposition (WPD), 293
 WAV format file, 66–67
 Wearable sensors, 271
 Web applications, 48
 WebSockets protocol, 200
 White-headed duck (*Oxyura leucocephala*), 266–268
 annual statistics of individuals, 267–268
 biometric feature, 267
 general description, 267
 habitat and social behavior, 267
 IUCN Red List status of, 267, 267t
 monthly counts of, 267, 268f
 Wi-Fi, 187
 Wildlife monitoring, WSNs for, 276
 Wink, 191
 Wireless sensor networks (WSNs), xv, 253
 advantages, 273–274
 applications of, 5–6
 automatic recognition system for bird sounds
 using, 276, 277f
 environmental noise, 277–279, 278–279f
 wireless sensor nodes, limited computational
 capacity of, 279–280
 base station (BS), 3
 bus topology, 7
 characteristics and constraints of
 auto-configuration and auto-organization, 6
 fault tolerance, 6
 limited lifetime, 6

- processing and storing,
low capacities of, 6
reliability, 6
scalability, 6
disadvantages, 274
mesh topology, 7–8
micro-sensors, 3
network topology, 7–8
node structure, 274–275, 275*f*
 communication unit, 275
 power supply unit, 275
 processing unit, 275
 sensing unit, 275
optimization energy consumption, routing
 protocols for, xvi
principles, 273–274
routing protocol for (*see* Routing protocol,
WSNs)
sensor nodes, 3
star topology, 7
topology, 273–274, 274*f*
- tree topology, 7
for wildlife monitoring, 276
Wireless sensor nodes, 274–275, 275*f*
 characteristics of, 276*t*
 communication unit, 275
 computational capacity of, 279–280
 power supply unit, 275
 processing unit, 275
 sensing unit, 275
WSNs. *See* Wireless sensor networks (WSNs)
- X**
- XML, 127
 device registering, 207, 208*f*
- Z**
- Zero band filtering (ZBF) approach, 20–21
Zero-effort false acceptance rate (ZFAR),
 33–34, 39
Zero frequency filtering (ZFF), 172–175, 173*f*, 228
ZigBee devices, 187

This page intentionally left blank

ADVANCES IN UBIQUITOUS COMPUTING

CYBER-PHYSICAL SYSTEMS, SMART CITIES AND ECOLOGICAL MONITORING

Advances in Ubiquitous Computing: Cyber-Physical Systems, Smart Cities and Ecological Monitoring debuts some of the newest methods and approaches to multimodal user-interface design, safety compliance, formal code verification and deployment requirements, as they pertain to cyber-physical systems, smart homes and smart cities, and biodiversity monitoring. In this anthology, the authors assiduously examine a panoply of topics related to wireless sensor networks. These topics include interacting with smart-home appliances and biomedical devices, designing multilingual speech recognition systems that are robust to vehicular, mechanical and other noises common to large metropolises, and an examination of new methods of speaker recognition to control for the emotion-state of the speaker, which can easily impede speaker verification over a wireless medium.

This volume recognizes that any discussion of pervasive computing in smart cities must not end there, as the perilous effects of climate change proves that our lives are not circumscribed by the geographically sculpted boundaries of cities, counties, countries, or continents. Contributors address present and emerging technologies of scalable biodiversity monitoring: pest control, disease transmission, environmental monitoring, and habitat preservation. The need to collect, store, process, and interpret vast amounts of data originating from sources spread over large areas and for prolonged periods of time requires immediate data storage and processing, reliable networking, and solid communication infrastructure, along with intelligent data analysis and interpretation methods that can resolve contradictions and uncertainty in the data—all of which can be bolstered by modern advances in ubiquitous computing.

This book is an invaluable reference to system designers, speech scientists, biomedical researchers, and engineers, as well as emergency healthcare management agencies and first responders in the field.

KEY FEATURES

- Examines the history, scope, and advances in ubiquitous computing, including threats to wildlife, tracking of disease, smart cities, and wireless sensor networks.
- Discusses user-interface design, implementation and deployment of cyber-physical systems, such as wireless sensor networks, Internet of things devices, and other networks of physical devices that have computational capabilities and reporting devices
- Covers the need for improved data sharing networks

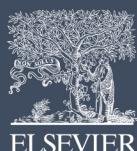
VOLUME EDITOR

Dr. Amy Neustein has served as editor-in-chief of the International Journal of Speech Technology since 2008. She is the author/editor of 12 academic books that span various topics from speech and automata in healthcare, advances in speech recognition and natural language processing, text mining of web-based medical content, acoustic modeling of speech disorders, forensic speaker recognition, voice technologies for speech reconstruction and enhancement, to acoustic analysis of pathologies from infancy to young adulthood. She serves as editor of three book series and has authored over 40 papers and chapters. She heads up a think tank in Fort Lee, NJ, Linguistic Technology Systems, and has developed a novel dataset creator framework for publishing research objects.

ABOUT THE SERIES

Series Editors: Dr. Nilanjan Dey, Dr. Amira S. Ashour, Dr. Simon James Fong

Advances in Ubiquitous Sensing Applications for Healthcare is a multivolume, interdisciplinary series exploring the groundbreaking field of ubiquitous sensing for healthcare (USH), with a focus on real-world healthcare applications. It is an essential reference for researchers in engineering, software design, communication systems, and physicians and academics working directly on the development of these U-healthcare systems.



ACADEMIC PRESS

An imprint of Elsevier

elsevier.com/books-and-journals

ISBN 978-0-12-816801-1



9 780128 168011