Alma Mater Studiorum – Università di Bologna

DOTTORATO DI RICERCA IN

Elettronica, Telecomunicazioni e Tecnologie dell'Informazione

Ciclo **XXV**

Settore Concorsuale di afferenza: **09/E3**

Settore Scientifico disciplinare: **ING-INF-01**

TITOLO TESI

Tecnologie per l’intelligenza ambientale: dagli smart objects alle reti di sensori

Presentata da: **BOJAN MILOSEVIC**

Coordinatore Dottorato	**Relatore**
Prof. **Alessandro Vanelli Coralli**	Prof. **Luca Benini**

Esame finale anno 2013

Technologies for Ambient Intelligence: from Smart Objects to Sensor Networks

Bojan Milosevic

DEI - Dpt. of Electrical, Electronic and Information Engineering

University of Bologna

A thesis submitted for the degree of

Doctor of Philosophy

2013

I would like to dedicate this thesis to my loving family.

Abstract

In the last few years, the vision of our connected and intelligent information society has evolved to embrace novel technological and research trends. The diffusion of ubiquitous mobile connectivity and advanced handheld portable devices, amplified the importance of the Internet as the communication backbone for the fruition of services and data. The diffusion of mobile and pervasive computing devices, featuring advanced sensing technologies and processing capabilities, triggered the adoption of innovative interaction paradigms: touch responsive surfaces, tangible interfaces and gesture or voice recognition are finally entering our homes and workplaces. We are experiencing the proliferation of smart objects and sensor networks, embedded in our daily living and interconnected through the Internet. This ubiquitous network of always available interconnected devices is enabling new applications and services, ranging from enhancements to home and office environments, to remote healthcare assistance and the birth of a smart environment.

This work will present some evolutions in the hardware and software development of embedded systems and sensor networks. Different hardware solutions will be introduced, ranging from smart objects for interaction to advanced inertial sensor nodes for motion tracking, focusing on system-level design. They will be accompanied by the study of innovative data processing algorithms developed and optimized to run on-board of the embedded devices. Gesture recognition, orientation estimation and data reconstruction techniques for sensor networks will be introduced and implemented, with the goal to maximize the tradeoff between performance and energy efficiency. Experimental results will provide an evaluation of the accuracy of the presented methods and validate the efficiency of the proposed embedded systems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The term *Ambient Intelligence* (AmI) refers to a near future vision of the information society where small, unobtrusive and ubiquitous electronic devices will augment the environment, making it sensitive and responsive to the presence of people and their activities. This form of context awareness of the environment will be possible thanks to the pervasive presence of invisible technology, embedded in the surroundings. People will seamlessly interact with such a smart environment, without the need to be aware of its presence or composition.

The concept of ambient intelligence was been developed in the late 90s and builds on three key technologies:

- **Ubiquitous Computing**: the integration of electronic systems and sensors into everyday objects like furniture, clothing, white goods, toys and even paint.
- **Ubiquitous Communication**: the ability of these objects to communicate with each other and the user by means of ad-hoc and wireless networking.
- **Intelligent User Interfaces**: which enable the inhabitants of the AmI environment to control and interact with it in a natural (voice, gestures) and personalized way (preferences, context).

The objective of AmI is to broaden the interaction between human beings and digital information through the usage of ubiquitous computing devices. Conventional computing primarily involves user interfaces (UIs) such as keyboard, mouse, and a visual display unit; while the large amount of ambient space that encompasses the user is not utilized as it could be. AmI on the other hand uses this space in the form of, for example, shape, movement, scent and sound recognition or output. These information media become possible through new types of interfaces and will allow for drastically simplified and more intuitive use of computing devices. Examples of such devices range from common items such as pens, watches, and household appliances to sophisticated computers and production equipment.

The development of the AmI vision was consolidated in 2001, when the ISTAG (Information Society Technology Advisory Group) defined four scenarios to offer a provocative glimpse of the future that could be realized in the following 10 years [64]. Each scenario had a script that was used to work out the key developments in technologies, society, economy, and markets necessary to arrive at the realization of the vision. In particular, a wide range of different technologies were identified as key enablers for a pervasive ambient intelligence, such as: the development of very unobtrusive and miniaturized computing and sensing hardware, a seamless mobile/fixed communication infrastructure, dynamic and massively distributed device networks, natural user interfaces, dependability and security. In the following years, the efforts of the scientific community and the industry contributed to a substantial technological advance in every one of these fields.

Sensors and sensing devices are one of the technologies enabling the AmI vision, since they contribute to bridge the gap between the physical and the digital worlds. From the user perspective, the development of innovative sensors, such as miniaturized Micro Electro-Mechanical Systems (MEMS), wearable sensors or depth cameras, enabled the adoption of natural interaction paradigms. Physical objects, augmented with sensing and communication capabilities, became *smart objects*, which integrate into the smart environment and give the user the ability to exploit its physical and gestural abilities to interact with the digital world. Sensors deployed in the environment, on the other hand, compose the sensing interface towards the physical world of the ubiquitous digital ecosystem. With the data gathered from the environment, intelligent and distributed systems can monitor the ambient or the user activities and collect context information.

Large amounts of mobile and fixed sensors embedded into the environment can be exploited when organized in *Wireless Sensor Networks* (WSNs), one of the most relevant enabling technologies for AmI. The WSN design space is very wide and spans from small, fixed *Body Area Networks* (BANs) for health or motion analysis, composed of a handful of sensors nodes placed over the body; to large, dynamic networks for environment monitoring consisting of thousand of nodes.

Despite the design of sensor nodes and networks being strongly influenced by the application, an ubiquitous goal is the optimization of the tradeoff between miniaturization and low-power consumptions on one side and advanced processing and communication capabilities on the other. Each sensor node typically includes one or more sensors, a wireless radio, an energy source and a low-power microcontroller, usually with limited computational resources. The small size of the devices, needed for an unobtrusive and effective embedding in the environment, limits the size of the batteries and components that can be used. This gives a limit to the power consumption of the system, which can be improved by employing energy harvesting techniques from available sources and carefully optimizing the performed operations and data transfers, allowing the devices to spend most of the time in low-power states. Power consumptions are also minimized by the adoption of ad-hoc wireless communication protocols, the radio communications being the most power demanding task.

The advances in hardware design of embedded devices are accompanied by advances in data processing algorithms and software development. In particular, the availability of multiple sensor sources and large amounts of data triggered the development of sensor fusion algorithms and artificial intelligence techniques to extract useful information and accurate results from the gathered data. Even in the presence of a small number of nodes, a centralized collection and processing of sensor data is inefficient due to the high cost of radio communications and the poor scalability of this approach. The implementation of distributed on-board processing techniques for the sensor nodes improves the overall system performance, while reducing the amount of data circulating within the network. A variety of processing techniques, ranging from data filtering and compression algorithms to advanced sensor fusion approaches, can be optimized to fully exploit the limited resources available on small and low-power microcontrollers embedded in smart objects or sensor nodes. In the case of networked devices, local processing and the parallelization of the algorithms make it possible to accurately achieve complex tasks relying on multiple resource-constrained devices.

A decade after its definition, the AmI vision is still not as a widespread concept as it was intended and has not penetrated our lives according to the expectations. The vision, instead, contributed to a decade of scientific and technological progress, and to a wider debate on the present and future of our society. There are still formidable challenges to be tackled by the scientific community and there is no guarantee we will solve all of them, but the more progress we make on some, the closer we will be to a stage where some of the initial aims are adopted.

In the last few years the vision of our connected and intelligent information society has evolved to embrace novel technological and research trends. The diffusion of ubiquitous mobile connectivity and advanced handheld portable devices, such as smartphones and tablets, amplified the importance of the Internet as the communication backbone for the fruition of services and data. This shift towards mobile and pervasive computing devices triggered the adoption of innovative interaction paradigms: touch responsive surfaces, tangible interfaces and gesture or voice recognition are finally entering our homes and workplaces. We are experiencing the proliferation of smart objects and sensor networks, embedded in our daily living and interconnected through the Internet. This ubiquitous network of interconnected devices, always available and often referred to as the *Internet of Things* (IoT) is enabling new applications and services, ranging from enhancements to home and office environments, to remote healthcare assistance and the birth of *smart cities*.

In the context of the technological evolution introduced, the work presented in this dissertation focuses on hardware and software co-design of embedded systems and sensor networks.

From the hardware perspective, we aimed on system-level design: building on top of the latest hardware advances and employing state-of-the-art integrated technologies, such as MEMS sensors and power efficient microcontrollers, different embedded devices were designed. Starting from the *Smart Micrel Cube* (SM-

Cube), a smart object equipped with an accelerometer for tangible and gestural interaction, we designed the *SmartPen*, a wireless, pen-shaped, 3D interaction device, which embeds a full Inertial Measurement Unit (IMU) composed of an integrated digital accelerometer, gyroscope and magnetometer. In conjunction with a low-cost video tracking system, we can accurately track the position and orientation of the pen used for interactive sketching or editing of surfaces in a *CAD* environment. The IMU design was then refined and miniaturized into a wearable inertial sensor node, developing an unobtrusive device for motion analysis, the *EXLs1* sensor node. This node was used as the building block for the development of a wireless body area network for motion analysis and posture tracking, with applications in remote healthcare, support for motor rehabilitation or human computer interaction. We can track an evolution path for the proposed hardware: a smart object with a single sensor evolved in a multi-sensor device, which then was enhanced with the use of video tracking and finally embedded into a wearable device used to form a body area network of cooperating nodes.

The development of the hardware devices introduced were accompanied by the study and implementation of advanced processing techniques to analyze the generated sensor data and obtain higher level information. In the case of the SMCube, a gesture recognition algorithm was designed, to recognize meaningful gestures performed with the device and employed as interaction primitives to control applications. For the IMU-equipped devices, optimized filtering and sensor fusion techniques were adopted to convert raw sensor readings to orientation and position information, used as features for further analysis of the motion or as an input for interactive applications. Furthermore, advanced data compression and reconstruction techniques were examined and developed in the more general context of wireless sensor networks, to improve the accuracy of collected data while reducing the energy consumptions. Here also, we can track an evolution in the proposed algorithms, following the evolution of hardware solutions: from gesture recognition applications based on a single sensor data stream, we went through multi-sensor data fusion techniques to complex data reconstruction algorithms for wide sensor networks.

For all the cases, the tradeoff between energy efficiency and system accuracy was a key factor. All of the presented solutions and algorithms have optimized embedded implementations for the proposed hardware, with the needed elaborations taking place directly on board of the devices. This approach reduces energy consumptions, removing the need to send the raw data stream from the nodes to other host devices, while maintaining the desired accuracy and facilitating the integration and scalability of networked solutions.

1.1 Thesis outline

Chapter 2 will provide a technological background for the work, giving an overview of Ambient Intelligence and related projects, introducing the hardware architecture of embedded systems and providing state of the art examples of sensor nodes. Moreover it will provide an overview of the software layer and the processing techniques used to elaborate data from the sensing devices.

The **Chapter 3** introduces the *SMCube*, an interactive smart object equipped with an accelerometric sensor. This device is used to enable natural interaction modalities and a gesture recognition algorithm was developed and optimized for the implementation on the embedded microcontroller integrated in the device. The chapter will describe the device, the proposed algorithm and will present experimental results validating its performance and the computational costs.

The following **Chapter 4** describes the development of the *SmartPen*, a wireless pen-shaped device for 3D user interaction. It is equipped with an inertial measurement unit and combined with a low-cost, infra-red, stereo vision system, to allow its use as a free-hand sketching tool. In conjunction with a software layer, it can be used to rapidly reconstruct or modify virtual 3D models of objects and surfaces. Details on the hardware setup and the developed tracking and surface reconstruction algorithms will be provided, along with the results of some validation experiments.

In **Chapter 5**, a wearable sensor node for the analysis of motion will be presented. Here the hardware solutions of the *SmartPen* have been optimized to produce the *EXLs1* sensor node, a small and light wearable node. It embeds an inertial measurement unit and Bluetooth communication capabilities for the use in a body sensor network to capture complex human motions. A set of sensor fusion techniques have been analyzed and implemented locally on the device, to estimate its orientation in a distributed and energy efficient way.

Chapter 6, on the other hand, will study energy efficient data reconstruction techniques for wireless sensor networks. Evolving from limited body area networks to wide area ones, we propose two techniques to reconstruct the original captured signals from their undersampled versions. By applying extreme duty-cycling policies on the sensor nodes, we obtain a reduction of the power consumptions in the network. The missing data is reconstructed at the collection center, exploiting redundancies and correlations present in the gathered data to meet the required accuracy. Experimental validation of energy consumptions and reconstruction performance is presented using different data sets and a real sensor network deployment.

Finally, in **Chapter 7**, a summary and the conclusions of the work are provided.

Chapter 2

Background

This work is focused on system-level hardware and software co-design and prototyping of embedded devices for various AmI applications, such as human computer interaction (HCI), healthcare and rehabilitation therapy. In this chapter a background overview on the subject will be introduced, to better understand the technological perspective of the vision and the state of the art in the development of embedded systems and algorithms for AmI.

2.1 Ambient Intelligence

Ambient intelligence is the vision of a future in which environments support the people inhabiting them, providing seamless access to digital data and services in an unobtrusive, dynamic and intelligent way. Traditional computational devices, such as PCs, keyboards or mice, tend to disappear, while processors and sensors are integrated into everyday objects, allowing us to manipulate digital content in a natural and personalized way. In this scenario, clothes, household devices, furniture or vehicles became smart agents in an interconnected digital ecosystem, communicating between them to fulfill and even anticipate the needs of the users. This smart environment is aware of the personal requirements and preferences of its inhabitants and interacts with them in a user-friendly way, recognizing gestures voice and even implicit actions or emotions.

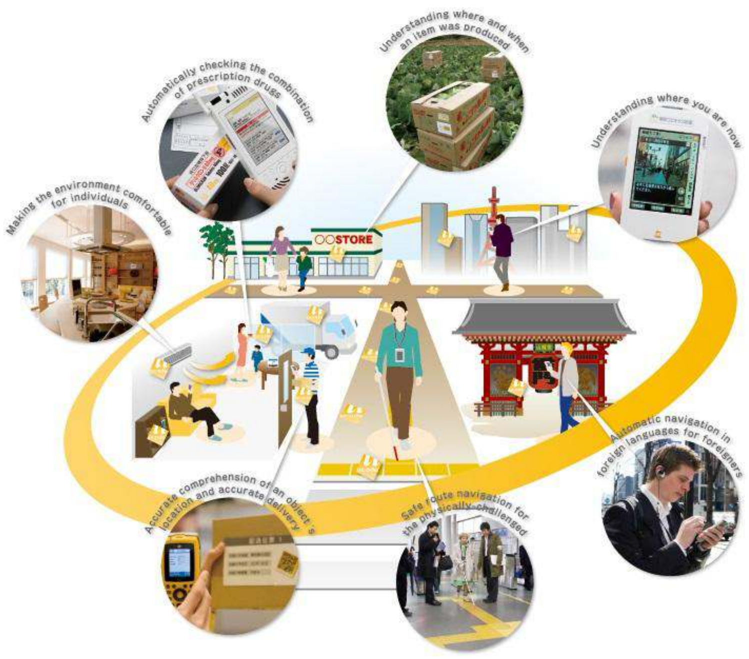

Understanding where and when an item was produced
Automatically checking the combination of prescription drugs
Understanding where you are now
Making the environment comfortable for individuals
OOSTORE
Automatic navigation in foreign languages for foreigners
Accurate comprehension of an object's location and accurate delivery
Safe route navigation for the physically challenged

Figure 2.1: Ubiquitous smart environment.[1]

The term ambient intelligence emerged from the work of the European Community's Information Society Technologies Programme (ISTAG). The initial report layed out the concept aimed to give a strategic orientation for the programming of the annual research agendas of the IST [64]:

> *Start creating an ambient intelligence landscape (for seamless delivery of services and applications) in Europe relying also upon test-beds and open source software, develop user-friendliness, and develop and converge the networking infrastructure in Europe to world-class.*

The AmI vision may be thought of as the convergence of several computing areas: ubiquitous computing, ubiquitous communication, intelligent user-friendly interfaces and artificial intelligence. Mark Weiser [171] describes ubiquitous computing as the concept of computers weaving "themselves into the fabric of everyday life until they are undistinguishable from it". He believes that ubiquity is the

[1]Source: http://www.tronshow.org/guidebook/2010/tron/e/u-05.html

key to providing effective, flexible, and context-aware environments, through the combined efforts of a multitude of small devices embedded in the surroundings.

An ubiquitous and reliable communication infrastructure is a key component of this vision, and it is necessary to achieve the needed cooperation of the many computational or sensing devices deployed in the environment. Moreover, seamless interoperability of heterogeneous networks and protocols is fundamental, to successfully integrate various devices and services, including wired or wireless sensor nodes, mobile terminals, gateways and Internet connectivity. The variety of AmI scenarios, ranging from the use of a single smart object, to wide networks of heterogeneous nodes, calls for the development of adaptive, multi-user, multi-system, distributed wireless communication platforms (see Fig. 2.1).

The adoption of the AmI vision can be realized only through the development and adoption of natural and user-friendly interfaces between the users and the multitude of the digital devices surrounding them. Traditional user interfaces are limited to a desktop environment and allow the user to type or point elements on a screen, getting a visual feedback for their actions. This interaction paradigm does not take advantage of the space surrounding the user or of the natural capabilities of people to manipulate objects, speak or listen in order to interact and accomplish complex tasks. The realization of the AmI vision pushes towards novel input and output modalities, to allow the user to manipulate digital data and services in mobility and when interacting with complex ensembles of devices which are often hidden in the environment. Touch surfaces and screens, gesture and voice recognition, tangible interfaces, recognition of implicit actions or user emotions, accompanied by multimodal output methods in the form of visual, auditory or haptic feedback, are key enablers for the adoption and a productive utilization of the AmI vision.

The evolution of Artificial Intelligence (AI) and its ubiquitous adoption is a common element for all the aspects and applications of the AmI vision. The presented computing areas rely on heterogeneous sensor and device networks and on the extraction of useful information from huge and noisy data streams. A variety of AI algorithms, deriving from different fields, need to be integrated and applied in order to achieve the envisioned AmI functionalities. Through the adoption of AI techniques, AmI applications can accomplish complex tasks such as interpreting the environment's state or learning about it, modeling, representing and simulation of entities or information, planning decisions or actions and interacting with humans.

Inevitably the social impact and the acceptance of such potentially powerful and intrusive technologies are topics of debate, and have been since their conception [34]. The success and acceptance of AmI by the public will depend on how secure and reliable it is and to what extent it is perceived to allow the protection of the rights and privacy of individuals.

Even if the vision emerged in Europe, in the past years this concept was rapidly spread worldwide, with many related projects and research programs. The academic interest in this area originated several international conferences, collections of papers and special issue publications. AmI has potential applications in many areas of life, including in the home, office, transport, and industry; entertainment, tourism, recommender systems, safety systems, e-health, and supported living of many different variations. In the next section some recent AmI projects will be reviewed, while further examples can be found in [147, 148].

2.1.1 Ambient Intelligence projects

The Computer Science and Artificial Intelligence Laboratory (CSAIL) at the Massachusetts Institute of Technology (MIT), together with several industrial partners, developed the *Oxygen* project [127]. Its aim is to integrate device, network and software technologies to enable pervasive, human-centered computing, to support highly dynamic and varied human activities at home, at work or on the go. In the project's vision, computational devices embedded in homes, offices and cars, sense and affect the environment, while handheld devices allow the user to communicate and perform computational tasks anywhere they are. Ubiquitous communications rely on dynamic, self-configuring networks, which help also the localization of people, services or resources. This whole platform is supported by a software layer that adapts to changes in the environment or in user requirements and helps the user to complete the needed tasks. The project team showcased several applications demonstrating the impact of the different aspects and its usage in real life scenarios.

One of the well explored areas of application of AmI projects is the home environment [71]. The *iRoom* project, sponsored from the French national research center (CNRS), built an experimental smart environment as a testbed for researchers for intelligent buildings [84]. This environment features numerous sensors and devices interconnected via a dedicated network infrastructure and is the basis for technological and sociological research. From the technological point of view, new devices and interaction modalities are studied, while sociology researchers analyze the user behavior and interactions in such a smart environment. On the other hand, the *Casattenta* project, started by the Italian T3LAB consortium in collaboration with the University of Bologna and several industrial partners, focuses on the development of an interactive and smart home environment [68]. This project targets elderly people living alone in their house, providing interactive and non-intrusive monitoring aimed at improving their safety and quality of life. The system is composed of fixed smart sensors in the environment and wearable devices, monitoring the inhabitants' health and activity to report and avoid when possible dangerous events, such as falls or injuries.

The *e-Sense* project was started by a consortium of 23 industrial and academic partners and supported by the European 6th Framework Programme (FP6) [65]. Its main contribution is the capture of ambient intelligence for mobile communications through very low power, highly efficient wireless sensors networks. The project's effort is to interface sensorized environments to a mobile wireless network, providing seamless access to context information for mobile users. Technical objectives of the project include the design of energy-efficient sensor nodes and networks, ranging from localized body area networks to vastly distributed environment sensor networks; the study of the interactions between the different networks; their integration with a mobile communication infrastructure and the adoption of efficient protocols and distributed processing paradigms.

A different approach is at the basis of the *PERSIST* project, founded by the European Union under the 7th Framework Programme (FP7) and carried out by ten research partners [137]. Here the central component is a *Personal Smart Space* which is associated with the portable devices carried by the user and which moves around with him/her, providing context-aware pervasiveness at all times and places. This personal smart space is tailored to the needs of its carrier and dynamically adapts to the surrounding environment, following and learning the users' preferences. The adopted approach has the goal to address the problem of isolated smart spaces, created by the current trends in the design of pervasive systems which lead to the formation of isolated and often non-interoperable smart environments.

SOFIA (Smart Objects for Intelligent Applications) is a three-year *ARTEMIS* project involving eighteen partners from four european countries, including our research group at the University of Bologna [151]. This project aims at the creation of an open platform to promote cross-industry interoperability for smart spaces and to facilitate the connection between the physical and the information worlds. It promotes innovation, while maintaining the value of existing legacy, and creates new user interaction paradigms tailored for the fruition of services offered by smart environments. The key contribution of the project is the proposal of a common interoperability platform (*Smart M3*) to facilitate the cooperation and the use of heterogeneous devices and embedded systems. The project already provided several large scale pilots around Europe and is building an online community to facilitate the developers to adopt innovative services and interaction approaches for smart environments.

Beside examples of projects targeting enabling technologies and platforms for AmI, there are also more specific ones, targeting the realization of particular aspects of the vision. One example is the recent European FP7 project *CuPiD*, powered by an eight member consortium led by the University of Bologna [57]. This on-going three-year project aims to provide technology-driven personalized rehabilitation exercises for people with Parkinson's disease at home. It is an example of the application of AmI technologies in the home environment and their use to provide tailored rehabilitation programs to support and enhance the clinical therapy for patients with motor disabilities. The project is creating a home-based rehabilitation system relying on wearable sensors and local real-time processing of the data, interconnected with a telemedicine infrastructure for remote supervision by expert clinicians. This platform will provide a personal health system composed by versatile sensor nodes and a central home processing unit, to enable multi-modal training and feedback for the patient, together with an user-friendly interface and a secure connection for the interaction with the

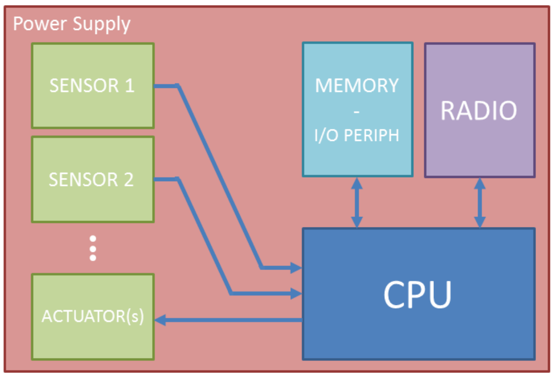


Figure 2.2: System architecture of a typical sensor node.

remote clinic center.

2.2 Hardware Architecture

This section will examine the system architecture of devices and embedded systems empowering the realization of the AmI vision, introducing the characterization of the main components and the analysis of some examples. The main building blocks and components for the development of embedded systems will be introduced, while low level analysis of integrated circuits and the optimization of silicon-level hardware architecture is out of scope of this work.

The variety of the proposed applications rises the need for very different hardware solutions, but we can outline the common characteristics and some guidelines for the designer. A typical sensor node can be identified by five main building blocks:

- Power unit
- Computational unit
- Sensing and actuation
- Memory
- Communication

A block diagram of such a system is shown in Figure 2.2. This is a very general node architecture and can be applied to the vast majority of devices used for AmI, from interactive smart objects to network nodes deployed in the environment. While the system architecture is shared, the different applications may require very different specifications and constraints for the single components. In the rest of this section we will examine the main characteristics of each of these building blocks.

Power unit

Sensor nodes are usually powered by a battery and a power supply circuit based on a DC-DC converter. The power needs of large wireless sensors network (maybe deployed in a harsh environment) are the current biggest impediment that keeps them from becoming completely autonomous, forcing them to be either connected to an external power source or have life-cycles that are curtailed by batteries. Furthermore, in applications where sensors need to be enclosed in devices with limited size or in unobtrusive wearable systems, battery size is also one of the relevant factors. For this reasons, in the last years, energy harvesting has emerged as an alternative to provide perpetual power solution for sensor networks.

Computational unit

The Central Processing Unit (CPU) is responsible of the management of all the operations of the sensor node, including the sampling of the available sensors, the processing of the data and the correct forwarding of the information when needed. The CPU should be able to manage the sensor node activity while meeting the energy consumption, size and cost constraints. There are a large number of microcontrollers, microprocessors and FPGAs suitable to be integrated in sensor nodes. Modern embedded microcontrollers, ranging from ultra low-power 8 bit solutions, to high performance 32 bit ones, are equipped with a vast range of on-board peripherals including timers, ADCs, serial communication controllers. Considering the computational and energy requirements, it is fundamental to chose the best fit for the application.

Sensing and actuation

Sensors are used to measure various physical properties such as temperature, force, pressure, flow, position, light intensity, acceleration, incident infrared radiation, etc. Sensors may be classified in a number of ways. One useful way is to classify them either as active or passive. The former require an external source of power, thus they consume power even when nothing is detected. The latter generate their electrical output signal without requiring external voltage or current. Another way to classify sensors is based on the nature of their output signal, which can be in the form of an analog voltage or they can have digital outputs. Most sensors require an output conditioning circuit to amplify and filter their output in order to be processed by a microcontroller. Typical sensor conditioning circuits include amplifier, filtering, level translation, impedance transformation for analog ones and the use of proper communication interfaces for the digital ones.

Memory

Microcontrollers used for embedded sensor nodes usually have a limited amount of on-board memory, in the range of 10-100 KByte of RAM and up to 1 MByte of nonvolatile EEPROM memory. When additional storage space is required, external modules have to be integrated in the system to fill the needs of the application. This can be done using integrated Flash modules or removable storage solutions, such as removable memory cards. The availability of additional memory can be very important for sensing devices, enabling the local log of the desired data, when real-time information is not needed. In this way the power consumptions of the node can be reduced, since the use of on-board memories is more energy efficient when compared to a continuous wireless transmission of the data.

Communication

The wireless communication channel enables the device to communicate with the external world, and to establish a WSN to cooperate with other devices. This is one of the critical components of a sensor node, because it regulates the modalities of the network communications and it usually has the biggest impact on the power consumptions of the device. Several hardware solutions and protocols have been developed to better address the needs of various sensor networks (IEE 802.15.4, ZigBee, Bluetooth, ANT to cite a few) and the optimization and development of new ones represents an active research field.

2.2.1 State of the art

In this section some of the recent sensor nodes will be presented.

TelosB

Crossbow's TelosB mote platform is an open source, low-power wireless sensor module designed to enable cutting-edge experimentation for the research community. The TelosB bundles all the essentials for lab studies into a single platform including: USB programming capability, an IEEE 802.15.4 compliant, high data rate radio with integrated antenna, a low-power MCU with extended memory and an optional sensor suite.

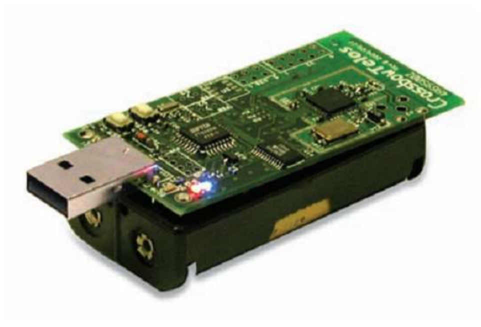


Figure 2.3: TelosB sensor node.

Mica2

The Mica2 Mote platform is a third generation device used for enabling low-power, wireless sensor networks available in 2.4GHz and 868/916 MHz. The

MICAz Mote offers a 2.4 GHz, IEEE/ZigBee 802.15.4 board and the MICA2 is an 868/916 MHz Multi-channel radio transceiver used for low-power, wireless, sensor networks. The MICA Mote platforms are fully compatible with the MoteWorks Software Platform and enable users to set up ad-hoc wireless networks.

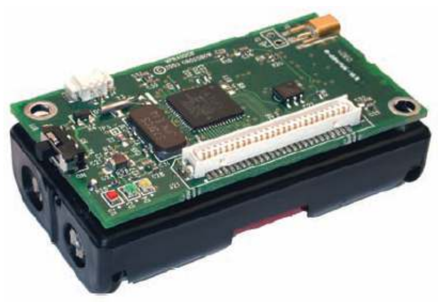


Figure 2.4: The Mica2 sensor node.

Shimmer

Shimmer, originally developed by Intel Research Labs in 2006, is a small wireless sensor platform that can record and transmit physiological and kinematic data in real-time. Designed as a wearable sensor, Shimmer incorporates wireless ECG, EMG, GSR, Accelerometer, Gyro, Mag, GPS, Tilt and Vibration sensors. Shimmer is an extremely extensible platform that enables researchers and industry to be at the leading edge of sensing technology.

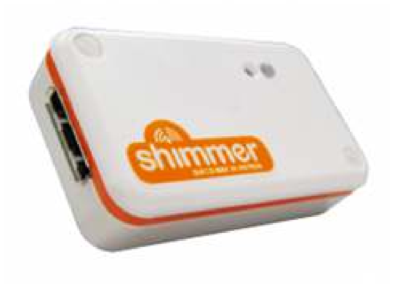


Figure 2.5: The Shimmer sensor node.

ETHOS

The ETH Onbody Sensor (ETHOS) implements a wearable sensor platform that is optimized for long-term monitoring of human body-segment orientation. It was developed in 2010 at the Wearable Computing Lab at ETH. The core component is an inertial measurement system (tri-axial accelerometer, magnetometer and gyroscope sensors). An internal temperature sensor, used for compensation of sensor drifts, can be interfaced, too. Gathered data can be stored in a raw format, or fused by an on-board DSP to estimate the orientation in an Euler-angle representation. In both cases, data can be transmitted via wired (USB) or wireless interface (ANT+) for real-time display, and stored on internal non-volatile memory for offline analysis after the recording.

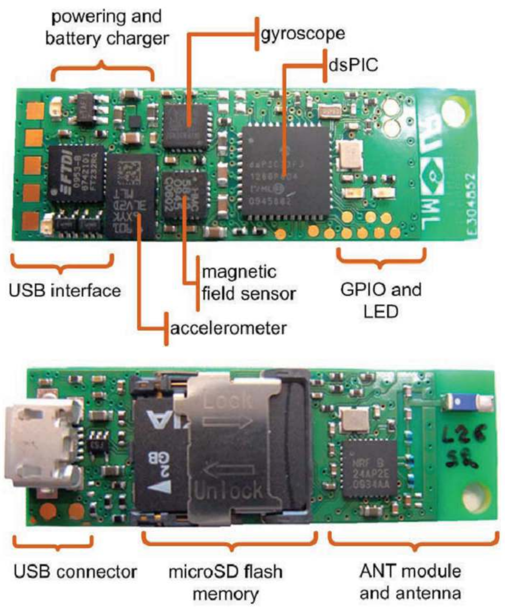


Figure 2.6: The ETHOS sensor node.

iNEMO v2

The iNemo is an inertial sensing platform from STMicroelettronics, now at its second review. It combines accelerometers, gyroscopes and magnetometers with pressure and temperature sensors to provide 3-axis sensing of linear, angular and magnetic motion, complemented with temperature and barometer/altitude readings, representing the new ST 10-DOF (degrees of freedom) platform. This 10-DOF inertial system represents a complete hardware platform which can be used in numerous applications such as virtual reality, augmented reality, image stabilization, human machine interfaces and robotics. A complete set of communication interfaces with various power supply options in a small-size form factor (4 x 4 cm) make the iNEMO v2 a flexible and open demonstration platform.

Figure 2.7: The iNEMO sensor node.

2.3 Software Layer

This section will introduce a typical data processing chain used for the analysis of sensor data. Depending on the system or the application needs, the components of this software layer are implemented on board of the device, or can be optionally left for the execution on a more capable host system.

2.3.1 Sensor data processing

Several techniques to fuse data or features captured by sensing devices take the name of pattern recognition techniques. Pattern recognition can be defined as the act of taking in raw data and taking an action based on the category of the pattern. In general a pattern recognition system establish a mapping between the measurement space and the the space of potential meanings (classes). This mapping is performed in six steps.

Sensing

Data is collected from the one or more sensors. Ususlly the microcontroller coordinates this action and uses timers for the correct temporization of the sampling process.

Pre-processing

Pre-processing include all the steps necessary to condition the signal for further processing. Typically this steps includes a filter to reduce signal noise or the conversion of sampled values in meaningful measurement units.

Segmentation

Sensors provide a continue stream of data, segmentation aim at extracting only the data related to a single entity to classify. Segmentation is one of the deepest problems in pattern recognition application on continuous streams of data like speech or gesture recognition, hand written recognition etc.

Feature extraction

This steps aim to reduce the data dimension. The objective here is to extract quantities that are distinctive of a certain class. The task off feature extraction is strictly problem and domain dependent. In general we can place a conceptual boundary between feature extraction and classification since an ideal feature extractor would yield a representation that makes the job of the classifier trivial and vice versa, a perfect classifier would not need the help of a feature extractor. Typically it is not possible to define features that are good for all problems and the developer experience plays an important role.

Classification

Classification uses the information provided by the features to assign a category to that data pattern. To perform the classification step we rely on a set of tools called classifiers. A large number of classifiers have been developed to address several problems in patter recognition. In general we can sort them into two categories.

Supervised classifiers. In supervised learning, a teacher provides a category label or cost for each pattern in a training set, and we seek to reduce the sum of the costs for these patterns. The classifier is trained offline using this set of samples. Typically training is a computational expensive operation while normal classification is much more lightweight and suited for real time operation.

Unsupervised classifiers. In unsupervised learning or clustering there is no explicit teacher, and the system forms clusters out of the input patterns. Unsupervised classifiers are used when a training set is not available or too expensive to be created. Typically they rely on a set of assumption on the underlying probability densities and assume that the only thing that must be learned is the value of an unknown parameter vector. The development of pattern classification systems rises a number of issues, many are domain or problem specific, and their solution will depend upon the knowledge and insights of the designer.

Post processing.

Exploit further context information other than from the target pattern itself to improve performance.

Other important characteristics of the sensor data processing chain include the following.

Overfitting For supervised classifiers may sound obvious the idea that a larger training set will result in a more complex, but more performing classifier. Experience showed that increasing the complexity of the classifiers may result in poorer performance during normal operation. In fact while an overly complex model may allow perfect classification of the training samples, it is unlikely to give good classification of novel patterns. This situation is known as overfitting. One of the most important areas of research in statistical pattern classification is determining how to adjust the complexity of the model.

Model Selection A huge amount of models have been developed for classification. In general is hard to know when a hypothesized model differs significantly from the true model underlying our patterns.

Prior Knowledge, Context awareness Incorporating prior knowledge and context awareness to improve the classification accuracy. However context can be highly complex and abstract and often came from different spaces than our features vectors.

2.4 Key Aspects and Challenges

The introduced background puts the fundamental basis for the on-going development of technologies and applications aiming at the realization of the AmI vision. In the last years the hard work of the research community have brought great innovation and a huge impact on our daily life, bringing us one step closer towards the visionary future imagined a decade ago. Still, each new achievement opens the way for new challenges and further studies.

Some of the technological challenges with a broad impact on the realization of this vision include energy efficiency and advanced power management methodologies for all the involved devices; development of methodologies and techniques to process and extract information from the growing amount of available data; standardization and wider interoperability of the proposed platforms to facilitate their adoption and the diffusion of applications and services targeting home and professional environments and enabling the fruition of the results of the technological advances.

The realization of a pervasive and intelligent digital environment has several key aspects which are addressed and continuously updated by the research community. They include:

Interactivity. The ultimate purpose of every digital system is to satisfy a need of its user. No matter how complex the system is, the user wants to have fast and intuitive access to the information or services needed and this should happen in a simple and natural way. We are experiencing a continuous innovation in terms of user interfaces and usability of the digital systems, and a key feature for every future technology is an intuitive and simple to use interface. This simplicity and the adoption of natural interaction paradigms is very challenging from the technological point of view and opens research opportunities spanning across every scientific community.

Networking and interoperability. The AmI vision builds upon the concept of ubiquitous computing and communication, which can only be achieved through the collaborative work of a multitude of networked devices. Information and services has to be available from anywhere and the exchange of data between heterogeneous systems and devices is a key enabler for future technologies. Interoperability between various systems, devices and services is one of the key issues that has to be addressed in the always increasing world of research and industrial solutions.

Data analysis. The increasing number of sensing devices, and their integration with our environment and our daily life, creates huge amounts of data. This data itself is usually too complex and has no direct meaning, until it is processed and transformed in useful information. Data processing techniques are assuming a growing role in our society and are becoming the enabling technology for a multitude of services and applications. The mining of heterogeneous data sources and their fusion to extract information is a key challenge affecting all the levels of technological research, from small and constrained embedded systems, to worldwide distributed data-centers.

Context awareness. A fundamental characteristics of intelligent systems is the analysis and recognition of context. It enables the system to automatically adapt to the dynamically changing status of the surrounding environment and to apply the most appropriate behavior. The context represents another layer of information, which can be used to better understand the available data. The future digital ecosystem will need an always better capability of context recognition to automatically fulfill the user's needs and to satisfy his preferences.

Energy efficiency. Every aspect and device of the digital environment we are creating today can not avoid the need for the energy to power its components. Being it a small and unobtrusive device which can disappear into our clothes or in the environment, or being it a server farm which needs to elaborate huge quantities of information, they need to carefully optimize their power consumption. Research perspectives and future challenges are more and more shifting towards the optimization of tradeoffs between performance, consumption and costs, rather than the search for ultimate results in only one of those aspects.

Chapter 3

Smart Objects for Interaction

Smart objects are physical devices enhanced by the integration of embedded electronics, which gives them sensing and communication capabilities. They can be used to introduce new interaction paradigms between the user and the digital world, allowing the user to exploit his/hers natural capabilities to manipulate objects and to interact by means of spatial motion or gestures. This chapter will introduce the development of the *Smart Micrel Cube* (*SMCube*), a cube-shaped smart object with gesture recognition capabilities, used to interact with tabletop surfaces and smart environments.

3.1 Overview

Traditional user interfaces define a set of graphical elements (e.g., windows, icons, menus) that reside in a purely electronic or virtual form. Generic input devices like mouse and keyboard are used to manipulate these virtual interface elements. Although, these interaction devices are useful and even usable for certain types of applications, such as office duties, a broad class of scenarios foresees more immersive environments where the user interacts with the surroundings by manipulating the objects around him/her.

Tangible User Interfaces (TUIs) introduce physical, tangible objects that augment the real physical world by coupling digital information to everyday objects.

The system interprets these devices as part of the interaction language. TUIs become the representatives of the user navigating in the environment and enable the exploitation of digital information directly with his/her hands. People, manipulating those devices inspired by their physical affordance, can have a more direct access to functions mapped to different objects.

The effectiveness of a TUI can be enhanced if we use sensor augmented devices. Such *smart objects* may be able to track the user's movements or recognize gestures and improve human experience within interactive spaces. Furthermore, the opportunity to execute a gesture recognition algorithm on-board of such devices brings several advantages:

1. The stream of sensors readings is not sent over the wireless channel. This reduces the radio use and extends the battery life.

2. The reduced wireless channel usage allows the coexistence of a larger number of objects in the same area.

3. Each object operates independently and in parallel with the others, improving the system scalability.

4. The handling of objects moving between different physical environments is facilitated.

5. No other systems, such as video cameras, are required to detect and classify user movements, thus the system cost is reduced.

The *SMCube* is a tangible interface developed as a building block of the *TANGerINE* framework, a tangible tabletop environment where users manipulate smart objects in order to perform actions on the contents of a digital media table [27]. The SMCube is a cube case with 6.5 cm edge. It is equipped with sensors (a digital tri-axes accelerometer) and actuators (infrared LEDs, vibro-motors). Data from the accelerometer is used to locally detect the active face (the one directed upward) and a set of gesture performed by the user (cube placed on the table, cube held, cube shaken and tap [40]). These information are wirelessly sent to the base station that controls the appearance and the elements of the virtual scenario projected on the digital media table for processing. Furthermore, through the LEDs, the node is tracked by a vision based system and can be used as a pointing interface on top of the interactive table, in a multi modal activity detection scenario.

The recognition algorithms developed in previous work rely on time invariant features extracted from the acceleration stream [40]. In a more general case the information used to recognize a gesture is contained in the sequence of acceleration rather than in some particular features. For this class of problems a different family of algorithms have been developed, relying on pattern recognition techniques. In particular, Hidden Markov Models (HMMs) have been extensively used for gesture recognition since they tend to perform well with a wide range of input modalities and with temporal variations in gesture duration.

HMMs belong to the class of supervised classifiers, thus they require an initial training phase to tune their parameters prior to normal operation. Even if the training of a HMM is a complex task, classification is performed using a recursive algorithm called *forward* algorithm. Although this process is a lightweight task compared to training, several issues must be considered in order to implement it on a low-power, low-cost 8 bit microcontroller such as the one embedded on the SMCube.

HMMs have been broadly applied to gesture recognition [106, 129], but embedded implementation on low performance interactive devices are limited to high resource mobile-devices and 32 bit microcontrollers [21]. In this work we characterize our fixed point implementation of the *forward* algorithm, highlighting the issues related to the implementation of this algorithm on low-computational power, low memory devices that can not rely on floating point arithmetic. Starting form the analysis of the standard floating point implementation of the algorithm, we propose a solution to these issues and compare the performance of our embedded solution with a standard one [182].

The HMM classifier needs to segment the gesture from the continuous sensor data steam: the start and stop of the performed gesture has to be provided to the recognition algorithm in order to classify the executed gesture. Gesture segmentation from the stream of sensor data often relies on user collaboration (e.g. pushing a button wile executing a gesture [97]) or integrates information from other types of sensors (e.g. ultrasonic [154], microphones [168]). To allow simple and natural interaction modalities, we present an algorithm for gesture segmentation with minimal effort by the user. The algorithm detects the beginning and the end of motion segments and then uses HMMs to recognize the executed gesture [125].

We characterized our implementation by evaluating the recognition performances on a set of seven gestures that a person can use to navigate into virtual spaces. Furthermore we assess the performance of our algorithm on a multi user scenario where up to four people share the same object.

3.2 Related Work

Almost two decades ago, research began to look beyond current paradigms of Human Computer Interaction based on computers with a display, a mouse and a keyboard, in the direction of more natural ways of interaction [171]. Since then concepts as wearable computing [121] and tangible interfaces [86] have been developed.

The use of TUIs has been proposed in many scenarios where users manipulate virtual environments. This have been proved to be useful specially in applications for entertainment and education [85]. An analysis of the impact of tangible interaction within a school scenario is presented in [135]. According to this work, research from psychology and education suggests that there can be real benefits for learning from tangible interfaces. An early study on different interaction technologies including TUIs has been presented in [143]. In this study the authors highlight how graspable interfaces push for collaborative work and multiple hand interaction. In [165] authors developed an educational puzzle game and compared two interfaces: a physical one based on TUIs and a screen-based one. Results show how TUIs are an easier mean to complete assigned task and have higher acceptance among the 25 children between 5 and 7 years old involved in the test.

TUIs are also used to enhance the exploration of virtual environments. Virtual Heritage (VH) applications aim at making cultural wealth accessible to the worldwide public. Advanced VH applications exploit Virtual Reality (VR) technology to give immersive experiences to the user, such as archaeological site naviga-

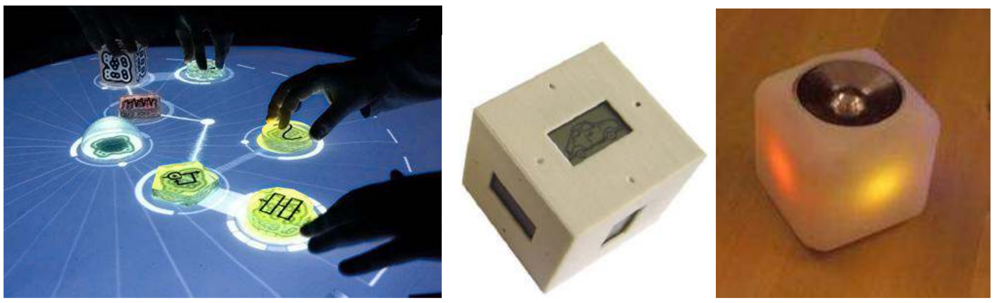

(a) (b) (c)

Figure 3.1: Examples of tangible interfaces: (a) ReacTable, (b) Display Cube, (c) Music Cube.

tion, time and space travel, and ancient artifact reconstruction in 3D. Navigation through such virtual environments can benefit from the presence of tangible artifacts like palmtop computers [67] or control objects [82]. In [136] the *Tangible Moyangsung*, a tangible environment designed for a group of users that can play fortification games, is presented. People, manipulating tangible blocks, can navigate in the virtual environment, solve puzzles or repair damaged virtual walls in an evocation of historical facts.

Interactive surfaces are a natural choice when developing applications that deal with browsing and exploration of multimedia contents. On these surfaces users can manipulate elements through direct and spontaneous actions. For example in [26] multiple users can collaborate within an interactive workspace featuring vision based gesture recognition to perform knowledge building activities such as brainstorming. On the *reacTable* [90] several musicians can share the control of the instrument by caressing, rotating and moving physical artifacts with dedicated functions on the table surface. The *TViews* is a LCD based framework where users can interact with the displayed contents through a set of TUIs (textslpucks) [124]. The *puck* is used to select and move virtual objects and its position is tracked using acoustic and infrared technologies. Another example is the Microsoft Surface Computing platform, now known also as *PixelSense* [55], where multiple users can share and manipulate digital contents on a multitouch surface.

The expressiveness of TUIs can be enhanced by the use of on-board sensors. The *MusicCube* is a tangible interface used to play digital music like on mp3 players [38]. The cube is able to understand the face pointing upwards and a set of simple gestures. This ability, together with a set of controls and buttons, is used to chose the desired playlist and to control music volume. The *Display Cube* is a cube shaped TUI equipped with a three-axes accelerometer, 6 LCD displays (one per face) and a speaker, used as a learning appliance [104]. *SmartPuck* is a cylindrical shape multi-modal input-output device having an actuated wheel, a 4 way button, LEDs and speaker and it is used on a plasma display panel [99]. SmartPuck allows multiple user interaction with menus and application and has been tested by using it to navigate within the Google Earth program in place of a mouse.

Entertainment and gaming industry adopted recently tangible interfaces for the interaction with applications and games. The Wiimote is a controller developed by Nintendo for its Wii console [56]. This controller embeds an accelerometer, an infrared camera and a Bluetooth transceiver and is used to interact with a large number of applications and videogames. Its success prompted the adoption of similar solutions by other vendors, such as the Playstation Move by Sony [152]. It embeds a set of inertial sensors and is equipped with a colored light to integrate sensor data with video tracking and enhance the interaction experience. The low price and the huge diffusion of these devices triggered an active involvement of the research community, leading to the reverse engineering of the products and their use in several research projects [174, 175].

3.2.1 Gesture recognition

Gesture recognition algorithms typically are made up of four steps: data acquisition from the sensors, data preprocessing to reduce noise, extraction of relevant features from the data stream and classification. Several design choices are available at each step, depending on the application scenario, the activities that have to be recognized, and the available computational power.

When features are time invariant (e.g. zero crossing rate or frequency spectrum), simple time-independent classifiers can be used (e.g. Support Vector Machines or decision trees). In a more general case features are time dependent, and classifiers suited for temporal pattern recognition are used. Typical approaches include Dynamic Time Warping [100], neural networks [24], and Hidden Markov Models. HMMs are often used in activity recognition since they tend to perform well with a wide range of sensor modalities [154] (they are also used successfully in other problem domains, such as speech recognition, for which they were initially developed [140]).

Even if several classification algorithms have been proposed for implementation on smart objects [88, 146], the solutions proposed to recognize gestures performed with TUIs typically rely on vision systems [92, 156], or on the collection and processing of data from an external PC [133, 169], or on the recognition of simple gestures through the analysis of time invariant features [38, 184].

Several variants of HMMs have been proposed to recognize inertial gestures: in [97] 5-state ergodic discrete HMMs are evaluated with the Viterbi algorithm to classify gestures performed with a handheld sensor device in several tasks (interaction with a TV, a presentation or a CAD environment). The work of Mantyla et al. [122] uses 7-states Left-to-Right models and the forward algorithm to classify actions performed with a mobile phone equipped with an accelerometer. Both implementations have similar performance and rely on a PC to execute all computations.

An exception to the previously cited papers is the work proposed by Ueda et al. [160]. In this work the authors present the m-ActiveCube, a physical cube equipped with sensors (ultrasonic, tactile and gyroscopes) and actuators (buzzer, LEDs and motors) that acts as bidirectional user interface toward a 3D virtual space. Multiple cubes can be connected and collaborate in achieving a defined task. They evaluate a fixed point implementation for HMMs able to perform speech recognition. Since the proposed algorithm can not be implemented on a single cube, the basic idea here is to balance the computation among several cubes. One of the main limits of this work is that the critical issue of data synchronization among different cubes that participate to the computation is not considered. Furthermore the authors assume that all the cubes always participate to the speech recognition, so every node of the network is a point of failure for the whole system. Finally the recognition ratio of the proposed algorithm is not evaluated, therefore it is not possible to evaluate the performance of this solution.

In contrast to the work presented in [160], here we present an algorithm able to recognize complex gestures and that can be implemented on a single cube with much lower computational power and memory than the ones available on the *m*-ActiveCubes. As a consequence in our solution each cube is independent from the others, hence (1) it does not need any synchronization, (2) it is not a point of failure for the whole system, (3) multiple users can operate on the table top at the same time, (4) wireless communication need is reduced (only indications of gestures are sent) resulting in longer battery life and interference reduction.

Using HMMs to classify gestures from a continuous stream of data brings another issue to solve: the recognition procedure needs to discriminate the actually executed gestures from all the other arbitrary movements. Hoffman et al. [81] use a sensorized glove to recognize hand gestures: to segment the data stream they compute the velocity profile of the sampled accelerations and apply a threshold to identify the motion segments. In [49] a Gaussian model of the stationary state is used with a sliding window approach to find pauses in movements, which identify the beginning and the end of a gesture. Amft et al. [20] presented an algorithm to recognize arm activity during meal intake, with accelerometers placed on the arm and the wrist of the user. To segment gestures they use the Sliding Window and Bottom-up (SWAB) algorithm [98] and the angle of the lower arm as the segmentation feature. While those works have focused to develop recognition solutions, none of them deals with computation or memory limited devices. We found a similar solution implemented on a wristwatch device, using a 32 bit ARM microcontroller [21], but there are no works targeting low-cost, low-power 8 bit microcontrollers, such is the Atmel ATmega168 used in this work.

3.3 Hidden Markov Models

The Hidden Markov Model (HMM) is a powerful statistical tool for modeling generative sequences that can be characterized by an underlying process generating an observable sequence. HMMs have found application in many areas interested in signal processing.

The HMMs belong to the class of Markov processes, which are models used to describe the evolution of a system. A Markov process describes a system which at any given time t can be in one of N states $S_1, S_2, ..., S_N$. At each time step, the system changes its state according to a set of probabilities associated with the actual state. The output of the process is the set of states at each instant of time, where each state corresponds to a physical event, thus we refer to this model also as observable Markov model [140].

In many cases of interest, the state of the system cannot be directly observed, but inferred though measurements of other variables called observations. This implies that the state of a system at any given time can be treated as a hidden random variable that generates the observables that we measure. A HMM is a probabilistic model used to describe sequences of observations $O = \{o_1, o_2, ..., o_T\}$ and their corresponding hidden state $Q = \{q_1, q_2, ..., q_T\}$.

Two fundamental hypotheses are given:

1. The state of the system at any given time t depends only on the state at time $t - 1$.

$$p(q_t|q_{t-1}, o_{t-1}, q_{t-2}, o_{t-2}, ..., q_1 o_1) = p(q_t|q_{t-1}) \tag{3.1}$$

2. The observable O at any given time t depends only on the state at time t.

$$p(o_t|q_t, q_{t-1}, o_{t-1}, q_{t-2}, o_{t-2}, ..., q_1 o_1) = p(o_t|q_t) \tag{3.2}$$

A Discrete HMM is characterized by the following parameters:

- A set of N states $S = \{s_1, s_2, ..., s_N\}$. Although they are hidden, often they are related to some physical significance.
- A set of M discrete observables $V = \{v_1, v_2, ..., v_M\}$ which represent the physical values observed as the output of the system.
- The state transition probability matrix $A = \{a_{ij}\} = P(q_{t+1} = s_j|q_t = s_i)$. Each element a_{ij} of the matrix defines the probability of being in state s_i at time t and in state s_j at time $t + 1$.

- The observation probability matrix $B = \{b_i(k)\} = P(o_t = v_k|q_t = s_i)$. Each element $b_i(k)$ of the matrix defines the probability of observing the symbol v_k in state s_i.

- The initial state distribution vector $\Pi = \{\pi_i\} = P(q_1 = s_i)$. Each element π_i of the vector defines the probability of being in state s_i at the beginning of the sequence.

The compact notation of a HMM is $\lambda = (A, B, \Pi)$.

Continuous HMMs differ from Discrete HMMs only because the observables can assume continuous values. In this case B is typically represented through a mixture of gaussian distributions, thus this matrix is replaced by a vector of means and one covariance matrix for each state.

There are three main problems associated with HMMs:

1. Given a sequence of observation $O = \{o_1, o_2, ..., o_T\}$ and a model λ, find the probability that the model generated that sequence $P(O_j|\lambda)$. This is also called the *evaluation* problem. The solution of this problem is equivalent to perform the classification of data.

2. Given a sequence of observation $O = \{o_1, o_2, ..., o_T\}$ and a model λ find the probability of the most probable sequence of states that generated that sequence. This is also called the *decoding* problem. Since a physical status can be associated to each state of the model, the solution of this problem is equivalent to filter out the noise on the observations.

3. Given a set of observations $O_1, O_2, ..., O_l$ find the model λ that best describes that observations. This is the *estimation* problem. The solution of this problem optimizes, by training, a model for solving problems 1 and 2.

In the following section we present the solution to the *evaluation* problem for discrete HMMs, which will be used and optimized to perform gesture classification on our smart object.

3.3.1 Evaluation problem and the forward algorithm

The most straightforward way of calculating the probability of a sequence $O = \{o_1, o_2, ..., o_T\}$, given a model λ, is trough the enumeration of every possible state sequence Q of length T. Assuming the statistical independence of observations, the probability of each sequence Q is:

$$P(O|Q,\lambda) = \sum_{t=1}^{T} P(O_t|Q_t,\lambda) \tag{3.3}$$

The final probability of the sequence can be obtained by summing the products between the probability in 3.3 times the probability of the sequence Q over all possible sequences:

$$\begin{aligned} P(O|\lambda) =& \sum_{allQ} P(O_t|Q,\lambda)P(Q|\lambda) \\ =& \sum_{q_1,q_2,...,q_t} \pi_{q_1} b_{q_1}(O_1) a_{q_1,q_2} b_{q_2}(O_2)...a_{q_{T-1},q_T} b_{Q_T}(O_T) \end{aligned} \tag{3.4}$$

However the calculation presented in Eq. 3.4 involves the order of $2 \cdot T \cdot N^T$ calculations. This is computational unfeasible, since T in most cases is in the order of hundreds or thousands of samples.

For this reason we rely on a more efficient, recursive, procedure called the *forward* algorithm. It is a recursive algorithm that relies on a set of support variables $\alpha_t(i) = P(o_1, o_2, ..., o_t, q_t = s_i|\lambda)$ and allows to find the probability that a certain model generated an input sequence $P(O|\lambda)$. It is made up of three steps.

1. Initialization: $\alpha_1(i) = \pi_i(o_1)b_i(o_1)$, $1 \leq i \leq N$
2. Induction: $\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i)a_{ij}]b_j(o_{t+1})$, $1 \leq j \leq N$ and $1 \leq t \leq T-1$
3. Termination: $P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$

The $\alpha_t(j)$ are sum of a large number of terms in the form

$$\alpha_t(j) = \prod_{r=1}^{t-1} a_{s(r),s(r+1)} \prod_{r=1}^{t} b_{s(r)}(o_r) \tag{3.5}$$

Since both the a_{ij} and the $b_i(k)$ are smaller than 1, as t becomes large, $\alpha_t(j)$ tends to zero exponentially and soon it exceeds the precision of any machine.

In order to avoid underflow, the $\alpha_t(j)$ are normalized at every step using the scaling factor $c_t = \frac{1}{\sum_{i=1}^{N} \alpha_t(i)}$. The scaled $\widehat{\alpha}_t(j) = \frac{\alpha_t}{c_t}$ are used in place of the $\alpha_t(j)$.

When using normalization we can not simply sum the $\widehat{\alpha}_t(i)$ in the termination step, since their sum is equal to 1. However we can notice the following [140]:

$$\begin{aligned}
\sum_{i}^{N} \widehat{\alpha}_t(i) = \prod_{t=1}^{T} c_t \sum_{i}^{N} \alpha_T(i) &= C_T \sum_{i}^{N} \alpha_T(i) = 1 && (3.6)\\
\prod_{t=1}^{T} c_t P(O|\lambda) &= 1 && (3.7)\\
P(O|\lambda) &= \frac{1}{\prod_{t=1}^{T} c_t} && (3.8)\\
log[P(O|\lambda)] &= -\sum_{t=1}^{T} log[c_t] && (3.9)
\end{aligned}$$

Where the use of the logarithm in the last step is necessary in order to avoid underflow, since the c_t are smaller than 1 and their product tend to zero exponentially.

3.3.2 Fixed point solution

The low power microcontroller embedded on the SMCube includes a multiplier but not a divider and does not have hardware support for floating point operations. Therefore it can efficiently compute the steps required for the forward algorithm for discrete HMM, but not the one of the standard normalization procedure. In fact, as shown in the previous section, this algorithm requires to perform N divisions each time a new sample is processed.

To find a solution suitable for our MCU we must notice that the objective of the normalization procedure is simply to keep the $\widehat{\alpha}_t(j)$ within the range of the machine. Thus it is not necessary that $\sum_{i}^{N} \widehat{\alpha}_t(i) = 1$. We propose an alternative scaling procedure [182]:

1. at each time step t, once computed the $\alpha_t(i)$, check if the highest $\alpha_t(i)$ is smaller than $\frac{1}{2}$, otherwise scaling is not needed at this step;
2. calculate the number of shift to the left l_t needed to render the highest $\alpha_t(i)$ greater than $\frac{1}{2}$;
3. shift all $\alpha_t(i)$ to the left of l_t bits.

If, at a certain time t, all the $\alpha_t(i)$ are equal to zero they are all placed at $\frac{1}{2}$ and l_t is equal to the number of bit with whom we represent our data ($datasize$). This procedure requires only shifts and can be efficiently implemented on the low power microcontroller embedded into the SMCube.

Another problem arises when we need to compute the logarithm of the c_t (see equation 3.9). However the proposed scaling procedure eases this task. In fact in this case the final probability is given by $\log P(O|\lambda) = \log(r) - \sum_{t=1}^{T} \log 2^{l_t}$, where $r = \sum_i^N \widehat{\alpha_T}(i) \neq 1$. By using $\log_2$ we already have the value of $\sum_{t=1}^{T} \log 2^{l_t}$ by keeping track of how many shifts we performed for scaling. Furthermore, we do not need to compute $\log(r)$ since logarithm is a monotonically increasing function. Thus, to compare 2 models, we simply check for the one that required less shifts for scaling, in case of tie the one with higher r is the model with higher $P(O|\lambda)$.

3.3.3 Algorithm complexity

Classification of a new instance using HMMs is performed by computing, through the forward algorithm, the probability that an input sequence is generated by each

Table 3.1: Basic operations computational complexity

Operation	**Cost**
Shift	1
Variable compare	1
Sum 8 bits	1
Sum 16 bits	4
Sum 32 bits	8
Multiplication 8 bits	4
Multiplication 16 bits	15
Multiplication 32 bits	35

Table 3.2: Algorithm complexity

Algorithm	**Cost**
α_{t+1} Calculation	$(N^2 + N)\, mul + 2 \cdot (N^2 - 1)\, sum.$
Normalization	$(N - 1) + (N + 1) \cdot (data\ size - 1)$
Single step (8 bit)	$C \cdot [6N^2 + 12N + 4]$
Single step (16 bit)	$C \cdot [23N^2 + 31N + 7]$
Single step (32 bit)	$C \cdot [51N^2 + 67N + 14]$

model associated to a gesture. The instance is classified as belonging to the class of the model that results in highest probability. Therefore, once we detected the beginning of a new gesture, each time the MCU samples a new data from the accelerometer it must preprocess the input data, execute one step of the forward algorithm with all models and normalize the $\alpha_t(i)$ of all the models.

According to the algorithm presented above, one step of forward algorithm (i.e. calculate $\alpha_{t+1}(i)$, $1 < i < N$) requires

1. the product between an $N \times N$ matrix (the transition probabilities matrix A) and the old $N \times 1$ vector of the α_t;
2. execute an element by element product of the resulting vector with the column of the observing probabilities matrix (B) associated to the output o_{t+1}.

For a total of $N^2 + N$ multiplication and $2 \cdot (N - 1)$ sums.

The scaling algorithm proposed first finds the highest $\alpha_t(i)$, than computes the number of shifts needed and finally shifts the other $\alpha_t(i)$. To execute this

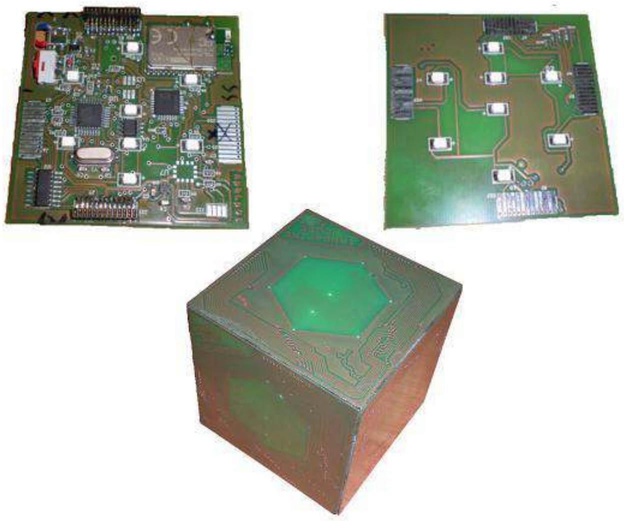


Figure 3.2: The SMCube. The cube edge is 6.5 cm long. On the top left the inner surface of the master face that includes the microcontroller, the accelerometer and the transceiver. On the top right the inner surface of the other five faces of the cube

procedure, in the worse case, we need to make $N-1$ variable comparisons to find the highest one and $datasize-1$ comparisons to find the number of shifts. Finally we perform $N \cdot (datasize-1)$ shifts.

In table 3.1 we present the computational cost of the basic operations used to evaluate the complexity of our implementation. A summary of the complexity of the steps outlined above is presented in table 3.2 (where C is the number of gestures we want to recognize).

The memory cost is given by $\frac{\text{data size}}{8} \cdot C \cdot (N^2 + N \cdot M + N)$. The models can be stored either in the MCU RAM or EEPROM.

3.4 Smart Micrel Cube

The Smart Micrel Cube (SMCube) is a cube shaped artifact with a matrix of infrared emitter LEDs on each face (see Fig. 3.2). It embeds a low-cost, low-power 8-bit microcontroller (Atmel ATmega168 [1]), a Bluetooth transceiver (Bluegiga WT12 Bluetooth module [5]) that supports Serial Port Profile (SPP) and a MEMS

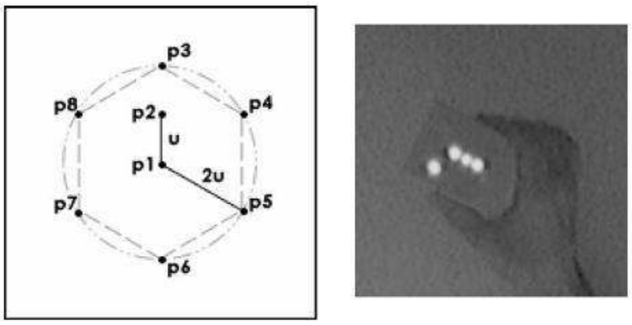


Figure 3.3: The SMCube LED patterns present on each face.

tri-axial accelerometer (STM LIS3LV02DQ [4]) with a programmable full scale of 2g or 6g and digital output. The cube is powered through a 1000 mAh, 4.2 V Li-ion battery. With this battery the cube reaches up to 10 hours of autonomy during normal operation.

The ATMega168 features a RISC architecture that can operate up to 20 MHz and offers 16 KB of Flash memory, 1 KB of RAM and 512 Bytes of EEPROM. The microcontroller includes a multiplier and several peripherals (ADC, timers, SPI and UART serial interfaces, etc.) but no floating point unit. In our prototype the CPU operates at 8 MHz. The firmware has been implemented in C using the Atmel AVR Studio 4 IDE that, used in conjunction with avr-libc and the gcc compiler, provides all the APIs necessary to exploit the peripherals and perform operations with 8, 16, and 32 bit variables. Being written in C, the code is portable on other devices. A wireless bootloader is used to load a new firmware on the cube without the need to use programmers or disassemble it. Each cube is identified by an ID number stored on the cube flash memory, which helps disambiguating when more than one cube is present on the scene at a certain time.

The LEDs pattern on every face of the cube is composed of 8 points (see Fig. 3.3). In the basic configuration only points p1, p2, p3, p5 are switched on, the remaining points are used as a binary encoding of the cube id (note that this visual id is not related to the cube id stored on the MCU flash). In addition to the LEDs, actuation can be provided through six vibromotors mounted on the cube's faces.

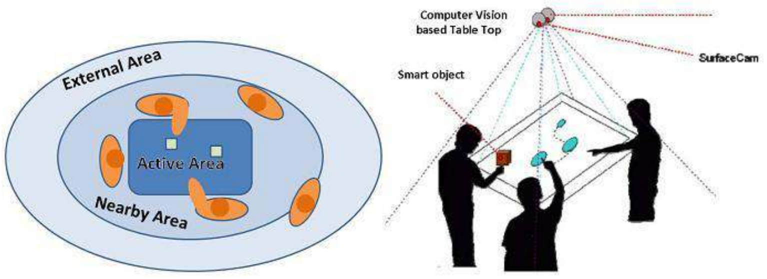


Figure 3.4: TANGerINE platform setup.

The SMCube is the main interactive device for the TANGerINE platform, a tangible tabletop environment in which users can interact with digital information manipulating tangible smart objects in order to perform actions on the contents of a digital media table [27]. The TANGerINE system layout consists of a ceiling mounted case that embeds all of the required elements: computer, projector, camera and illuminator, targeting the horizontal surface of a normal table that is positioned under the case, where also the interface is visualized (see Fig. 3.4).

The tabletop scenario is characterized by different contexts according to the area where the interaction occurs:

- **Active Context** (AC): it is the horizontal visualization surface, typically the scene where users interact with tangibles (recognized by the system) as well as digital elements. In this area there is a direct mapping between the position and orientation of tangible objects and the digital ones.
- **Nearby Context** (NC): it is the area right around the tabletop where both intentional and non intentional actions can be performed. The body of the user can be tracked and this information can be used to study his behavior. The position of the user can be useful also for attributing the ownership of actions performed in the AC. In this context tangibles position could not be precisely tracked, as a result of a less constrained user behavior in dealing with physical objects (e.g. user could occlude the object), but can still be manipulated and provide information about their orientation in space.
- **External Context** (EC): it is the outer area, unrelated with the first two

contexts. In this area no position tracking occurs, but the user can still interact with the tangible object and carry it with him across different tabletops. The object therefore becomes a bridge between different interactive artifacts. The user could perform some actions on a tabletop and use the same tangible on other artifacts, in this case the physical object can become a container of different kind of information (e.g. session data or user profile).

A combination of computer vision tracking techniques and signal analysis of the sensors embedded in the SMCube are employed to allow natural and effective interaction modalities in each of the three contexts. Gesture recognition algorithms allow the user to perform gestures with the cube in proximity of the table, which are mapped to actions on the digital content represented on the table. Computer vision techniques are applied to obtain LEDs detection and tracking in order to understand the cube's position on the tabletop surface. The analysis of LEDs pattern gives us the absolute orientation of the SMCube. The matrix of LEDs pictured in Fig. 3.3 has been designed to provide both the 2-dimensional orientation of the cube and the identification of the cube. The orientation is evaluated in relation to the absolute axis perpendicular to the table surface. The cube form-factor and the border size of the face provide enough space to avoid ambiguous detections, allowing cubes to be adjacent in every orientation.

3.5 Implementation of the Gesture Recognition Algorithm

The goal of our work was to implement the whole gesture recognition algorithm on board of the SMCube. Typical activity recognition systems are made up of four steps: (1) data preprocessing, (2) segmentation, (3) features extraction, (4) classification. At each step several design choices must be taken. Our approach was the optimization of each step for the resource-constrained microcontroller embedded in the SMCube.

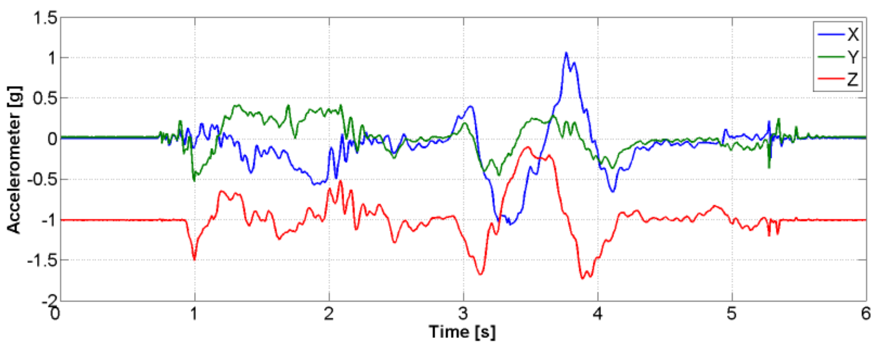
In the pre-processing stage, sampled accelerometer data are filtered with a mean filter to eliminate high frequency noise. This filter computes the mean value of the last 4 samples: this window length was chosen to use simple shift operations instead of divisions. In addition an offset filter removes the stationary gravity acceleration, measured during a calibration phase which consist in sampling data when the device is placed still on a table or held still in the user's hand.

3.5.1 Segmentation

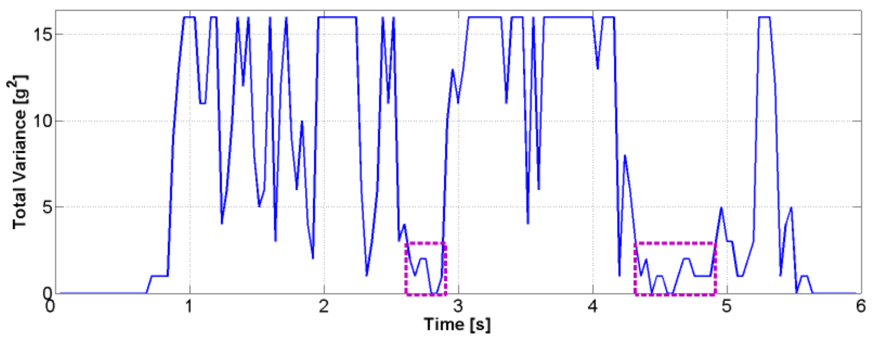
The next stage of the recognition chain implements a motion detection algorithm, used for segmentation. This step identifies the start and the end points of an executed gesture, allowing the classifier to process only the selected portions of the sampled signal in order to recognize the performed gesture.

The segmentation algorithm was implemented ad-hoc, and even if it was developed and optimized for this particular setup, it can be used in other similar scenarios. Using only the data from one accelerometer, it is very difficult to recognize gestures performed with a device if they are part of a continuous stream of unconstrained movements. To overcome this problem we added a limitation in the gesture execution: users must hold still the device for few instants before and after a gesture. In this way, it is fundamental for the algorithm to identify when the device is still in user's hand and when a movement starts and ends. To evaluate the state of the device we compute the variance of the filtered signal and use a Finite State Machine (FSM) to find out if the device is in one of the 4 possible states: still on a table, still in user's hand, in movement or shaken. The variance uses sliding windows of 4 samples, it is calculated for each axis and then summed to have a total information of the intensity of the movement.

When the cube is placed still on a surface, the variance values observed are near zero. If a user holds the device in a hand, we measure a low and uniform variance, always within a limited interval, while movements bring higher and variable variance values. The shake gesture, on the other hand, leads to extremely high variance levels for prolonged intervals of time, while the gesture is executed. An example of the sampled signals and the computed variance is shown in Figure 3.5. It is possible to classify those conditions with empirically determined threshold values and model the transitions between them as the states of a FSM. We



(a)



(b)

Figure 3.5: Accelerometer data (a) and total computed variance (b) during a sample gesture: at the beginning and the end of the plot the device is still on the table; in the highlighted sections it's hold still in the user's hand to segment the executed gesture.

introduced a few sample of delay in the recognition of a state to avoid spurious transitions.

In this way, we are able to segment every movement that is encapsulated between two states when the device is still in user's hand, which are the candidate gestures. Since all the used gestures have limited duration, we added a condition on the minimum and maximum time for the movements to be segmented as gestures. This helps to avoid many unwanted movements being segmented as gestures. Despite those conditions, this algorithm still identifies some of the random movements as gestures, leading to false positive results from the classifier, as shown later in the results section.

To improve the performance and the usability of the device, we introduced a new operation mode, called *Assisted Segmentation*. In this mode the user performs a shake gesture to enable and disable the segmentation and the recognition of all the other gestures. The shake gesture is recognized using the segmentation FSM and has a high accuracy: during our tests we obtained a correct classification ratio of 100% within 80 executions of the gesture and only one false positive. By default, gesture recognition is disabled, and the user can move the device in any way (e.g. walking with the device or using the device as a pointer on an interactive table). When needed, the user performs a shake gesture, activating the recognition algorithm and then executes the desired gestures to interact with the system. During this time the user can pay attention to the movements performed, to avoid the recognition of random movements as gestures. Another shake disables the interaction capabilities of the device, and the user can move it freely. This user-assisted segmentation technique increased the overall performance of the device, reducing drastically the number of false positive recognitions.

3.5.2 Feature extraction

To improve the performance of the classifier, we computed a quantized feature from the accelerometer data stream. The main feature used for gesture recognition is the direction of the movement, represented by the direction of the acceleration vector after the removal of the static gravity component. This information is obtained converting the 3D acceleration vector $Y_a = \{a_x, a_y, a_z\}$ in spherical coordinates, and using only the two resulting angles $\{\phi, \theta\}$.

Each sampled acceleration vector Y_a is converted to equivalent spherical coordinates $\{\phi, \theta, r\}$, as represented in the left part of Fig. 3.6. From those vectors, magnitude information r is discarded and the angles ϕ and θ are used to identify the direction of the movement performed, represented as a point on a unitary sphere. In order to cluster this data for the discrete HMMs, a quantization algorithm is applied, with k vectors or codes in the codebook. In this case, codebook vectors are uniformly distributed points on the unitary sphere, as illustrated in the right part of Fig. 3.6. Since k must be determined empirically we decided to conduct tests to find a codebook size delivering a satisfying trade-off between results and algorithm complexity.

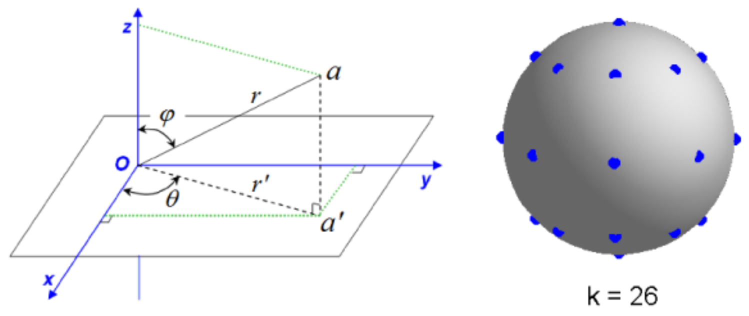


Figure 3.6: Spherical coordinates (left) and codebook vectors used for direction quantization (right).

To efficiently compute the two angles of the acceleration vector we applied an integer version of the CORDIC algorithm [167]. Using the notation in Fig. 3.6, this algorithm first estimates the phase θ and the magnitude r' of the complex number $(a_x + ia_y)$, then again estimates the angle ϕ, using r' and a_z. All computations are done with integer values, giving us a resolution of 1 degree and a maximum error of 2 degrees, which is acceptable since we are dealing with human motions and don't need higher accuracy.

3.5.3 Classification

Discrete HMMs are less computationally demanding than those operating on continuous observations, so they are the best choice in our case, since we are focusing on a limited resource implementation. As input to the discrete models we need to use a discrete feature symbol to represent the directional information. The two angles calculated, that identify the arbitrary 3D orientation of a unitary vector, are quantized to the nearest vector of the codebook by a simple minimum distance classifier. In this way, the stream of two angles is converted in a stream of codebook indices, which is a suitable input to discrete HMMs. The number of vectors in the codebook was empirically determined and a codebook with $k = 26$ vectors uniformly distributed on the spherical surface resulted to be the best trade-off between quantization accuracy and processing complexity.

The HMM training phase builds a model for each of the gestures to be recognized, using sample instances of the gestures. We used the standard Baum-Welch algorithm for this purpose, and initialized the training models with several random probability distributions. Among the resulting HMMs, those with the lowest training error were chosen.

During the training phase we collected several gestures to build and validate models for each gesture. The training was implemented on a PC in Matlab using the HMMtoolbox [2], which implements the Baum-Welch algorithm. We chose to use discrete HMMs, with 7-state Left-to-Right models for all gestures, according to the results of a preliminary exploration. For the on-line recognition a modified fixed-point version of the forward algorithm was implemented in Matlab to test its performance and in C on the AVR platform for the SMCube.

To improve the model behavior, when dealing with input gestures that are slightly different from those used during training, we modified the symbol observation probability distribution (i.e. the observation matrix B in the discrete case). A model with a uniform observation matrix B_0 recognizes every gesture with a same low probability. We interpolated the trained models with the uniform one, by weighting the observation matrices with a factor ϵ as given by the equation $B' = \epsilon B + (1 - \epsilon)B_0$. The optimal ϵ factor was empirically obtained and is equal to 0.8.

The on-line recognition algorithm evaluates the executed gesture with all of the trained models, and selects the model with the highest probability. For this purpose we used a fixed point version of the forward algorithm, as introduced in Section 3.3.2. This implementation deals with the lack of a division unit in the low power microcontroller embedded in the device, and proposes a different scaling procedure that uses shifts and a logarithmic representation of the computed probabilities.

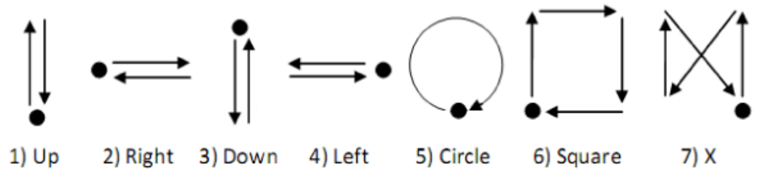


Figure 3.7: Gestures performed to validate the algorithm. The dots indicate the start and end position.

3.6 Experimental Setup and Results

3.6.1 Experimental setup

For the validation of our algorithm we used a set of 7 gestures, illustrated in Fig. 3.7. All gestures are formed by natural movements, start and end with the user holding the cube in approximately the same position and are executed on the vertical plane in front of the user, holding the device with the same orientation trough the duration of the gesture.

We collected gestures executed by four people, working in our laboratory. To build and validate the HMMs each user executed 80 instances of every gesture, during different session in different days. Even if all the performers are people in the field of computing engineering, they have been asked to perform the gestures without any particular training except a single initial visual demonstration.

To the purpose of evaluating the use of our fixed point implementation with respect to the standard one, the set of chosen gestures is not crucial. Therefore, we selected a set of movements that can be representative of the ones used to interact with computing systems. For example the gestures *Square* and *Circle* could represent actions like cut or copy on a set of selected items. On the other hand the 'X' could mean to delete a set of items and the directional movements (*Flip*) to navigate through the digital content or change the context of an application.

An analysis of the computational and memory cost as a function of the number of gestures that have to be detected is presented in the following sections.

During our test the accelerometer on the SMCube has been sampled at 100 Hz. Raw data have been sent via Bluetooth to a base PC. This enables to use this data set as a reference data set, and to later assess the effect of various types of data representation and processing through simulations.

Manual data segmentation and labeling was done by a test supervisor through a simple acquisition application running on the PC. This allows to obtain reliable ground truth, and to separate the problem of gesture classification from that of data segmentation. From each user we also collected several continuous streams, containing gestures and random movements, to simulate an actual usage of the device and test the overall recognition algorithm. No feedback from the device or the PC was given to the users during the execution of the gestures. To easily test the performance, the algorithm was implemented in Matlab, taking care to simulate the computational constrains of the 8 bit microcontroller, using only integer computations with controlled variable size.

3.6.2 Tests and simulations

Our first objective was to understand how our implementation, that uses fixed point data and the proposed scaling technique, performs with respect to a reference implementation that follows the standard algorithm and uses double precision for data representation. Therefore, we used the collected dataset to train a set of HMMs for each tester using the floating point notation with double precision. Each model has been trained using 15 reference instances, 15 loops for the Baum-Welch training algorithm, and 10 initial random models. We used twofold cross-validation to use the whole available instances for validation. Thus, the instances have been divided into two groups. Two models have been trained, each one using a different group of instances and validated on the group of instances not used for training. As a consequence we can draw our results on the whole dataset. From now on we refer to these models as floating point models and we use them in all tests to provide reference results.

The same models have been converted into fixed point notation using different accuracies (8, 16, and 32 bits) and the accuracy of these models has been compared to the one of the floating point models. Performance is evaluated using the Correct Classification Ratio (CCR), expressed as:

$$CCR = \frac{\text{number of correctly classified instances}}{\text{total number of instances}}$$

Table 3.3 summarizes the results obtained by our multi-user dataset for the different data precisions analyzed. Here we compared the performance of the standard floating point model with its fixed point implementations that use the proposed scaling technique. In addition, we evaluated the algorithm in a multiuser scenario where a single cube is shared among different users. Tables 3.3 present the CCR when the models trained on a user are validated on the other users for different implementations. This includes the single-user scenario, represented by the values on the diagonal of the tables, where we used gestures from the same user for both training and testing.

As can be seen from these results the 16 and 32 bits implementations show performance comparable to the one of the floating point implementation, while using only 8 bit for data representation results in more than 20% drop of CCR. Fig. 3.8 compares the CCR of the fixed point implementations with the floating point one. From this picture, it is clear that the 16 and 32 bit solutions show accuracies comparable with the floating point one. We can use 16 bit for data representation with minimum recognition accuracy reduction while decreasing by 50% the memory cost and by 84% the computational cost. Tables 3.4 and 3.5 show respectively the best and the worst case we encountered for the 16 bit implementation of our algorithm.

For some users we noticed that the performance of the 8 bit classifier was higher than the other implementations. This behavior is tied to the fact that HMMs are a representative model of the gestures based on the training set. Furthermore, the Baum-Welch training algorithm does not guarantee that we find the global maximum of the likelihood, but only a local one. Therefore, the error introduced by the not perfect data representation may affect the likelihood evaluation in a way that increases the recognition performance on the validation set. Overall the results show how the selected gesture recognition algorithm presents poor results when recognizing gestures from a different user than the one who trained the algorithm. However this behavior is not related to data representation but only on the classifier used in this study.

Table 3.3: Multiuser scenario classification performances with different variable sizes.

(a) Floating point implementation

Training Set	**Validation set**			
	Usr. 1	Usr. 2	Usr. 3	Usr. 4
Usr. 1	0.842	0.496	0.475	0.388
Usr. 2	0.408	0.858	0.375	0.221
Usr. 3	0.438	0.308	0.704	0.358
Usr. 4	0.333	0.317	0.254	0.663

(b) 8-bit fixed point implementation

Training Set	**Validation set**			
	Usr. 1	Usr. 2	Usr. 3	Usr. 4
Usr. 1	0.604	0.446	0.375	0.321
Usr. 2	0.446	0.825	0.354	0.208
Usr. 3	0.379	0.292	0.596	0.375
Usr. 4	0.333	0.321	0.263	0.642

(c) 16-bit fixed point implementation

Training Set	**Validation set**			
	Usr. 1	Usr. 2	Usr. 3	Usr. 4
Usr. 1	0.808	0.504	0.438	0.329
Usr. 2	0.396	0.804	0.358	0.192
Usr. 3	0.425	0.338	0.683	0.279
Usr. 4	0.329	0.235	0.254	0.604

(d) 32-bit fixed point implementation

Training Set	**Validation set**			
	Usr. 1	Usr. 2	Usr. 3	Usr. 4
Usr. 1	0.808	0.504	0.438	0.379
Usr. 2	0.391	0.800	0.358	0.208
Usr. 3	0.425	0.338	0.683	0.346
Usr. 4	0.329	0.325	0.254	0.604

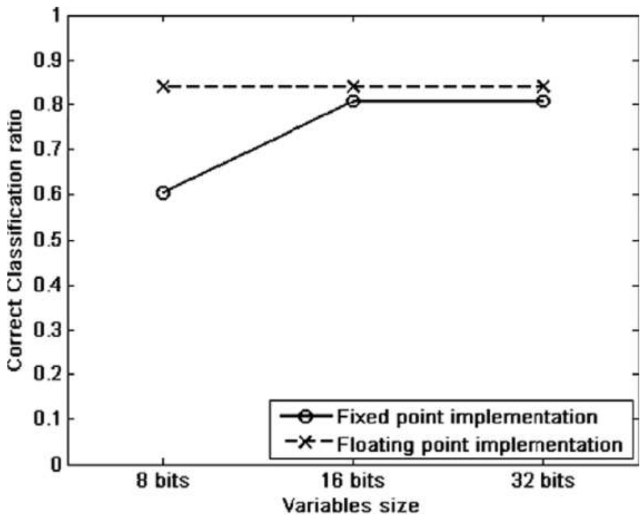


Figure 3.8: Comparison of the correct classification ratio of the fixed point implementation (continuous line) and the floating point one (dashed line) when different variable sizes are used.

Table 3.4: Classification matrix for the best case

Performed Gesture	**Classified as**						
	Up	Right	Down	Left	Circle	Square	X
Up	62	0	0	0	0	3	0
Right	0	65	0	0	0	0	0
Down	0	0	65	0	0	0	0
Left	0	0	0	65	0	0	0
Circle	0	0	0	0	65	0	0
Square	0	0	0	0	0	65	0
X	0	0	0	0	0	0	65

To overcome the limitations in the multi-user scenario, we put together all the gesture instances, regardless to the user who executed them and build a global model for each gesture. These models were trained using 15 randomly chosen gestures and validated on 200 gestures from all the users. The results of this case are presented in Table 3.6. In this case we have a notable improvement when the deivce is shared among multiple users, having a global model which can suite all of them.

With this global model we evaluated the overall algorithm performance when

Table 3.5: Classification matrix for the worst case

Performed Gesture	Classified as						
	Up	Right	Down	Left	Circle	Square	X
Up	33	0	0	0	1	28	3
Right	0	25	0	30	6	1	3
Down	12	1	43	1	0	7	1
Left	0	3	0	60	1	1	0
Circle	0	0	0	5	42	15	3
Square	0	0	0	0	13	50	2
X	3	0	15	3	2	3	39

Table 3.6: Classification matrix for the global model trained and tested using gestures from all users

Performed Gesture	Classified as						
	Up	Right	Down	Left	Circle	Square	X
Up	194	0	3	0	2	1	0
Right	0	187	1	11	0	1	0
Down	1	0	199	0	0	0	0
Left	0	1	0	199	0	0	0
Circle	0	0	0	4	177	19	0
Square	0	0	0	0	13	187	0
X	3	0	12	0	1	2	182

the device is used in a continuous way and the gestures are segmented from random movements. For this purpose we used the collected continuous streams of data which contain gestures and random movements. In this way we could test the segmentation and recognition algorithms together. Table 3.7 presents the results of this analysis.

The automatic segmentation algorithm has good performance in recognition

Table 3.7: Global model continuous recognition performance

	Auto Segmentation	Assisted Segmentation
Executed Gestures	83	78
Correctly Classified	71	62
Insertions	45	2
Deletions	6	5

executed gestures, but gives also several false positive results, identified by the insertions. The performance depends on what the user is doing and how the device is moved: long and continuous movements are easily rejected, but short movements, similar to gestures, trigger the recognition algorithm leading to a false positive. Deletions indicate how many times the algorithm misses a gesture, and this happens only if the gesture is executed too quickly or too slowly. Minimum and maximum duration times are derived from the collected dataset, and deletions may happen only in extremely short or long gestures.

To improve the device usability we proposed the *assisted segmentation* algorithm, which lets the user disable the recognition of gestures when not needed. Classification rates for this algorithm are the same of the automatic one, since it uses the same HMMs, but in this case we have a better segmentation performance, with almost no insertions.

Processing Performance Results

All the tasks needed for the gesture recognition algorithm were implemented on the ATmega168 microcontroller, embedded in the SMCube. Table 3.8 presents the computational costs needed to perform the main operations at each frame with the 16 bit implementation, showing the CPU cycles and corresponding execution times when using a clock frequency of 8 MHz. Each gesture model requires 3 matrices of 16 bit variables and with the implementation choices (7-state models with a 26 vector codebook) we need 462 Bytes to store each model. The microcontroller used has only 1 Kbyte of RAM, so we stored the models in the 16 KB of FLASH memory, used as program space. The entire application uses up to 12480 bytes of FLASH and 360 bytes of RAM memory.

Table 3.8: Computational costs with 16 bit precision

	CPU Cycles	Time (ms)
Preprocessing	254	0.034
Segmentation	1504	0.203
Feature extraction	1262	0.171
HMM (1 gesture)	5375	0.729
HMM (7 gestures)	36900	5.000
Total	45295	6.138

3.7 Conclusions

The popularity of TUIs, physical objects used for human-computer interaction, is growing together with the development of VR applications and smart spaces. Effectiveness of TUIs can be enhanced by the use of smart objects able to sense their status and the gestures that the user performs with them. The on-board recognition of gestures therefore will play a central role in developing new TUIs in order to improve object batteries lifetime, system scalability and handling of moving TUIs.

This work presented and characterized an implementation of the HMM forward algorithm suitable for the class of low-power, low-cost MCU typically embedded into TUIs. HMMs are state of the art algorithms for gesture and speech recognition. The proposed solution was implemented on the SMCube, a tangible interface developed as a building block of the TANGerINE framework. The characterization of our algorithm in both single and multiuser scenarios demonstrates that the use of fixed point data representation results in recognition ratio comparable to the floating point case when using more than 16 bits.

We evaluated the computational and memory cost of implementing a solution able to recognize up to seven gestures with a set of HMMs. We show that the flash memory available on the SMCube is enough to store all versions of the model, and that if fast recognition capabilities are needed we can use HMMs with a lower number of states without excessive recognition loss. The recognition algorithm can be improved for a multi-user scenario employing a global model trained with gestures from various users.

The automatic segmentation algorithm proposed effectively finds executed gestures, but introduces also false positives. To address this issues, we introduced an assisted segmentation mode, which disables the gesture spotting algorithm when not needed, through the execution of a shake gesture.

Chapter 4

Multi-sensor Tracking of Interactive Devices

In the previous section we introduced the SMCube, a sensorized smart object used for gestural interaction. Its main characteristic is the on-board processing of the accelerometer data stream in order to recognize gestures executed by the user. In this section we will present a new interactive device, the *SmartPen*, which was developed to satisfy a different range of interactive possibilities. In this case we are interested in accurate real-time tracking of the position and orientation of the device, rather than in the recognition of single gestures. This information, provided by the fusion of on-board inertial sensors and a computer vision system, allows for the use of the pen as a free-hand interactive tool for input or editing in 3D applications. The *SmartPen*, supported by a reconstruction and visualization layer based on subdivision surfaces, can be utilized for rapid prototyping of 3D shapes and objects in CAD applications or for the editing of existing virtual models. To achieve accurate tracking of the input device, state of the art data fusion techniques were implemented, both on the device itself, to combine the data from its multiple inertial sensors, and on the host PC to combine sensor data with video tracking information.

4.1 Overview

Innovative interaction paradigms are one of the main goals of the AmI vision. While most of the efforts are oriented towards new scenarios and the interaction between the user and an ubiquitous digital ecosystem embedded in the environment, interesting research is also targeting the more consolidated desktop computing paradigm. Here the adoption of new technologies and advanced processing techniques allows innovative interaction modalities, which can introduce new functionalities or simplify the existing ones for specific applications. One notable example is the case of Computer Aided Design (CAD) applications, where the user draws and edits complex 3D virtual models or animations on a PC. In this case it is evident how traditional input methods, such as keyboard and mouse, can limit the user experience. New technologies and interaction paradigms can enable the user to exploit the physical space and his/hers manual abilities to ease the process of the creation or editing of 3D virtual models.

In the field of CAD, reverse engineering has become an effective method to create a 3D virtual model of a desired physical object. Reverse engineering has many applications in different fields, ranging from medical imaging to web commerce and all these applications can take advantage in different ways from the reconstructed 3D virtual model, e.g. for inspection, quality control, rapid prototyping or animation [163].

The traditional reverse engineering process is performed in two sequential steps, the measurement of the physical object and its reconstruction as a 3D virtual object. The physical object can be measured using 3D scanning technologies such as coordinate measuring machines or computed tomography scanners, which provide outputs in the form of an unstructured point cloud [105]. It forms a large set of vertices in a three dimensional coordinate system, which lacks topological information and therefore is generally not directly usable in most 3D applications. The second step of the reconstruction of the 3D virtual object from the dense point cloud is an inverse problem and generally does not admit a unique solution. The overwhelming number of points acquired and the lack of topological information in this data, combined with the presence of noise and inaccuracies, usually require complex and time-consuming solutions [158]. Moreover, the strict separation of this two fundamental steps makes this process non-iterative and non-interactive.

In this work, we introduce a wireless low-cost pen-like device, the *SmartPen*, designed and produced to support a reverse engineering system, named FIRES (Fast Interactive Reverse Engineering System) [33]. This solution enables the user to naturally interact with a CAD application for a fast and simple reconstruction and editing of virtual 3D models representing physical objects. The *SmartPen* has been designed to produce a low cost, low power consumption, wireless pen-like device, able to provide intuitive and easy 3D input/editing experience. The system tracks the real-time position of the device, which is used as a 3D user input and can be freely manipulated in the space in front of the user. It exploits the user's natural capabilities of interaction and its manipulative skills, by augmenting a familiar tool such as a pen, to bridge the gap between the physical and digital worlds. It is used for drawing and selecting points and curves in the virtual environment, introducing a natural way to draw and edit the style-lines of a physical object. The style-lines define the most characteristic curves of the shape to be modeled and they are traced by simply dragging the pen in space. If a physical version of the object to model is available, the pen can be used to trace its style-lines following the ones on the real object and thus to reconstruct its shape in the digital world.

Although the 3D virtual model acquired with FIRES and the *SmartPen* does not have the accuracy of more sophisticated reverse engineering systems, our approach allows us to significantly shorten the acquisition time and the immediate integration of an interactive model directly into the virtual environment. The work presented in this dissertation is aimed at an in-depth description of the acquisition and interaction steps in FIRES, using the novel *SmartPen* and the Minoru 3D stereo video acquisition system [11], to document the integration of the system. Details on the first developments of FIRES are documented in [33], and the refinements of the geometric modeling techniques used in the reconstruction of the virtual models are presented in [130].

4.2 Related Work

Computer modeling of 3D geometries using alternative three-dimensional user interfaces (3DUIs) and interaction techniques has received considerable attention in recent years. While a number of techniques involving 3DUIs, haptic devices and virtual or augmented reality (VR/AR) systems have been proposed, the usability of 3DUIs in many real world CAD applications is still surprisingly low [157].

Pantographs are the most widely utilized 3DUIs, with several commercial examples available: some examples are the Microscribe 3D digitalizer from Ghost3D [10] or the Phantom haptic device from Sensable [13]. These solutions are limited by their form factor, have a fixed support which limits the movements and the usability of the device. Research in 3DUIs has addressed the design of novel 3D input or display devices, new methodologies for the use of existing solutions and the development of design and/or evaluation approaches specific to 3DUIs [35, 72].

Much of the early work on 3DUIs focused on systems for inferring plausible 3D free-form shapes from visible-contour sketches, which involves the difficulty of interpreting 3D information from 2D input [83, 96]. The emphasis in such systems is to quickly generate a reasonable 3D shape rather than a precise modeling of the object. In gesture-based techniques designers' strokes are used for editing exiting primitive objects into the desired shape [183]. A number of template-based methods have also been proposed, where the desired 3D form is obtained by deforming an underlying 3D template, such as for example a six-faced topological template [94, 128] or a given network of curves [58]. Recently, a system for designing free form surfaces from a collection of projected 3D curves inserted through a 2D line drawing sketching system has been presented in [131]. This approach lacks from a direct control on the object shape since a functional optimization is used to construct the smoothed surface. In contrast to the presented solutions, our approach has the advantage to use the same tool for real 3D sketching, editing and interaction within a CAD environment. When coupled with the FIRES reconstruction engine it provides an intuitive and powerful interactive environment for the designer.

Recent appearance of low-cost video cameras that include RGB and depth sensors, such as Microsoft's Kinect [3], led to several works trying to reconstruct virtual models of objects and environments from combined color and depth images. For example the work in [170] presents a system that allows users to scan objects by simply turning them freely in front of a real-time 3D range scanner. They address the well-known loop closure problem: when performing a full scan around the object, the accumulation of registration errors leads to an offset at the scanning borders, resulting in visible artifacts. Instead of ignoring loop closure cases, their approach explicitly detects and compensates for them on-the-fly by deforming the on-line model appropriately. This leads to a robust on-line algorithm which eliminates the need for post-processing and allows the user to control the coverage and quality of the reconstruction, making the process fast and intuitive. Building on this work, [87] implemented *KinectFusion*, a GPU-accelerated algorithm to rapidly create detailed 3D reconstructions of an indoor scene. They demonstrate core uses of *KinectFusion* as a low-cost handheld scanner, and present novel interactive methods for segmenting physical objects of interest from the reconstructed scene. The paper shows how a real-time 3D model can be leveraged for geometry-aware augmented reality (AR) and physics-based interactions.

Literature papers present also several applications based on pen-like hardware, focused on different fields such as handwriting, gesture recognition and HCI. Different products are already available on the market, with most of the solutions for off-line handwriting recognition like in [6], where a tiny video sensor is embedded inside the pen. Instead, the VPen from OTM [12] includes a laser diode, detectors and optics to converts handwriting to ASCII text supporting Latin and Asian characters.

An interesting work about a pen equipped with a three-axes digital accelerometer for handwritten character recognition is presented in [52], where Hidden Markov Models are used for classification. In [108] the recognition of pen gestures on a paper is evaluated to create an interactive system and link specific content on paper with digital operations or edits. In [80] an optical 6-DoF tracker is designed using a few photo-sensors that can track the position and orientation of a LED cluster positioned on top of a handheld device. It is based on a source localization problem, where the source is a custom interactive device equipped with accurately oriented LEDs. They present some applications for the proposed device, such as interactions with a TV viewing environment, a tabletop surface or a smart space. In [144] a multi-sensor approach, based on electromagnetic tracking and inertial sensors, is used to improve the tracking of an endoscopic surgery instrument. This is not a CAD application, but deals with a similar tracking problem for HCI and has an interesting approach on sensor fusion to increase the precision of the system. Our system employs a hybrid approach, fusing stereo-vision tracking information with inertial sensors to increase the performance of the tracking of the interactive device. One of the objectives of FIRES was to pursue a low-cost technology strategy to achieve an optimal compromise between accuracy and cost. Moreover, an active involvement of the user in the acquisition process has different advantages. For example, it allows for a fast interaction by adding, modifying or discarding measures right during the acquisition process, and the detection of features of the objects like creases, corners and symmetries. In addition, the interactive capabilities of the device allow the user to manipulate the virtual model and to further edit it adding or changing some of its components.

4.3 Sensor Data Fusion

A concise definition of *Data Fusion* have been proposed to highlight the fact that similar problems of data association and combination occur in a wide range of engineering, analysis, and cognitive situations. According to this definition data fusion is the process of combining data or information to estimate or predict entity states [153]. Often we refer to data fusion also as *Sensor Fusion*. In this case we refer to the use of techniques that combine data from multiple sources (sensors or high level inferences), and related information from associated databases, to achieve improved accuracies and more specific inferences than could be achieved by the use of a single sensor alone [76]. The concept of multi-sensor data fusion is not a novel idea. Humans and animals use multiple senses to improve their ability to survive. Nowadays the development of new sensors, hardware and processing techniques make real-time fusion of data possible. The use of advanced embedded microcontrollers, in association of a variety of sensors, allows for on-board processing of sampled data directly in the sensing devices. In this way only the estimated high level information needs to be sent to the collected center, aggregating the information from the single sensor data streams. Fusing data from multiple sensors offers some advantages over standard algorithms [120]:

1. Improved confidence due to complementary and redundant information;
2. Robustness and reliability in adverse conditions;
3. Increased coverage in space and time;
4. Better discrimination between hypotheses due to more complete information;
5. System being operational even if one or several sensors are malfunctioning;
6. Possible solution to the vast amount available information.

4.3.1 Direct fusion of sensor data

We can distinguish between two main cases in date fusion: Direct fusion of sensor data and fusion of higher level features or information. Direct fusion of sensor data refers to the combination of input signals from a (heterogeneous) group of sensors in order to provide an output aggregated signal of greater quality. The output of the system can be of the same form as the original signals or it can consist in quantities derived from the sampled signals. The signals from sensors can be modeled as random variables corrupted by uncorrelated noise, and the fusion process can be considered as an estimation procedure. In the case of feature or information fusion, sensor data from heterogeneous sources id first processed to obtain meaningful features. Then a fusion technique is applied to conveniently combine information from the several sources in order to obtain more accurate global knowledge from the whole system. The two approaches can share the same techniques, but the information processed and obtained have different characteristics.

Predictive filters are widespread tools in modern science. They perform state prediction and parameter estimation in fields such as robotics, computer vision, and computer graphics. They belong to the class of Bayesian filters, since they apply the Bayesian rule of conditional probability to combine a predicted behavior with some corrupted indirect observation [74]. As compared to the other types of fusion, fusion of sensor data requires a higher degree of synchronization between data streams from the sensors. The most common techniques for this kind of fusion consist of weighted averaging and Kalman filtering.

4.3.2 Kalman filter

In 1960, R.E. Kalman published a paper describing a recursive solution to the discrete data linear filtering problem [91]. Since that time, due in large part to advances in digital computing, the Kalman filter has been the subject of extensive research and application. The Kalman filter is the simplest example of a predictive filter. It represents uncertainties as Gaussian random variables, fully described by a mean and a covariance matrix, and models the system with linear dynamics and observations. Since Gaussians are preserved under linear transformation, the Kalman filter's implementation uses only linear algebra operations. It can be shown that the Kalman filter is an optimal recursive data processing algorithm. One aspect of this optimality is that it uses all information that can be provided to it. It processes all available measurements, regardless of their precision, to estimate the current value of the variables of interest. Furthermore it uses the knowledge of the system and measurement device dynamics, the statistical description of the system noises, measurement errors, and uncertainty in the dynamics models, and any available information about initial conditions of the variables of interest.

The Kalman filter addresses the general problem of trying to estimate the state of a discrete-time controlled process that is governed by the following set of linear equations:

$$\begin{aligned} x_k &= Ax_{k-1} + Bu_{k-1} + w_{k-1} \\ z_k &= Hx_k + v_k \end{aligned} \tag{4.1}$$

Where x_k is the state of the process and z_k is the, noisy measurement observed to estimate the state of the process, w_k and v_k represent, respectively, the process noise and the measurement noise and are assumed to have a normal probability distribution with the following parameters:

$$\begin{aligned} P(w) &\approx N(0, Q) \\ P(v) &\approx N(0, R) \end{aligned} \tag{4.2}$$

Where Q is the process noise covariance matrix and R is the measurement noise covariance matrix. A, B and H are the equations that specifies how the state of the process evolves and is related to the measurement. The filter estimates a process by using a form of feedback control. The process evaluates the state at some time and then obtains feedback in the form of (noisy) measurements. A set of time update equations (or predictor equations) are responsible for projecting forward the current state and error covariance estimates to obtain the a priori estimates for the next time step. The measurement update equations (or corrector equations) generate a feedback used to incorporate a new measurement into the a priori estimate and obtain an improved a posteriori estimate.

Equations 4.3 and 4.4 present respectively the prediction and the correction equations

$$\begin{aligned} \hat{x}_k^- &= A\hat{x}_{k-1} + Bu_{k-1} \\ P_k^- &= AP_{k-1}A^T + Q \end{aligned} \tag{4.3}$$

$$\begin{aligned} K_k =& P_k^- H^T (H P_k^- H^T + R)^{-1} \\ \hat{x}_k =& \hat{x}_k^- + K_k (z_k - H \hat{x}_k^-) \\ P_k =& (I - K_k P) P_k^- \end{aligned} \tag{4.4}$$

Where $\hat{x}$ is the estimated value of the state x, $P = E[e_k, e_k^T]$ is the error covariance matrix where the error is calculated as $e_k = x_k - \hat{x}_k$, and K_k is the Kalman gain that decides how much the a priori estimates should be corrected by the k-th observation (note how the large the measurement noise, R, the smaller the correction).

The Kalman filter can be used to fuse multiple measurements to estimate the desired state of the process. Every available measurement will be weighted by its portion of the measurement noise covariance matrix R, leading to an optimal fusion of the different sources. In the actual implementation of the filter, the measure of the noise covariance matrix, R, is generally possible prior to the operation of the filter, since we should be able to measure the process to estimate its state. The determination of the process noise covariance, Q, is generally more difficult as often we do not have the ability to directly observe the process we are estimating. For this reason we rely on an off line tuning of the parameter of the filter. This tuning often take the name of system identification or training.

If the hypothesis on the linearity of the process and the probability distributions of the noise are respected, the Kalman filter can provide and exact solution for the estimation of the process. However, one of the main criticism to the Kalman filter is that the hypothesis on the linearity of process and on the noises models are too restrictive. For this reason several variants of the filter have been proposed:

- Extended Kalman filters.
- Unscented Kalman filters.
- Particle filters.

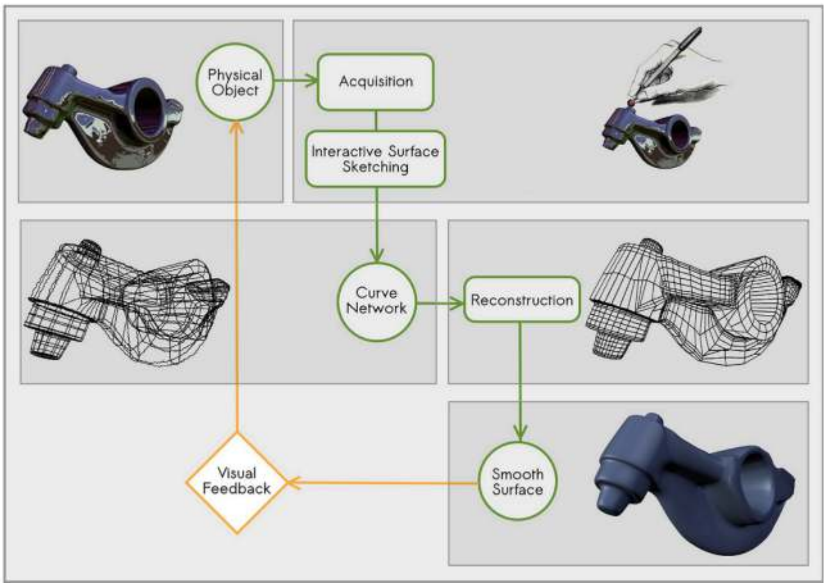


Figure 4.1: FIRES reconstruction pipeline.

The objective of this models is provide an approximate solution for an exact model rather than an exact solution for an approximate model.

4.4 System Architecture

In this section we illustrate the technical characteristics of the *SmartPen* and how the FIRES system takes advantage of its features in the reconstruction process.

The FIRES working pipeline (schematized in Figure 4.1), differently from existing reverse engineering solutions, integrates the steps of measuring and reconstruction into an iterative and incremental process that allows the user to have a real time visual feedback on the ongoing work. The measuring step is achieved through an stereo vision system provided by the Minoru 3D webcam and the *SmartPen*, equipped with 4 infrared light emitters and inertial measurement sensors. The 3D position of the *SmartPen* is tracked by the fusion, on the host PC, of video data from the stereo rig and the IMU data from the pen. The raw IMU sensor data is first processed on board of the device, sending to the PC only the needed features. To reconstruct a desired object the user can intuitively

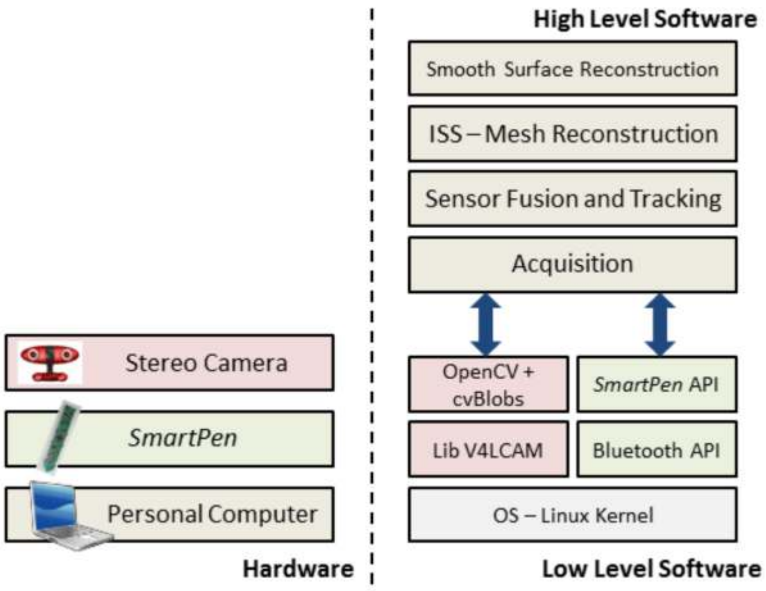


Figure 4.2: Hardware and software layers in FIRES

draw and refine the style lines of the object, i.e. the lines and curves that principally characterize the object's shape. This set of 3D curves is called the Curve Network and the process of interactively and incrementally drawing the curve network is called Interactive Surface Sketching (ISS). The ISS produces a curve network which is internally represented as a polyline mesh, that is, a mesh with faces, vertices and edges augmented with polylines associated to each edge. The reconstruction of a smooth surface from the polyline mesh is achieved through a multi-step process based on subdivision surfaces [33].

The active involvement of the user in the acquisition process has different advantages. For example, it allows for a fast interaction by adding, modifying or discarding measures right during the acquisition process, and the detection of features of the objects like creases, corners and symmetries. Moreover, using a specific set of tools, the user naturally provides topological information on the object to be reconstructed. As a consequence of this, the reconstruction step is simplified with respect to the classical reverse engineering approaches.

FIRES is able to work with different hardware configurations allowing a versatile setup. The minimal required hardware devices could be any active stereo vision system capable of real-time 3D tracking of the tip of a pen-like device.

Thus, any commercial or ad-hoc technology can be used, as long as it provides the right input to the ISS layer. The hardware and software layers involved in the FIRES system, with the new configuration with the Minoru webcam and the *SmartPen* are illustrated in Figure 4.2.

4.4.1 Hardware setup

The acquisition step in FIRES (see Figure 4.1) consists in a set of techniques that, combining a custom designed smart-pen and a stereo optical tracking system which exploits commercial infrared cameras, realize the 3D input system. The system, is composed by a stereo webcam (Minoru 3D), the SmartPen and a PC to run the tracking algorithm and the CAD application.

The *SmartPen* is the main interaction tool of the system and has been designed for this purpose. It's a pen-shaped tool meant to be freely used in the space in front of the user, as a three-dimensional user interface. The device, pictured in Figure 4.3, is an enhanced wireless inertial measurement unit (IMU) and is composed by a microcontroller (STM32F103, 32 bit ARM Coretex M3), inertial and magnetic sensors (ST L3G4200D digital gyroscope and ST LSM303DLH integrated digital accelerometer and magnetometer), a Bluetooth transceiver (Bluegiga WT12) and a set of Infra-Red LEDs on each side of the device. The IR-LEDs are tracked by the stereo camera, as will be shown in the following Section, and the inertial sensors are used to enhance the tracking performance.

The *SmartPen*, beside its use for acquisition and reconstruction purposes, is used as the main interaction tool with the proposed CAD application. That is, the user can intuitively control the virtual camera of FIRES directly moving the smart-pen around the scene pointing the pen tip towards the region of interest. Automatically, the system takes the position and direction of the smart-pen as respectively the origin and the virtual camera direction. In addition, it can be used to perform selection commands on the displayed scene and to edit the desired elements.

The Minoru 3D [11] is a low-cost stereo webcam with an USB connection. It's an unique device since it's the only consumer low-cost integrated stereo webcam

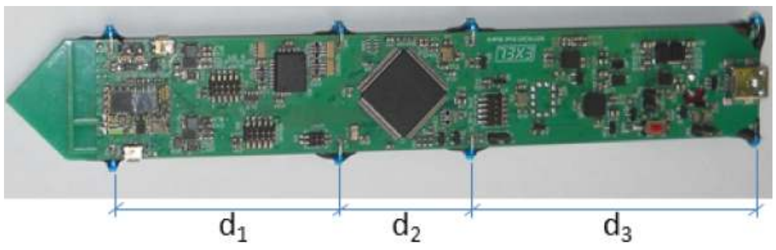


Figure 4.3: The *SmartPen* prototype with the distances between the four LEDs.

available on the market. The two video streams share the same USB connection to the host PC, limiting the total available bandwidth. With this setup it is possible to acquire the two images at a resolution of 320x240 pixels at 30 fps or 640x480 at 15 fps. To better track the *SmartPen* and take advantage of the IR-LEDs on the device, we modified the webcam and removed the IR-blocking filters present between the sensors and the lens, and introduced a simple visible light filter formed by a photographic film. An example of the raw output of the camera can be seen in Figure 4.5(a). Opportunely adjusting brightness and contrast settings of the acquired images, it is possible to obtain a video stream of just the led sources visible in the camera field of view. With this approach we obtain images with just the IR-LEDs visible and segmented from a black background, as shown in Figure 4.5(b).

On the host PC running Linux Ubuntu, a OpenGL-based application collects data from the cameras using the V2LCam library and from the pen Using the standard Bluetooth APIs and a custom protocol developed to communicate with the device. The position of the device is then computed and the 3D representation of the visible scene is represented, along with the device's position and some tracking information.

4.5 Data processing: from image points to 3D curves and surfaces

The raw data gathered from the cameras and the *SmartPen* are processed in the acquisition pipeline schematized in Figure 4.4. The projected led image points of

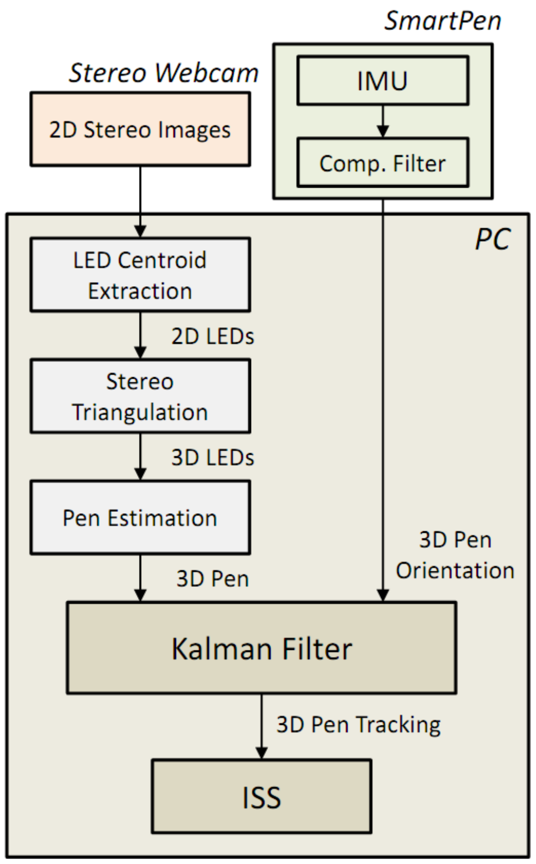
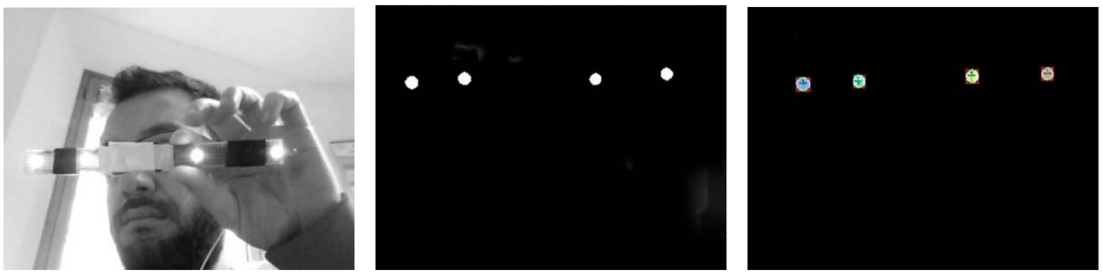


Figure 4.4: Block diagram of the Acquisition and processing system.

both cameras are first matched and then triangulated in order to produce the 3D point representing the position in space of the LEDs mounted on the *SmartPen*. After their identification and after discarding eventual noisy measures, the 3D points are used to estimated the position of the pen-tip, used as the main reference point. The sequence of the pen-tip's 3D positions in time represents a curve and is shown to the user after fusing the video estimate of the pen position and orientation with the orientation of the device estimated from IMU data.

4.5.1 LED centroid extraction and triangualtion

Starting from the pair of images acquired by the Minoru webacm, which are binarized with an opportune threshold to report just the visible IR sources, a



(a) (b) (c)

Figure 4.5: Video acquisition steps: .

blob detection algorithm is applied to estimate the 2D image coordinates of the LED centroids. This step is achieved with the openCV-based library CVblob [8]. The 2D coordinates of the leds in the two images are then triangualted using a standard stereo vision approach, to estimate their distance from the camera and to reconstruct their position in the three-dimensional space. To ensure an optimal application of the stereo algorithm, the video system has to be calibrated using a known pattern displaced in the space in front of the cameras. Both algorithms are well known in literature and we used the open source openCV implementation. Figure 4.5(c) shows the output of this step, with the image captured from one of the cameras and the superimposed blob segmentation and centroid estimation as computed by the algorithm.

4.5.2 Pen tip estimation

The triangulation step outputs the coordinates of the 3D points $\bar{\boldsymbol{p}}_i,\ i \leq 4$ computed from the ordered pairs of corresponding image points provided by the cameras.

As regards the active-pen tip estimation, we will proceed as follows. The redundancy of the number of LEDs on the active-pen has been designed in order to guarantee that even in case of occlusion of the pen tip we are always able to estimate its position. This estimate can be evaluated by knowing at least the positions of two LEDs, which allows us to identify the direction of the pen. The fourth LED has been included to handle the occlusion of at most two LEDs.

In particular, given the distances d_1, d_2, d_3 between the four LEDs and the distance between the pen tip and the tip LED (see Fig. 4.3), we call l_1, l_2, l_3, l_4 the cumulative distances between the LEDs starting at the tip LED, in other words the coordinates of the LEDs in a mono-dimensional reference system with center in l_1.

For every pair of captured images, the number of visible and identified LEDs can vary. If we call δ_i the discrete value, which is 1 if the i-th LED is visible, and 0 otherwise; then, at each frame, we have a quadruplet $(\delta_1, \delta_2, \delta_3, \delta_4)$. For each quadruplet with at least two $\delta_i = 1$, which represent a particular configuration of at least two visible and identified LEDs, we preliminarily compute and store the centroid $C \in \mathbb{R}$ of the LED positions l_i, that is $C = \sum_{l=1}^{4} l_i \delta_i / \sum_{l=1}^{4} \delta_i$ and its distance T from the pen tip.

At each frame, the position $\boldsymbol{p}$ of the pen tip is estimated as:

$$\boldsymbol{p} = \boldsymbol{C}' + \boldsymbol{v}_{pen}(C + T), \tag{4.5}$$

where $\boldsymbol{v}_{pen}$ is the direction of the smart-pen, $\boldsymbol{C}' \in \mathbb{R}^3$ is the centroid of the computed 3D points $\bar{\boldsymbol{p}}_i$, $C \in \mathbb{R}^2$ is the centroid of the real led positions and T is the distance of the real centroid from the pen tip. When more than two LEDs are visible, $\boldsymbol{v}_{pen}$ is given by the direction of the least-squares line estimated from the visible LEDs.

4.5.3 SmartPen data processing

The IMU sensor data are first processed on the SmartPen to estimate the orientation of the device. For this purpose a complementary filter (CF) is applied, which enables the reconstruction of the device's orientation from the measured data. In this CF approach, adapted from [114] for the embedded platform, two different estimates of the orientation are fused together to obtain an elevate accuracy. The first one is derived from the angular velocity measured from the gyroscope, which is integrated in time to obtain the orientation of the device at each sampling interval. The second one is derived from the data coming from the accelerometer and magnetometer, forming together an *eCompass*: their combined output gives the measure of two known vectors in the fixed earth reference frame (the gravity acceleration and the earth magnetic field) which can be used to estimate the sensor's orientation. The sensor orientation is represented as the rotation between the pen reference frame and the fixed earth reference frame and can be equally represented by Euler angles, a rotation matrix or a rotation quaternion. In our case the quaternion representation is used for its lower computational complexity and to avoid singularity problems present with the Euler angles. The CF approach uses an opportunely tuned interpolating factor to combine the two estimates, taking advantage of each one's strengths: the gyroscope gives more accurate results for higher dynamic frequencies, while the *eCompass* is accurate in a static situation.

Several alternative filtering techniques exist to estimate the orientation of a device from IMU sensor data, with Kalman Filters, non linear filters and alternative complementary filters as the most used ones. Our choice was based on a comparison of the different algorithms and the evaluation of their computational costs on the embedded platform. The CF approach resulted the one with the best trade-off between accuracy and computational costs, being able to run at higher frequencies on the embedded platform. A higher sampling frequency guarantees a higher accuracy: the SmartPen runs the filter at 120Hz, even if the samples are sent to the computer only at 30Hz, to match the video fps. This ensures a lower integration error and a more accurate response.

4.5.4 Data fusion and filtering

Finally the two estimates from the video and the IMU sensors are fused together using a Kalman Filter (KF) on the host PC. This filter has a standard KF approach: it estimates and updates the filter state, composed by pen's position, velocity and orientation, and uses position and orientation measurements from video and IMU to correct the predicted state and obtain the final estimate.

Following the model in Equations 4.1 - 4.4 we have a state vector composed by position, velocity and orientation $x = \{p, v, \phi\}$. The system prediction model takes account of the cinematic equations, predicting at each step the state vector $\hat{x}_k^-$ and propagating the state covariance matrix P_k^-. We then have two sets of measurements to correct the system: the position estimated from the video

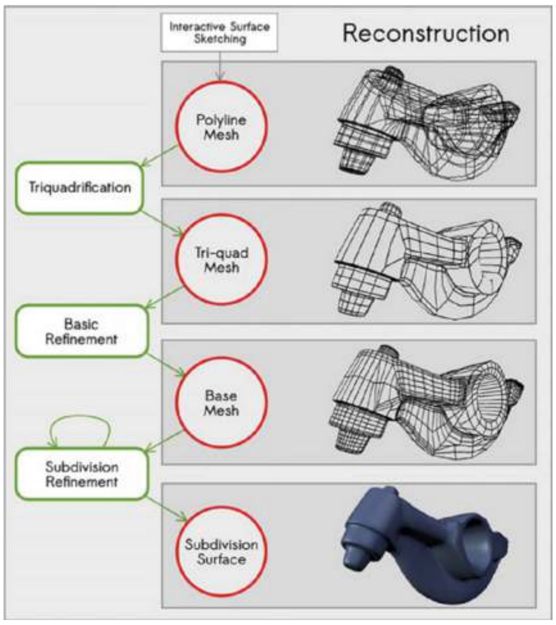


Figure 4.6: The FIRES multistep reconstruction process.

system and the orientation estimated from the SmartPen, resulting in a measurement vector $z_k = \{p_v, \phi_v, \phi_{IMU}\}$. When more than one LED is visible and tracked the full measurement vector is used to correct the estimated state of the system. When the video tracking information is not available due to occlusion problems, a reduced measurement vector consisting in only the ϕ_{IMU} is applied. This guarantees a better final estimation of the device's position and orientation during short periods of complete occlusion of the device, which could not be achieved using only the video estimates.

4.5.5 Surface reconstruction

The FIRES framework combines an interactive 3D curve acquisition system with geometry processing techniques, to provide intuitive design and editing of surface meshes by means of 3D curve network sketching. The system provides fast editing capabilities to support both the acquisition of physical objects and the editing of non-existing parts of them, which can be easily integrated in a CAD working session in real-time.

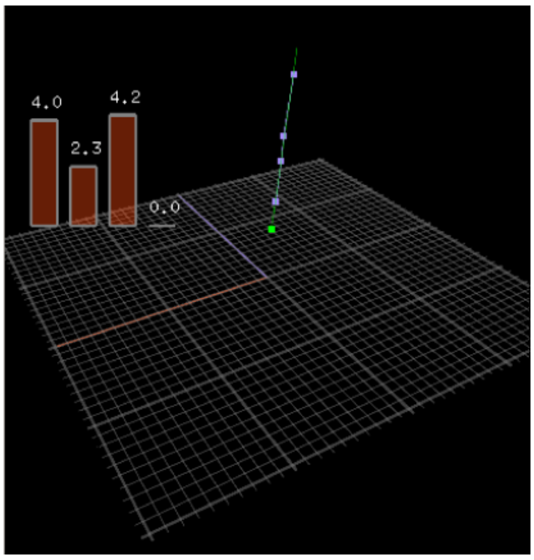
The ISS produces a curve network and, thanks to a set of editing rules, also an associated polyline mesh that is a mesh with faces, vertices and edges augmented with polylines associated to each edge. Irregular curve networks can lead to associated polyline meshes with n-sided, non-planar, and non-convex faces. The polyline mesh is the geometrical representation of the model underlying the curve network created by the user during the process of ISS. However, while the curve network is only a visual representation of the object in terms of spline curves in the 3D space, the polyline mesh contains also topological information. As surfaces of different topology may be compatible with a given curve network the ISS process naturally induces a unique topological structure on the polyline mesh according to the user actions.

The surface construction step is a multi-step process illustrated in Fig. 4.6 for a curve network that can contain sharp and non-sharp curves. Our current implementation allows the user to associate a sharpness value to each sketched curve by simply switching a button on the SmartPen. The system first generates a base mesh from the polyline mesh, then a surface mesh constructor transforms the base mesh into a refined mesh taking into account also the user sharpness constraints, which can eventually be represented as subdivision surface mesh. Any of the different model representation forms (curve network, coarse or surface mesh) can be integrated in a CAD system for further processing.

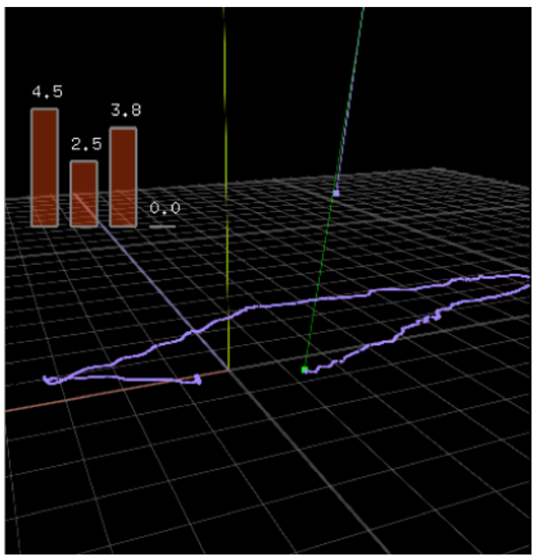
The basic refinement step tasselates each n-sided face in four-sided polygons by following the same splitting rules as of a generalized Catmull-Clark subdivision [48]. For each face a new vertex is placed at the face centroid and is connected with the midpoint of each edge. The basic refinement can be repeated iteratively as necessary.

After the basic refinement step, we obtained a mesh which is a good piecewise linear reconstruction of the shape defined by the curve network. The base mesh consists of four-sided faces with eventually extraordinary vertices, i.e. with more than four incident edges. Our next objective is to produce a smooth surface exploiting a subdivision surface scheme which naturally provides a unique representation of the arbitrary topology reconstructed object. For our reconstruction purposes we experimented with both interpolatory and approximating subdivision schemes.

In particular, we used the well-known approximating Catmull-Clark subdi-



(a)



(b)

Figure 4.7: Screenshots of the CAD application developed. (a) Tracking of the LEDs (in purple) and reconstruction of the pen's pose (green). (b) Tracking of the pen while tracing a curve.

vision scheme and an adapted version of the NULISS interpolatory subdivision scheme [32]. While the former takes into account only the vertices of the base mesh, the latter also integrates its associated polylines drawn by the user. This provides two different levels of control on the accuracy of the design process. Control curves used as handles for deformation right after their definition are approximated by the subdivision surface, while characteristic curves drawn to define specific features of the object to be reconstructed are interpolated. Moreover, the subdivision rules have been enriched by special rules for reconstructing sharp edges or corners.

4.6 Results

We used the acquisition system, as presented in sections 4.4 and 4.5 to acquire some SmartPen traces and then to test the capabilities of the system to reconstruct and edit virtual models of objects. Two sample screenshots of the appli-

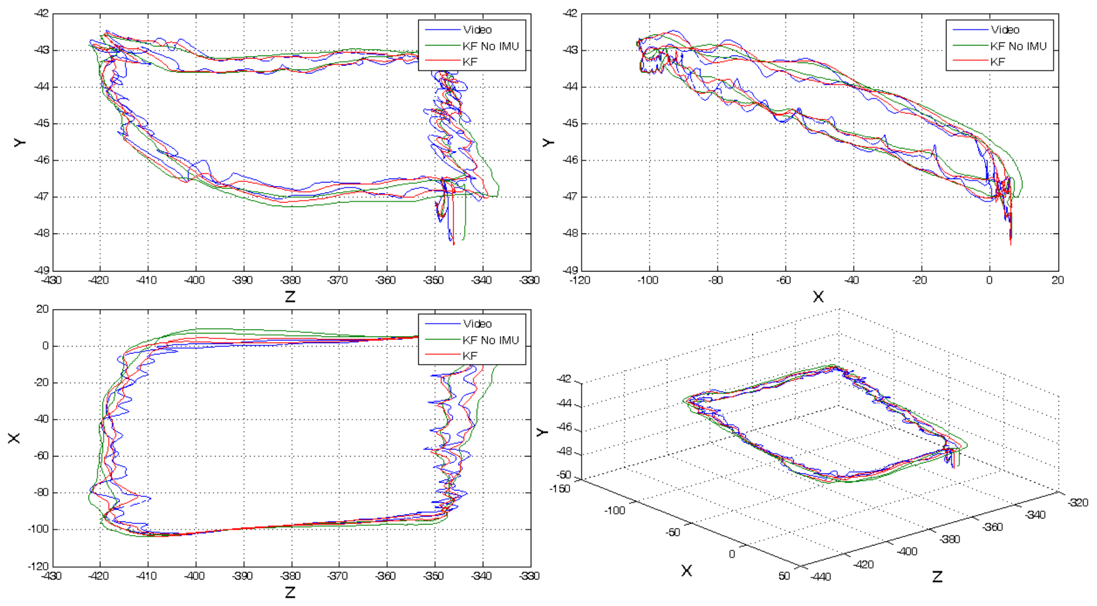
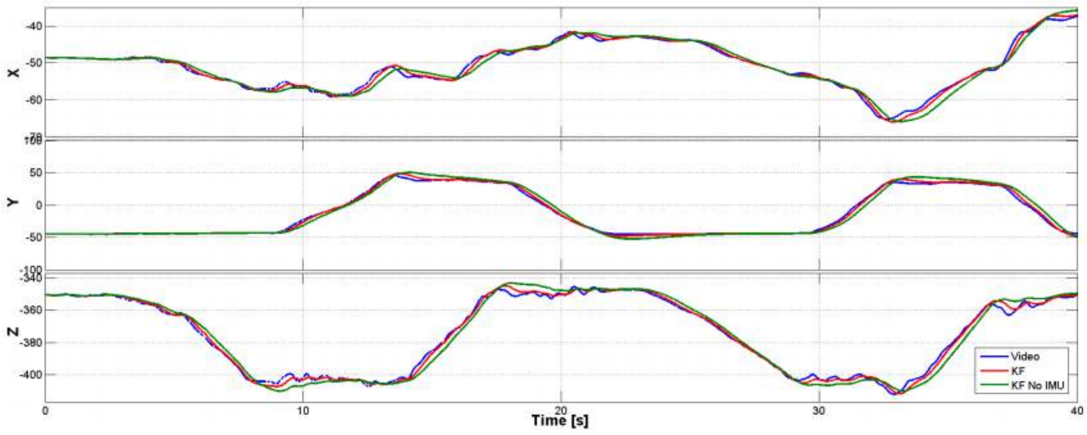


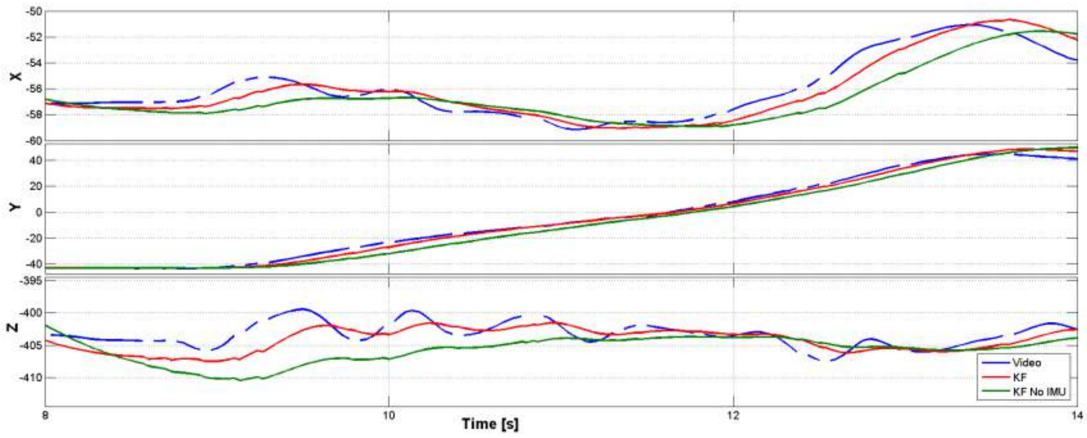
Figure 4.8: Example of a trace collected with the SmartPen

cation are shown in Figure 4.7, where we can see the result of the tracking of the device and its use to trace a curve in the CAD application.

In Figures 4.8 and 4.9 we can see three dimensional plots of the tracking of the device, when drawing a rectangle on the horizontal plane in front of the camera (plane XZ in the global reference frame). Figure 4.9(b) shows a zoom of the same trajectory. In the figures we have a comparison of three tracking methods: the raw video tracking, the KF filtered video tracking and the KF filtered video and IMU tracking. The first one is the raw output of the *Pen Estimation* block, as described in Figure 4.4, while the other two are the outputs of the Kalman filter, one with just the video data and the other using both video and IMU data. From the figures we can see how the tracking from the raw video streams suffers from problems and in particular has some points when the leds are not seen where tracking information is lost. The KF approach can overcome this problems, predicting the pen position in presence of small amounts of missing video data. The advantage of the filter with the IMU is that is has a smoother behavior when compared to the raw one and is more accurate when compared to the video KF one. The reason of the two filters having a similar result is in the



(a)



(b)

Figure 4.9: Example of a trace collected with the SmartPen.

nature of the movements performed. A precise interaction with a drawing tool involves slow and accurate motions which apply small or null accelerations on the device. In this case both KF approaches can model well the accelerations with a null mean gaussian noise, while the IMU one takes advantage of the accurate IMU estimations.

Next we will show some examples of the complete proposed reverse engineering pipeline. In particular we used the system to reconstruct a 3D virtual model of an old-style telephone pictured in 4.10(a). The SmartPen was used to acquire the style-curves of the object pictured in 4.10(b), while in 4.10(c) we can see the curves labeled as sharp during the acquisition (in red). The final model of the

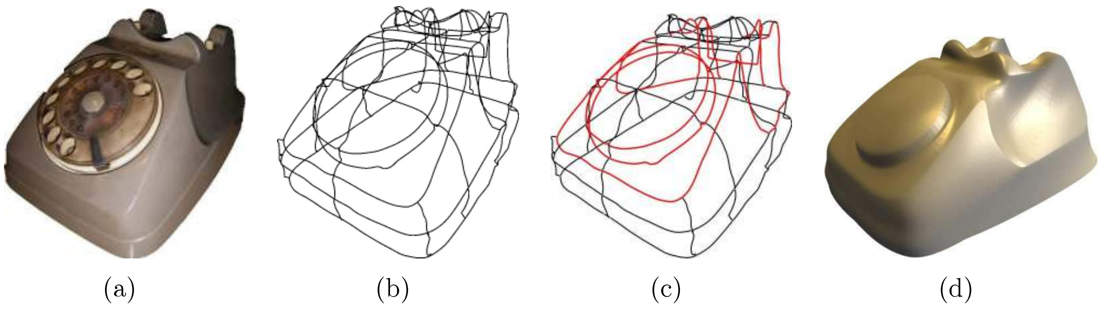


Figure 4.10: Reverse engineering of a telephone.

object, after the hybrid refinement process combining one step of the adapted NULLIS algorithm and 2 steps of Catmull-Clark subdivision, is shown in 4.10(d).

In the second example FIRES is applied for the reconstruction of the vinegar bottle shown in Figure 4.11(a). The skinning tool of the ISS is used to produce the curve network of Figure 4.11(b) which results into the corresponding polyline mesh of Figure 4.11(c). Finally the subdivision result obtained by 3 Catmull-Clark subdivision steps is given in Figure 4.11(d).

In order to show the capability of FIRES to add virtual parts to an existing object, in Figure 4.12 we designed a handle to add to the 3D model of Figure 4.11(d). At this aim, we first used the hole creation tool (curve network in Figure 4.12(a) and the associated polyline mesh in Figure 4.12(b)) and then we applied the border gluing to produce the handle (curve network in Figure 4.12(c) and the associated polyline mesh in Figure 4.12(d)). The final result after 3 steps of Catmull-Clark subdivision is shown in Figure 4.12(e).

4.7 Conclusions

This work has introduced a wireless low-cost pen-like device for a fast interactive reverse engineering framework named FIRES. The system enables real-time acquisition and manipulation of complex geometrical shapes through the SmartPen, a wireless and interactive input device.

The FIRES framework utilizes the SmartPen as the primary interaction tool to support the reconstruction and editing of virtual 3D models of objects in a

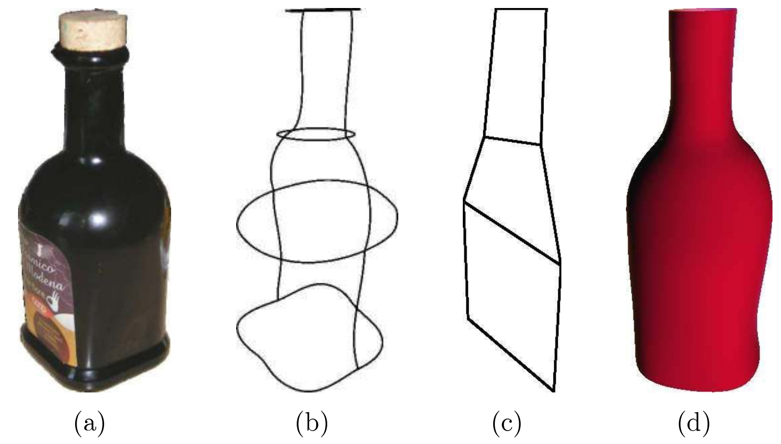


Figure 4.11: Reverse engineering of a vinegar bottle.

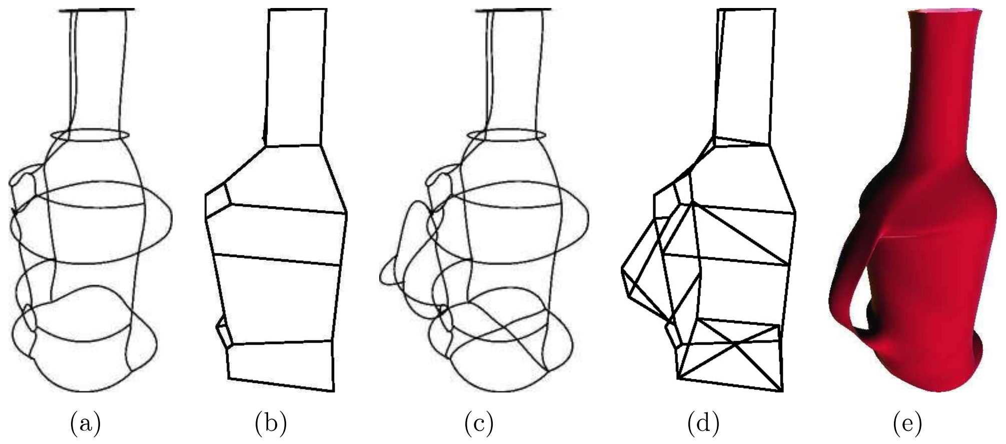


Figure 4.12: Adding a virtual handle to the geometric model illustrated in Figure 4.11.

CAD application. The SmartPen embeds inertial sensors and their data is fused with the output of a stereo computer vision system in order to accurately track the position and orientation of the device as it is moved in space by the user. The device can be used to draw the style-lines of the object to be modeled, which are then used to reconstruct its surface.

Chapter 5

Wearable Sensor Network for Motion Analysis

In the previous chapter we presented an interactive device equipped with an Inertial Measurement Unit (IMU) which was used as a 3D user interface. Starting from the results of that experience, we improved the hardware design of the sensing device to develop a network of miniaturized wearable IMUs for the analysis of human motion.

Human body movement mechanics is a topic of interest for science since ancient years and today many different disciplines use motion analysis systems to capture movement and posture of the human body. Thanks to the advances in sensing and processing capabilities, the use of wearable sensors is one of the most promising motion capture technologies. Its advantages include unobtrusiveness, simplicity of use, no need for the installation of equipment in the environment and extended spatial and temporal coverage. Active research is being carried out in both hardware integration and processing techniques to extract the desired features from the captured sensor data.

The work presented in this Chapter introduces a new hardware platform for inertial motion sensing, which integrates state of the art sensing and processing components into a small, light and accurate wearable sensor node. The device is equipped with a wireless Bluetooth transceiver to enable the communication with a broad range of devices (PCs and smartphones) and to allow the formation of a body area sensor network where several nodes can monitor articulated movements. Moreover, we studied advanced sensor fusion algorithms to process the sampled data and estimate useful features such as the orientation of each device. The algorithms were optimized and implemented on board of the devices, to allow distributed processing of sensor data in the network and improve the performance and energy efficiency of the system.

5.1 Overview

The study of human motion is an interesting and open field of research, with a broad range of applications. An accurate tracking of parts of human body is an enabling factor for several applications, such can be human-computer interaction [175], visualization and navigation in virtual environments [173], computer animation [145] and a wide range of health applications [134]. Depending on the application, several tracking technologies can achieve the desired levels of accuracy for the racking of needed motions [172]. The main approaches can be divided in video-based and sensor-based. Computer vision tracking systems have usually a high accuracy, but suffer from limitations such as the need to install the cameras in the environment, the limited range of operations and a high cost. Recent advances in the manufacturing technology of Micro Electro-Mechanical Systems (MEMS) have allowed to considerably decrease the prize and size of inertial sensors, making them a preferred choice for motion tracking applications. Sensor tracking solutions, based on wearable inertial sensing platforms, have the advantage of being self-contained, unobtrusive and without strict range limitations. However, these devices present some undesired characteristics that require meticulous calibration and signal preprocessing procedures before using the gathered data to compute position and extract information about movement with enough degree of accuracy and reliability [134].

In this work we present a miniaturized inertial measurement unit (IMU) for the analysis of human motion. The *EXLs1* device, developed in collaboration with EXEL, is a compact wearable sensor node equipped with inertial (accelerometer and gyroscope) and magnetic sensors, along with a memory module and a Bluetooth wireless transceiver. We employed state of the art digital MEMS sensors, achieving high accuracy with limited dimensions and power consumption. The system embeds an ARM Coretex M-3 microcontroller, enabling advanced onboard data processing with an optimized performance-power consumption tradeoff, tailored for battery-powered systems. Energy efficiency for the platform is achieved through an optimal use of the hardware resources.

The sampled sensor data is locally processed on the device, in order to reconstruct its relative orientation in space. This information is used for the analysis of the position and motion of body segments to which the device is attached. We compared and optimized several state of the art algorithms for the estimation of orientation from IMU sensor data, including the Kalman Filter (KF), its Extended (EKF) variation and different versions of Complementary Filters (CF). The different algorithms were optimized to run in real-time on the embedded microcontroller and we compared their performance in terms of accuracy and computational cost.

The use of a Bluetooth transceiver allows an immediate connection of the device with a vast range of systems supporting this standard, including personal computers and smatphones. This protocol was chosen for its wide adoption and the simplicity of its use, along with its bandwidth availability that allows the streaming of data from multiple devices simultaneously. With this choice we could set up a Body Sensor Network (BSN) with up to 7 nodes. The network of sensor nodes allows us to stream the desired information from multiple nodes to one collection center, increasing the range of movements we can analyze. The nodes can also store locally the data for a later offline elaboration. In this case we use the Bluetooth network to synchronize the different nodes but we don't stream the data, saving energy and prolonging the battery life.

5.2 Related Work

In the recent years there has been increased interest in body sensor networks for wearable applications, such as elder care [61], emergency response [110], studying athletic performance [19], gait analysis [185], and activity classification [126]. A great deal of work has focused on sensor and hardware design [79, 89], MAC and routing protocols [39] and algorithms for processing wearable sensor data [78, 159]. A comprehensive survey on sensor-based wearable systems for monitoring of human movement can be found in [150].

One of the most used approaches to process sensor data from wearable IMUs is to estimate the device's orientation. When sensor nodes are attached to body limbs (e.g. upper or lower arms), using the computed orientation and a kinematic model of the human body we can reconstruct and track the pose of the user [31]. IMUs were first developed and used in navigation applications as attitude and heading reference systems (AHRS), where several algorithms have been developed to track the orientation of a vehicle [69]. A standard approach is to use a Kalman filter to optimally fuse data from the available sensors and estimate the corresponding orientation [91]. Since its first adoption, several modifications of the KF have been proposed to enhance its performance such as the EKF [75] or the UKF [162]. Recently complementary filtering (CF) techniques based on non-linear observers gained popularity [114, 116]. They present a reduced complexity and need a simpler tuning phase, while maintaining the accuracy when compared to the KF approach.

With the availability of miniaturized inertial sensors and the appearance of wearable IMUs, the research community started to apply orientation estimation algorithms to the analysis of human motion. One example is the work presented in [112], where the proposed algorithm continuously corrects the orientation estimates obtained by mathematical integration of the 3D angular velocity measured using a gyroscope. The correction is performed using an inclination estimate obtained using a 3D accelerometer. This reduces the integration drift that originates from errors in the angular velocity signal. The method is realized using a KF that takes into account the spectra of the signals involved as well as a fluctuating gyroscope offset. A different approach is taken in [149], where an EKF is used for the fusion of inertial and magnetic sensing to estimate relative positions and orientations of a wearable device. The filter predicts the position and orientation based on the signals measured by the accelerometer and gyroscope. The system decides to perform a magnetic update only if the estimated uncertainty associated with the relative position and orientation exceeds a predefined threshold. In this case a magnetic coil is actuated to generate a known magnetic field, which is measured by a magnetometer and used to correct the inertial estimate.

A more application-specific solution, tailored for healthcare is presented in [111]. Here the authors present *Mercury*, a wearable, wireless sensor platform for motion analysis of patients being treated for neuro-motor disorders, such as Parkinson's Disease, epilepsy, and stroke. Mercury is designed to support long-term, longitudinal data collection on patients in hospital and home settings. Patients wear up to 8 wireless nodes equipped with sensors for monitoring movement and physiological conditions. Individual nodes compute high-level features from the raw signals, and a base station performs data collection and tunes sensor node parameters based on energy availability, radio link quality, and application specific policies. The work in [134] presents the development of *Wagyromag*, a wireless sensor network for monitoring and processing human body movements in healthcare applications. Here the hardware platform is introduced and different orientation estimation algorithms are evaluated, with the focus on healthcare applications.

While several works address the theoretical approach and evaluate the performance of orientation estimation algorithms, limited research has been carried out on their implementation for embedded systems. A real-time posture tracking system has been developed in [179] using a network of compact wireless sensor devices worn by the user. Each device is a complete inertial/magnetic tracking unit which performs in situ orientation estimation based on its own sensor readings, using a complementary quaternion-based filter. The work in [77] presents the design and evaluation of a novel miniature AHRS unit, named *ETHOS*, specifically designed for wearable use. It uses off-the-shelf sensor components integrated into a system offering local processing resources to compute orientation online. The system offers both, local data storage and ultra-low power wireless transmission options, and scalable processing capacity which can be adapted to the application's demands, e.g. regarding orientation streaming bandwidth. The work in [178] directly compares the most popular orientation estimation approaches and evaluates their computational cost by simulating the different algorithms. In [123] a CF-based nonlinear observer is optimized and implemented on an embedded

8-bit microcontroller for a miniature unmanned aereal vehicle. In both works the CF approach results several times less computationally demanding when compared to the EKF solutions, without a loss in accuracy.

Some commercial human motion capture systems are also available: Xsens offers the *MVN*, a full 6DoF Human motion tracking system using up to 17 wired IMUs [145]. Another interesting solution is the *iNemo* platform from STMicroelectronics [37] which embeds a 32-bit RISC microcontroller (MCU) and 3D miniaturized MEMS sensors, with the capability of on-board sensor fusion for orientation estimation. In our work we optimized and implemented several state of the art algorithms for orientation estimation, including KF, EKF, CF approaches. We evaluated their performance on the microcontroller embedded in the EXLs1 sensor node by means of accuracy, computational times and power consumptions.

5.3 Orientation estimation using inertial and magnetic sensors

A *rigid body* is an idealization of a body with volume and mass which has a shape that can not be changed. To describe the orientation of a rigid body in three dimensional space, it is conventional to choose a global coordinate system (G) attached to an appropriate inertial frame and a body fixed coordinate system (B) which is attached to the rigid body. The global reference frame is commonly defined as the local earth-fixed reference frame with the coordinate axes x, y and z pointing in the local north, west and up directions respectively (see Fig. 5.1).

The position of a point in space can be described using a three dimensional point vector. If a rigid body is described in terms of point vectors, it can be rotated or oriented by rotating each vector individually. This may be completed by multiplying each vector of the body by an appropriate rotation matrix, as expressed by $\boldsymbol{v}^G = R_B^G \boldsymbol{v}^B$, where $\boldsymbol{v}^G$ and $\boldsymbol{v}^B$ are the representations of the desired vector in the global and body frames respectively and R_B^G is the rotation matrix to perform the rotation from the body to the global frame. The notation employed to reference the rotations uses a capital subscript to indicate the frame being rotated and a capital superscript to refer the destination frame in which the

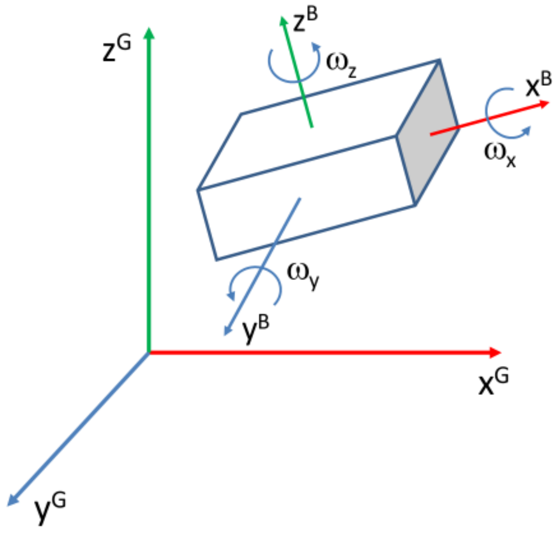


Figure 5.1: Global (G) and body (B) reference frames.

result is expressed. The inverse rotation, from the global frame to the body frame is obtained transposing this rotation matrix: $R_G^B = (R_B^G)^T$.

A rotation matrix can represent an arbitrary rotation between the two reference frames, but it can always be divided into three separate rotations, each around one of the three axes x, y, and z. For instance, a rotation of an angle ϕ around the x axes is described by:

$$\boldsymbol{v}^G = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & -sin\phi \\ 0 & sin\phi & cos\phi \end{bmatrix} \mathbf{v}^G = R_x(\phi)\boldsymbol{v}^B \tag{5.1}$$

Likewise, rotations of θ around y and γ around z can be expressed as:

$$\boldsymbol{v}^G = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix} \mathbf{v}^G = R_y(\theta)\boldsymbol{v}^B \tag{5.2}$$

$$\boldsymbol{v}^G = \begin{bmatrix} cos\gamma & -sin\gamma & 0 \\ sin\gamma & cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{v}^G = R_z(\gamma)\boldsymbol{v}^B \tag{5.3}$$

and the total rotation can be computed as: $R_B^G = R_z(\phi)R_y(\theta)R_z(\gamma)$. The angles $\{\phi, \theta, \gamma\}$ compose the Euler angles representation of the performed rotation and are often referred to as $\{roll, pitch, yaw\}$.

The use of Euler angles to represent rotations leads to the so called *gimbal lock* problem. It arises when two of the elementary rotations end up on a collinear axes (i.e. when the x unit vector points straight up the $roll$ and yaw gimbal axes are collinear) and leads to the loss of one degree of freedom in the orientation representation. To avoid the *gimbal lock* the equivalent quaternion representation can be used, which is not affected by this problem.

Quaternions are an extension of complex numbers defining a four-dimensional volume using one real and three imaginary parts. A quaternion $\boldsymbol{q}$ is defined as $\boldsymbol{q} = [w + xi + yj + zk] = (w, \mathbf{v})$, where w is its scalar part and $\mathbf{v} = [x, y, z]$ is the vector part. An arbitrary rotation of a rigid body can always be described as a rotation (ϕ) around a single inclined axes ($\boldsymbol{v} = [v_x,\ v_y,\ v_z]$). This rotation can be represented by the quaternion:

$$\mathbf{q}_\phi = [cos\frac{\phi}{2}, v_x sin\frac{\phi}{2}, v_y sin\frac{\phi}{2}, v_z sin\frac{\phi}{2}] \tag{5.4}$$

As with the rotation matrix, a three dimensional vector $\boldsymbol{v}$ can be rotated by a quaternion $\boldsymbol{q}$ applying the following relationship:

$$\mathbf{v}^G = \boldsymbol{q}_B^G \otimes \boldsymbol{v}^B \otimes (\boldsymbol{q}_B^G)^* \tag{5.5}$$

where $\mathbf{q}^*$ is the conjugate of the quaternion $\boldsymbol{q}$ defined as $\boldsymbol{q}^* = [w - xi - yj - zk]$ and $\otimes$ denotes the quaternion product, defined for two quaternions $\boldsymbol{a}$ and $\boldsymbol{b}$ as:

$$\begin{aligned}\mathbf{a} \otimes \mathbf{b} &= [a_1\, a_2\, a_3\, a_4] \otimes [b_1\, b_2\, b_3\, b_4] \\ &= \begin{bmatrix} a_1b_1 - a_2b_2 - a_3b_3 - a_4b_4 \\ a_1b_2 + a_2b_1 + a_3b_4 + a_4b_3 \\ a_1b_3 - a_2b_4 + a_3b_1 + a_4b_2 \\ a_1b_4 + a_2b_3 - a_3b_2 + a_4b_1 \end{bmatrix}^T \end{aligned} \tag{5.6}$$

To correctly use the quaternion product, the vector $\boldsymbol{v}$ contains a 0 inserted as the first element to make it a 4 element row vectors. A detailed background on orientation representations and quaternion arithmetics can be found in [69]. An overview of orientation estimation techniques from inertial and magnetic sensors will be presented in the reminder of this section, after the introduction of appropriate models for the used sensors.

5.3.1 Sensor models and calibration

Sensor nodes used for motion capture and orientation estimation are usually equipped with inertial (accelerometer and gyroscope) and magnetic (magnetometer) sensors. The raw measurements Y_g from the gyroscope, Y_a from the accelerometer and Y_m from the magnetometer, expressed in body frame, are modeled as:

$$\begin{aligned} Y_g &= w_i^B + b_g + \nu_g \\ Y_a &= a_i^B - g^B + b_a + \nu_a \\ Y_m &= m^B + \nu_m \end{aligned} \tag{5.7}$$

where ν_i, $i \in \{g, a, m\}$, are uncorrelated Gaussian white noise with variance denoted as σ_i^2, while b_g and b_a are time-varying biases modeled as a first-order Gauss-Markov process

$$\dot{b} = \beta b + \nu_b \tag{5.8}$$

characterized by time constant $1/\beta$ and uncorrelated Gaussian white noise ν_b with variance σ_b^2. The gyroscope reports the angular rates of the body frame with respect to an inertial frame (w_i^B) expressed in the body frame, for each of the three axes. The positive orientation of the rotations is defined by the right hand rule (see Fig. 5.1). The accelerometer senses the accelerations to which the device is subject, composed by the inertial acceleration applied to the device (a_i^B) and the fixed earth's gravity acceleration (g^B), both expressed in the body reference frame. The magnetometer measures the intensity of the magnetic field surrounding the device. Without the disturbances of external sources, it reports the local earth magnetic field, pointing to the magnetic north.

The MEMS sensors employed are factory calibrated, but for a correct use they need to be calibrated when integrated in a device, with respect to the body frame.

The gyroscope is affected by a constant bias b_{0g}, which can be easily estimated. When still, the sensor should report a null angular velocity along all three axes, but instead it has a constant bias. To compensate this unwanted effect, we put the sensor still for some seconds and take the mean value of the readings along each axis as the offset to be removed from future readings. The compensated gyro readings $\tilde{Y}_g$ can be expressed as:

$$\tilde{Y}_g = Y_g + b_{0g}. \tag{5.9}$$

The accelerometer measures are affected by scale and offset factors for each axis and misalignment factors between the sensor's axes and the body frame axes. The relationship between the calibrated $\tilde{Y}_a$ and the raw measurements Y_a can be expressed as:

$$\tilde{Y}_a = C_a Y_a + b_{0a} \tag{5.10}$$

where $C_a \in \mathbb{R}^{3\times3}$ summarizes misalignment and scale factors while $b_{0a} \in \mathbb{R}^3$ is an offset vector. The goal of the accelerometer calibration is to determine C_a and b_{0a}, so that the calibrated values can be obtained with any new raw measurement. To estimate the parameters the sensor is placed at 6 stationary positions with known desired output, i.e. with the gravity vector aligned with the positive and negative directions of the three axes. At each position we compute the average of 10 seconds of accelerometer raw data and then we apply the least square method to obtain the optimal calibration parameters.

The magnetometer measures the Earth's magnetic field, which is weak and can be corrupted by intrinsic sensor imperfections and external magnetic sources. Intrinsic imperfections cause again scale, offset and misalignment factors between the sensor's axes. External magnetic distortions can be divided into Hard-iron and Soft-iron distortion [70]. Hard-iron distortion is produced by materials that exhibit a constant magnetic field, thereby generating a constant additive bias to the output of each magnetometer axes. Soft-iron distortion is the result of a material that influences, or distorts, a magnetic field, in a way that cannot only be captured as the effect of an additive disturbance on the magnetic measures. Only the effects of materials that have a constant position and orientation with respect to the sensor can be compensated by the calibration procedure.

An extensive magnetometer calibration method is proposed in [70] and it is here applied to correct raw sensor readings. In particular the magnetometer measurement model adopted for the calibration is given by:

$$\begin{aligned} Y_m &= C_m \tilde{Y}_m + b_{0m} \\ &= \begin{bmatrix} \epsilon_1 & 0 & 0 \\ \epsilon_2 \sin \rho_1 & \epsilon_2 \cos \rho_1 & 0 \\ \epsilon_3 \sin \rho_2 \cos \rho_3 & \epsilon_3 \sin \rho_3 & \epsilon_3 \cos \rho_2 \cos \rho_3 \end{bmatrix} \tilde{Y}_m + b_{0m} \end{aligned} \tag{5.11}$$

where $\tilde{Y}_m$ is the desired corrected value, coefficients ϵ_k represent the sensor scaling errors and ρ_k the sensor misalignment angles. Equation (5.11) is solved for $\tilde{Y}_m$ and substituted into

$$\left\| \tilde{Y}_m \right\|^2 = \tilde{Y}_{mx}^2 + \tilde{Y}_{my}^2 + \tilde{Y}_{mz}^2 \tag{5.12}$$

where $\left\| \tilde{Y}_m \right\|$ is the (known) local magnetic field amplitude. This gives a quadratic polynomial in Y_{mx}, Y_{my}, Y_{mz}, whose coefficients are nonlinear functions of ϵ_k, ρ_k

and b_{0m}. The numerical values of these coefficients are found from a least-squares solution using Y_m, namely the raw data. Thereafter, the system of nonlinear equations is solved numerically, and the magnetometer measurements are corrected using:

$$\tilde{Y}_m = C_m^{-1}(Y_m - b_{0m}). \tag{5.13}$$

Magnetic measurement errors are more critical when compared to the other two sensors and primarily affect the yaw angle estimation [69]. The eventual ferromagnetic components present on the device influence the magnetic field sensed, but can be compensated because they have a fixed position in the body frame (hard iron disturbance). On the other hand, magnetic or metal objects encountered in the environment introduce a soft iron disturbance that can't be compensated. Hence the magnetic calibration procedure should be available as an on-line routine to re-calibrate the sensor when needed. Same results are shown by [28], where the impact of magnetometer calibration is validated in a helicopter navigation system.

To summarize, the calibration procedure is the following:

1. collection of static data with the device resting on the 6 different orientations;
2. computation of accelerometer and gyroscope offsets and scale factors;
3. three-dimensional rotations of the device, approximately around the three axis;
4. computation of the magnetometer parameters;
5. storage of all computed parameters and compensation of future samples.

5.3.2 Orientation computation from sensor data

Inertial and magnetic sensors can be used to estimate the orientation of the device, that is to estimate the rotation between the body frame attached to the device and the global inertial frame. In this section we will examine the techniques that allow

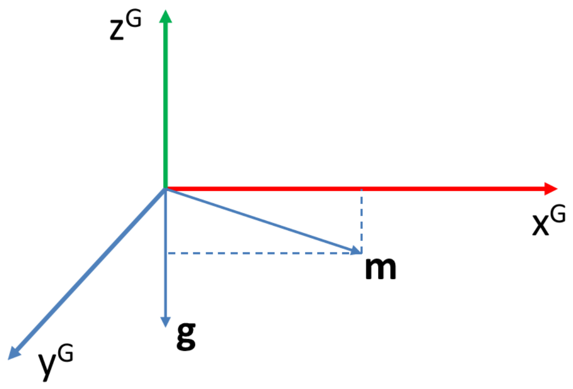


Figure 5.2: Earth's gravity acceleration ($\boldsymbol{g}$) and magnetic field ($\boldsymbol{m}$) in the global reference frame.

us to directly estimate the orientation from the available sensor readings. In this category we have the algorithms that employ accelerometer and magnetometer readings or alternatively the ones that integrate the gyroscope output to obtain the deisred information.

Accelerometer and magnetometer

In static conditions and without the influence of external magnetic sources, the accelerometer measures the earth's gravity acceleration $\boldsymbol{g}^B$, while the magnetometer measures the earth's magnetic field $\boldsymbol{m}^B$. Both vectors are expressed in the body frame and their representation in the global reference frame is known, which enables the reconstruction of the relative orientation between the two frames.

The global reference frame is an inertial earth-fixed frame opportunely chosen to have the z axis aligned with the gravity vector and the x axis aligned with the projection of the local magnetic field on the horizontal plane, as shown in Fig. 5.2. The local magnetic field points towards the magnetic north and usually has a declination angle with respect to the earth's tangent plane, pointing towards the ground. The declination angle varies with the geographical location of the measurement, but it is known and can be retrieved from [9].

From the measurement of the gravity vector $\boldsymbol{g}^B = [g_x, g_y, g_z]$, roll and pitch angles (ϕ, θ) defining the inclination of the body frame relative to the global frame, can be estimated by applying basic trigonometric equations as:

$$\begin{aligned}\phi =& atan2(g_y, g_z)\\ \theta =& atan(-g_x, \sqrt{g_y^2 + g_z^2})\end{aligned} \tag{5.14}$$

Knowing the two angles ϕ and θ, the magnetic measurement $\boldsymbol{m}^B = [m_x, m_y, m_z]$ is projected from the arbitrary rotated body frame to the horizontal plane to estimate the remaining yaw angle (γ). Indicating with m_{xh} and m_{yh} the magnetic components in the horizontal plane, we have:

$$\begin{aligned}m_{xh} =& m_x \cos\phi + m_y \sin\theta \sin\phi - m_z \cos\theta \sin\phi\\ m_{yh} =& m_y \cos\theta + m_z \sin\theta\end{aligned} \tag{5.15}$$

and finally

$$\gamma = atan2(-m_{yh}, m_{xh}) \tag{5.16}$$

From the computed Euler angles it is easy to obtain the rotation matrix or the equivalent quaternion with conversion formulas [69]. This method has the advantage of having a limited computational complexity, but it suffers from both the electronic random noise which affects the sensors and from external disturbances. In particular, if the device is subject to high dynamic accelerations, they will become the principal component of the accelerometer's output, which won't estimate the gravity vector needed to compute the orientation. In addition, the proximity of ferromagnetic materials or of a source of magnetic field, will disturb the magnetic sensor invalidating its measurement of the earth's magnetic field.

Gyroscope

To compute the orientation of the device using the gyroscope, we need to integrate the angular rates $w_i^B = [w_x, w_y, w_z]$ obtained from the sensor. This can be achieved integrating the derivative of the chosen representation method. For example, the quaternion derivative describing the rate of change of the body frame relative to the global frame ($\dot{\boldsymbol{q}}_B^G$) can be calculated as:

$$\dot{\boldsymbol{q}}_B^G = \frac{1}{2}\boldsymbol{q}_B^G \otimes \boldsymbol{w}_i^B \tag{5.17}$$

where a leading zero has been added to $\boldsymbol{w}_i^B$ to make it a four component vector. The estimated orientation at time t, $\hat{\boldsymbol{q}}_B^G(t)$, can be computed by numerically integrating the quaternion derivative, provided the initial conditions and the sampling period Δt, as expressed in Eq. 5.18.

$$\hat{\boldsymbol{q}}_B^G(t) = \hat{\boldsymbol{q}}_B^G(t-1) + \frac{1}{2}\Delta t[\hat{\boldsymbol{q}}_B^G(t-1) \otimes \boldsymbol{w}^B(t)] \tag{5.18}$$

This method has some advantages over the one based on accelerometer and magnetometer: (1) it is computed only using matrix and vector multiplication operations, without the need for trigonometric functions, (2) it produces a smoother signal, since the integrations steps filters out the electronic random noise with zero mean, (3) the signal is not affected by the dynamic accelerations, so, even under intense movements, a more accurate orientation can be estimated. On the other hand, this approach suffers from a continuous drift of the estimation, caused by the numerical integration and by the varying gyroscope bias. Recent MEMS sensors have limited values of bias, reducing the estimation drift, but still this method can't be used alone for prolonged intervals.

5.3.3 Sensor fusion techniques for orientation estimation

Sensor fusion strategies aim to use the information captured by different sensors to offer an accurate estimation of the desired quantities. This approach is based on the assumption that some sensors provide useful information where other sensors are not working properly, so the fused contribution will result in a better performance when compared to the single ones. In the orientation estimation case examined here, this results in a processing technique that filters the noise and dynamic disturbances from the accelerometer signal and removes the integration drift present in the gyroscope estimation. Notable approaches to perform this operation are the Kalman filter or the complementary filter.

Kalman filter

The KF is the simplest example of a predictive filter [91]. It represents uncertainties as Gaussian random variables, fully described by a mean and a covariance matrix, and models the system with linear dynamics and observations. Since Gaussians are preserved under linear transformation, the Kalman filter's implementation uses only linear algebra operations. The KF processes all the available measurements, regardless of their precision, to estimate the current value of the variables of interest. Furthermore it uses the knowledge of the system and measurement device dynamics, the statistical description of the system noises, measurement errors, and uncertainty in the dynamics models, and any available information about initial conditions of the variables of interest.

The KF addresses the general problem of trying to estimate the state of a discrete-time controlled process that is governed by the following set of linear equations:

$$\begin{aligned} \boldsymbol{x}_k &= A\boldsymbol{x}_{k-1} + \boldsymbol{w}_{k-1} \\ \boldsymbol{z}_k &= H\boldsymbol{x}_k + \boldsymbol{v}_k \end{aligned} \tag{5.19}$$

where $\boldsymbol{x}_k$ is the state of the process and $\boldsymbol{z}_k$ is the noisy measurement observed to estimate the state of the process, $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ represent, respectively, the process noise and the measurement noise and are assumed to have a zero-mean Gaussian distribution. More detail on the KF can be found in Section 4.3.2 or in [69, 91]

In the case of interest, the state of the process represents the orientation to be estimated: $\boldsymbol{x} = \boldsymbol{q}_B^G$. The measurement used to update the process is composed of two independent estimates of the orientation, the one obtained from the accelerometer and the magnetometer (denoted with the subscript am), and the one obtained integrating the gyroscope (denoted with the subscript w): $\boldsymbol{z} = [(\boldsymbol{q}_B^G)_{a,m}, (\boldsymbol{q}_B^G)_w]^T$. Hence the measurement noise will also be composed of two parts, the one relative to the accelerometer and magnetometer estimate and one relative to the gyroscope estimate. By opportunely tuning the two components, the KF framework fuses the two separate estimates to produce an accurate final result.

This approach can be refined by using the sensor models introduced in Section 5.3.1 and introducing the computations of the two orientation estimates directly in the KF. In this case, the state dynamics is represented directly by the derivative of the orientation introduced in Eq. 5.18 and the measurements are the accelerometer and magnetometer outputs used to estimate the gravity and earth's magnetic fields respectively. Based on the sensor models and the attitude dynamics, the complete system con be modeled as follows:

$$\begin{aligned} \dot{q}_B^G &= \frac{1}{2} q_B^G \otimes w_i^B \\ \dot{b_g} &= \beta_g b_g + \nu_{bg} \\ \dot{b_a} &= \beta_a b_a + \nu_{ba} \end{aligned} \tag{5.20}$$

$$\begin{aligned} \tilde{Y}_a &= R_G^B g^G + b_a + \nu_a \\ \tilde{Y}_m &= R_G^B m^G + \nu_m \end{aligned} \tag{5.21}$$

The system model introduced in Eq. 5.20 - 5.21 is nonlinear, hence the KF approach can not be applied directly. Instead the Extended KF (EKF) approach is applied. The EKF approximates the non-linear equations 5.20 and 5.21 with the Taylor series expanded around the estimated state vector $\hat{x}$ and then applies the KF update-correction iteration to estimate the system state. To ensure a better convergence of the linearized EKF, a complementary version of the filter can be employed, which uses the errors $\delta x = x - \hat{x}$ and $\delta y = \tilde{y} - \hat{y}$ as system state and measurements respectively. The system in the new form can be rewritten as:

$$\begin{aligned} \delta\dot{\rho} &= -R_G^B(\delta b_g - \nu_g) \\ \delta\dot{b_g} &= \beta_g \delta b_g + \nu_{bg} \\ \delta\dot{b_a} &= \beta_a \delta b_a + \nu_{ba} \end{aligned} \tag{5.22}$$

$$\begin{aligned} \delta g^n &= [-g^n \times]\rho + R_G^B(\delta b_a + \nu_a) \\ \delta m^n &= [-m^n \times]\rho + R_G^B \nu_a \end{aligned} \tag{5.23}$$

where for a vector $v = [v_1,\, v_2,\, v_3]$, $[v\times]$ is defined as:

$$[v\times] = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}.$$

This system can be expressed in the standard EKF form:

$$\begin{aligned} \delta\dot{x} &= F(\hat{x})\delta x + G(\hat{x})w \\ \delta\dot{y} &= H(\hat{x})\delta x + v \end{aligned} \tag{5.24}$$

The orientation error $\delta\rho = [\delta\phi, \delta\theta, \delta\gamma]^T$, is used instead of a quaternion based error to avoid computational difficulties during the propagation step of the EKF, and it is defined as:

$$R_G^B = (I + [\rho\times])\hat{R}_G^B\,. \tag{5.25}$$

Using the same prediction and update frequency, at every sample the new orientation is predicted integrating the gyro readings and then corrected with the accelerometer and magnetometer readings. The predicted value $\hat{x}^-$ is obtained evaluating equation 5.20 and then the filter is linearized and discretized into the standard form:

$$\begin{aligned}\delta x_{k+1} &= \Phi \delta x_k + w_k \\ \delta y_{k+1} &= H_k \delta x_k + v_k\end{aligned} \tag{5.26}$$

where Φ_k is the transition matrix, H_k is the output matrix, w_k and v_k are white noise sequences with covariance Q_k and R_k. The system 5.26 is used to run an EKF update and find the error δx_{k+1} used to correct the predicted state to x^+_{k+1} and compute the covariance of updated estimate P^+_{k+1}. Full details of the filter implementation can be found in [69].

Complementary filter

Complementary filtering is based upon the use and availability of multiple independent noisy measurements of the same signal. If the measurements have complementary spectral characteristics, transfer functions may be chosen to minimize the estimation error. The general requirement is that one of the transfer functions complement the sum of others. In the case of sensor-based orientation estimation, we have a situation where complementary filters can be directly applied. As was noted, we have two independent estimations of the orientation with different characteristics: the accelerometer and magnetometer one is reliable in static conditions and is affected by high frequency noise, while the gyroscope one is reliable in dynamic conditions, but is affected by a slowly varying bias and integration drift. Thus an appropriate CF for orientation estimation should incorporate low-frequency information from the first one and high frequency information from the second one. A block diagram of such a filter is shown in Fig. 5.3, where φ_{am} is the orientation estimation from accelerometer and magnetometer, $\dot{\varphi}_w$ is the derivative of the one from the gyroscope and φ_{est} is the final estimate. A more detailed theoretical background on the CF can be found in [22, 44].

Compared with the KF, the CF approach has the advantage of being simpler from the computational point of view and having a smaller number of parameters, allowing an easy tuning for the use in a wide range of situations. Considering also the accurate orientation estimation provided, comparable or even better than

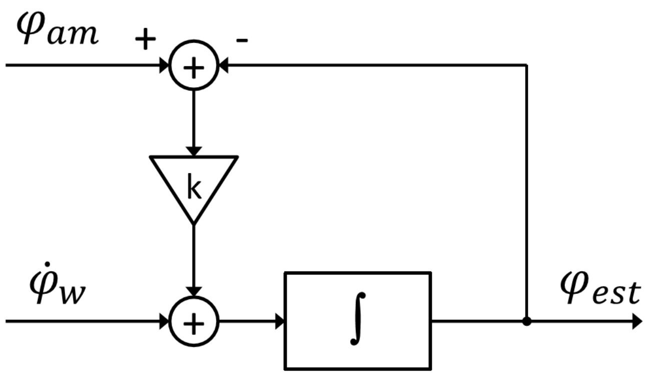


Figure 5.3: Block diegram of a complementary filter for orientation estimation.

the KF one, the CF is an appealing solution and is subject of intense research activity. Starting from the CF approach, several filter implementations have been developed, especially for aerial vehicles navigation, where its use for orientation estimation has first been adopted [116, 123]. There are also some examples of its usage for body limbs orientation estimation, such as [22, 114].

Given the promising performance/computational cost tradeoff of this approach, we adapted two recent algorithms for our embedded platform and compared them with the KF and EKF. In particular, we compared the nonlinear observer for attitude estimation by Mahony et al. [116] and a quaternion-based filter that incorporates magnetic distortion and gyroscope bias drift compensation proposed by Madwick [114].

5.4 EXL-s1 Sensor Node

This section will describe the prototype of the EXLs1 wireless sensor node employed to acquire information about the user's movements and posture. The description of the device is divided into hardware and software perspectives.

5.4.1 Hardware

The prototype of the sensor node is shown in Figure 5.4. As shown in the block diagram depicted in Figure 5.5, the wearable device is controlled and supervised by

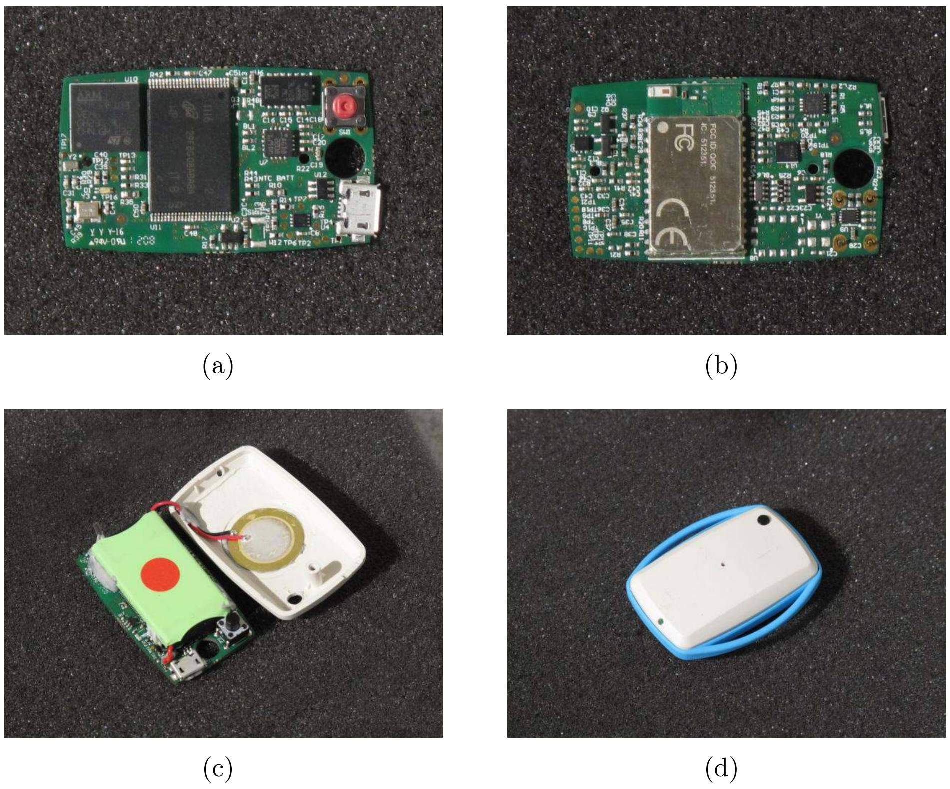
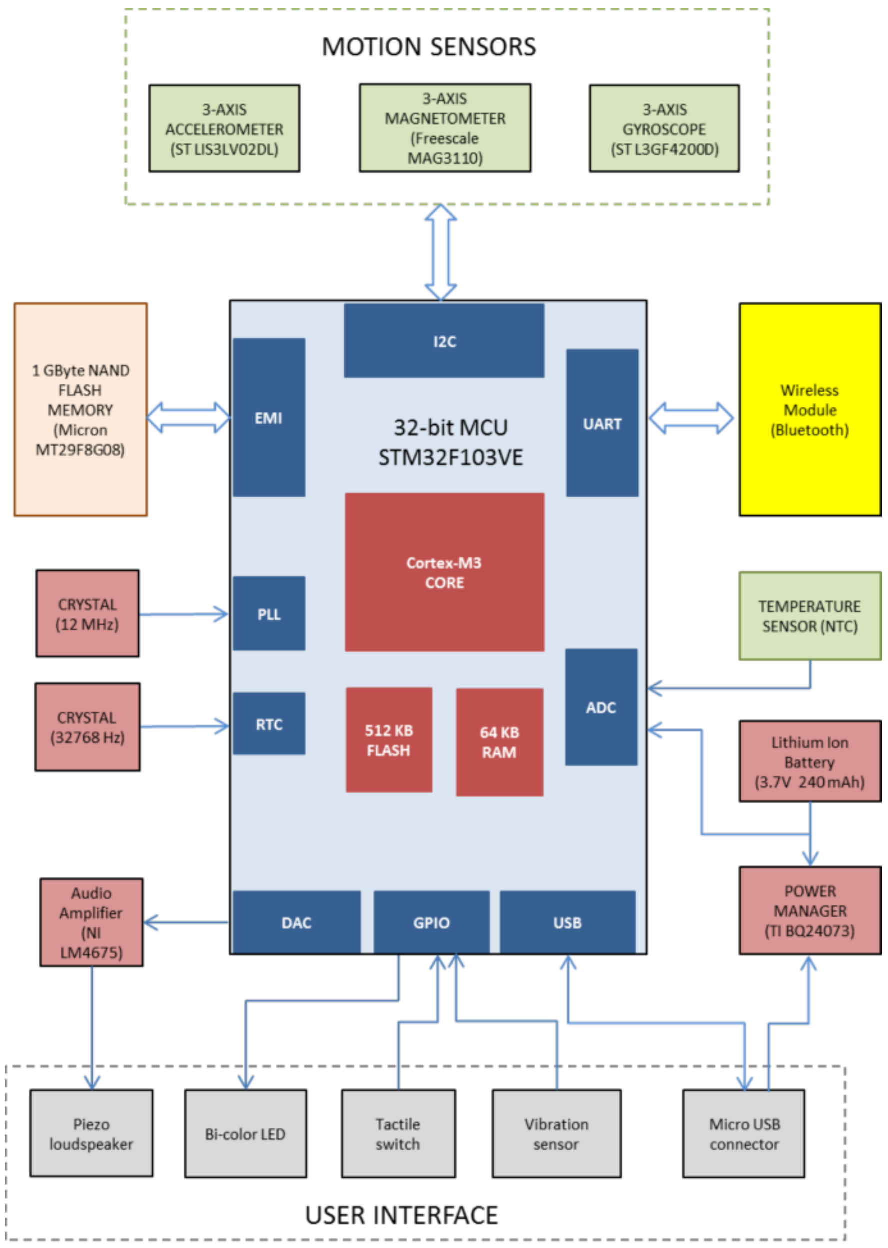


Figure 5.4: Hardware prototype of the *EXLs1* sensor node: (a) top side, (b) bottom side, (c) assembly with the battery, (d) the final device with the enclosure.

a 32-bit microcontroller unit (MCU) which is surrounded by several peripherals required to achieve the desired functionalities. The communication between the MCU and the peripherals is performed using different technologies and communication channels. In the following paragraph more details about each functional section are given.

Microcontroller unit

As MCU, the *STM32F103VE* from ST Microelectronics' portfolio has been chosen. It belongs to the STM32 family, powered by an ARM Cortex-M3 RISC core and combines a high degree of integration and performances with a low price and low power consumption. It runs at a clock frequency of 72 MHz and, besides the presence of on-chip 512 KB flash memory for code storage and 64 KB of RAM, it features the presence of a broad range of peripherals such as UART, SPI, USB,



MOTION SENSORS
3-AXIS ACCELEROMETER (ST LIS3LV02DL)
3-AXIS MAGNETOMETER (Freescale MAG3110)
3-AXIS GYROSCOPE (ST L3GF4200D)
1 GByte NAND FLASH MEMORY (Micron MT29F8G08)
I2C
EMI
32-bit MCU STM32F103VE
UART
Wireless Module (Bluetooth)
Cortex-M3 CORE
CRYSTAL (12 MHz)
PLL
TEMPERATURE SENSOR (NTC)
CRYSTAL (32768 Hz)
RTC
512 KB FLASH
64 KB RAM
ADC
Lithium Ion Battery (3.7V 240mAh)
POWER MANAGER (TI BQ24073)
Audio Amplifier (NI LM4675)
DAC
GPIO
USB
Piezo loudspeaker
Bi-color LED
Tactile switch
Vibration sensor
Micro USB connector
USER INTERFACE

Figure 5.5: Block diegram of the EXLs1 sensor node.

EMI, DMA, RTC, I2C, ADC, PWM, DAC, and timers that well fit our hardware and software requirements, thus minimizing the external components count.

Motion Sensors

The wearable IMUs are usually physically attached to the body segments (trunk, arms, legs) of the patient and their main task is to track the body segment movements and orientation. This is accomplished by using a set of MEMS sensors embedded on the device which acquire basic inertial measurements. Subsequently such data is processed in order to extract movement parameters and orientation. The sensor set is composed of a 3-axis accelerometer, a 3-axis gyroscope and a 3-axis magnetometer giving the raw measurements of linear acceleration, angular rate, and terrestrial magnetic field. The communication between the sensors and the MCU takes place by means of I2C shared bus, since all three sensors have a digital interface.

User Interface

Due to the ergonomic requirements of the system, the interaction between the wearable device and the user has been kept as simple as possible. There is only one button (tactile switch) that can be used in different ways, depending on the software implementation. It might be used, for instance, to control the activation/deactivation of the device or just to label specific situations that the patient encounters. A multicolor LED can be used to signal specific operating conditions, such as *idle*, *operating* or *battery low*, by changing the color (red/yellow/green) and the blinking frequency. Furthermore the piezoelectric loudspeaker can be used to give additional audio signaling.

In the current prototype a USB connector has been provided to allow for a standard communication channel towards a controlling PC. This is especially useful during the development process, since it allows fast firmware upgrading and and an easy download of collected data (in addition to the wireless communication channel). The USB port is also used for battery recharging with standard the 5V voltage.

Data Storage Memory

To enable the device to operate in a continuous data logging mode, a NAND Flash Memory with adequate capacity has been included. A Micron *MT29F8G08* 1GByte single-level NAND Flash memory, which includes an asynchronous data interface for high-performance I/O operations, was chosen. This device uses a highly multiplexed 8-bit bus (DQx) to transfer commands, addresses and data. There are only five control signals used to implement the asynchronous data interface with the MCU (CE, CLE, ALE, WE, RE). This hardware interface creates a low pin-count device with a standard pin-out that remains the same from one density to another, enabling future upgrades to higher densities with no board redesign. The chosen capacity is sufficient to cover a broad range of scenario requirements. For instance, using a sampling frequency of 100Hz for all sensors, an overall storage capacity of approximately 130 hours is available.

Wireless module

The integration of communication capabilities between the sensor node and a host node (PC or smartphone) is a key requirement for several user scenarios. It enables collecting data generated by the end nodes worn by the patient in the central processing unit for processing and storage. In our prototype we included the Bluegiga WT12 module, a fully integrated Bluetooth 2.1+ EDR, class 2 module, combining antenna, Bluetooth radio and an on-board *iWRAP* Bluetooth stack. It is connected to the MCU through a USART serial port and can be used to stream sensor data to a host device or to download the data saved in the node's memory. In addition a wireless bootlaoder has been adopted to allow the programming of the device without the need of particular connections.

5.4.2 Software

A sensor-based system for motion capture implies the use of one or more wearable devices attached to the user's body in order to accurately track and analyze its movements. For this scope, each wearable device has to collect the data from the different on-board sensors (accelerometer, gyroscope, magnetometer), to optionally process the sampled data and to send them via Bluetooth to the

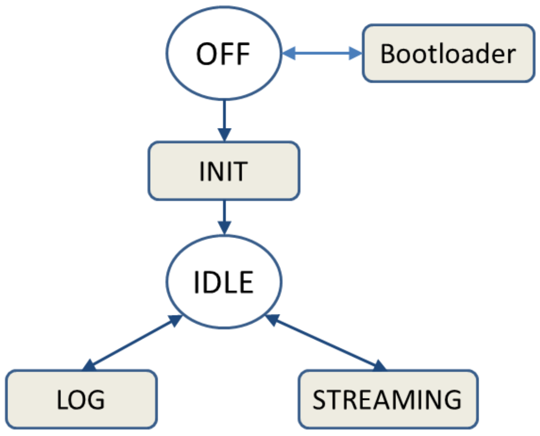


Figure 5.6: Block diagram of the firmware flow on the EXLs1 sensor node.

host node, which can further processes them and build a global model. In the following sections we will illustrate the functionalities of the nodes and their software.

General Firmware Architecture

Given the device hardware configuration illustrated in the previous section, the basic operation modalities are the streaming of sensor data to a host device and the logging of the sampled data on the integrated flash memory. Both operations can be performed in a stand-alone scenario or using up to 7 devices connected in a network. The shared flow of the executed operations is presented in Figure 5.6.

The sensor node, turned on by pressing its button, executes the initialization routine (*INIT*) and goes to an *IDLE* state waiting for user input. Another button press or a command received via Bluetooth can select one of the two available applications: data logging or data streaming. The initialization routine ensures to turn on and properly configure all the needed hardware components. At this stage, the microcontroller configures its internal peripherals: the General Purpose Input/Outputs (GPIO) used for the button and the LED, the I2C serial bus used to communicate with the sensors and the USART serial port to communicate with the Bluetooth module. The sensors are configured next, setting internal parameters such as the desired full scale and sampling frequencies. Some operation parameters of the nodes can be updated by sending configuration strings

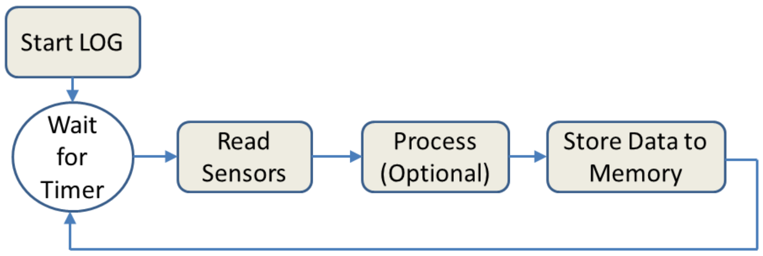


Figure 5.7: Block diagram of the logging application.

via Bluetooth: for example, it is possible to update the nodes' calendar clock or the sensor's sampling frequencies and full-scale values. For major changes to the application it is possible to upload a new version of the firmware using a custom bootloader. The bootloader can be accessed when turning on the device, by a long press of the power button. It allows uploading a new firmware simply by sending the desired binary file from a host device via the Bluetooth connection. After the update, the device restarts and executes the new application.

Data Logging

In the data logging mode, the node samples the sensors, optionally pre-processes the data and stores it in the on-board flash memory. In this modality the data is not streamed via Bluetooth, therefore saving energy. The data logging mode can be started both via Bluetooth, by sending the corresponding command (*Start LOG*) or by pressing the button on the device while it is in the *IDLE* state. With the latter method, the device does not need an active Bluetooth connection and can operate independently. The operation is terminated when the corresponding stop command is received or by another press of the button. Figure 5.7 summarizes this operation mode. In this modality, the node waits for a timer interrupt to signal the desired sampling period. After this, the current sensor data is read by the microcontroller and the desired processing algorithm is applied if needed. According to the application, the sensor data can be stored as sampled or after the application of a processing step, chosen between several algorithms such as low-pass filters, feature extraction or orientation estimation algorithms. The data stored in memory can later be downloaded to a PC via the Bluetooth connection, for offline processing and analysis.

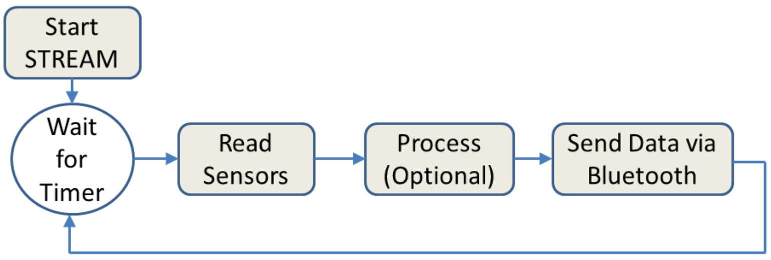


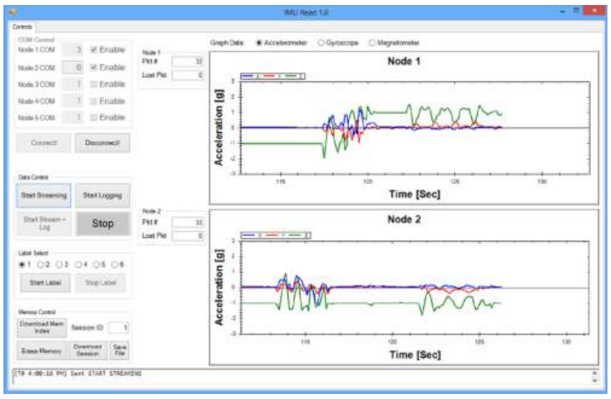
Figure 5.8: Block diagram of the streaming application.

Streaming

In the streaming mode, a node is connected via Bluetooth to a host device (PC or smartphone), which sends commands to start/stop the streaming of the sensor data. After a start command is received, the device samples the sensor data and applies the desired processing algorithm if needed, exactly as in the logging mode. What is different is that now the device streams the desired data to the host device, which can be used in real-time for further processing or interactive applications. This operation mode is illustrated in Figure 5.8.

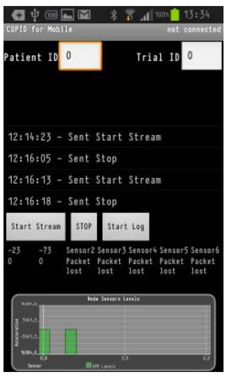
Networked Operations

Combining more than one device, we can build a Bluetooth network and collect/log data from up to 7 nodes at the same time. For the streaming application there is no difference compared to the single-node case: a host PC connects to the desired devices, sends them the command to start/stop streaming and then handles the incoming data streams. The Bluetooth protocol implementation sets a limit to the number of connectible nodes to 7. In practice, depending on the desired sampling frequency, the communication may fail for that many sensors. Our tests indicated that it is not possible to communicate with more than 5 nodes sampling at 70Hz connected to a PC running a custom application on Windows (see Fig. 5.9(a)). With an Android smartphone, it was possible to stream data from 7 nodes sampling at 100Hz, using again a custom application shown in Fig. 5.9(b). These differences are due to different driver and Bluetooth implementations on the two tested platforms. Synchronization of the data gathered with a PC is ensured by a timestamp recorded when receiving data packets. On a mobile phone, the arriving data packets are buffered and processed in blocks, so further



```
[18 4:00:16 PM] Sent START STREAMING
```

(a)



```
12:14:23 - Sent Start Stream
12:16:05 - Sent Stop
12:16:13 - Sent Start Stream
12:16:18 - Sent Stop
```

```
-23  -73  Sensor2 Sensor3 Sensor4 Sensor5 Sensor6
0    0    Packet  Packet  Packet  Packet  Packet
          lost    lost    lost    lost    lost
```

(b)

Figure 5.9: Screenshots of the developed applications for data collection and logging: (a) for Windows PCs and (b) for Android smartphones.

processing is needed to ensure packet-level synchronization.

In the logging mode, one of the devices acts as the master and initiates the synchronized recording session. It is possible to have one of the nodes with a dedicated *master* firmware, a Personal Computer or an Android phone to initiate the network, connect to the desired slave nodes and send the start/stop log commands. In this case no data is streamed through the network: every node stores the desired data in the internal memory, together with the shared network clock counter, retrieved every 15 minutes for synchronization purposes. This operation takes about 90 ms and no data is logged during its execution. It allows to synchronize the logged data from the network of nodes with a precision that can reach 25ms. If the node is controlled by a mobile phone or a computer, additional data such as patient ID, recording session and trial date/time can be stored. In the current implementation, with one gigabyte of memory, each node can store up to 130 hours of sampled sensor data at 100Hz.

Data Processing

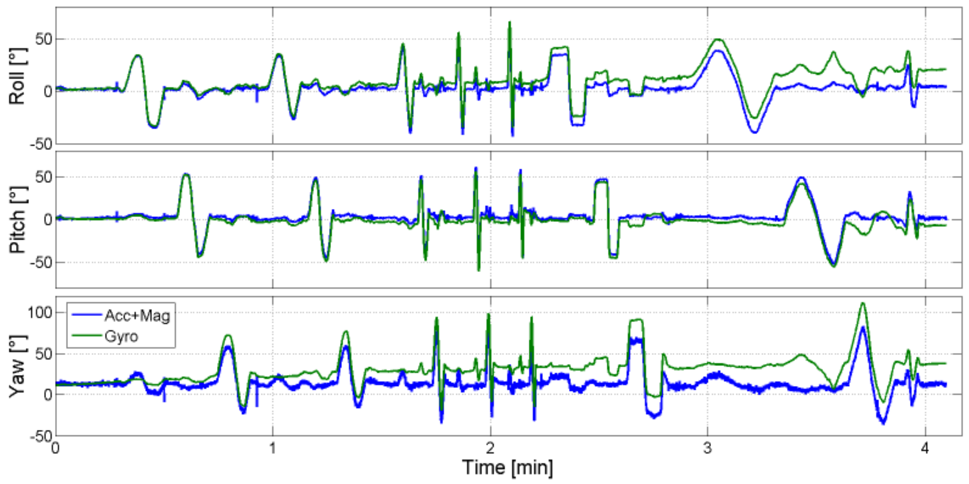
When employing more than one sensor with a central collecting node, the overall performance of the system can be optimized by processing locally the sampled data on each node. This solution avoids the creation of a communication and computation bottleneck on the central node by exploiting the computational capabilities of the embedded microcontroller available on each node and optimizing the operations to achieve better energy efficiency.

Inertial sensor data can be used to estimate the orientation of the device, providing implicit information on the posture of the user, given the location of the sensors on the user's body. With the available inertial sensors, each node can compute its orientation using different methods, such as the integration of the angular velocity provided by the gyroscope or the tilt-compensated electronic compass which employs accelerometer and magnetometer data. Both solutions have limitations and several sensor fusion algorithms have been developed to combine their outputs.

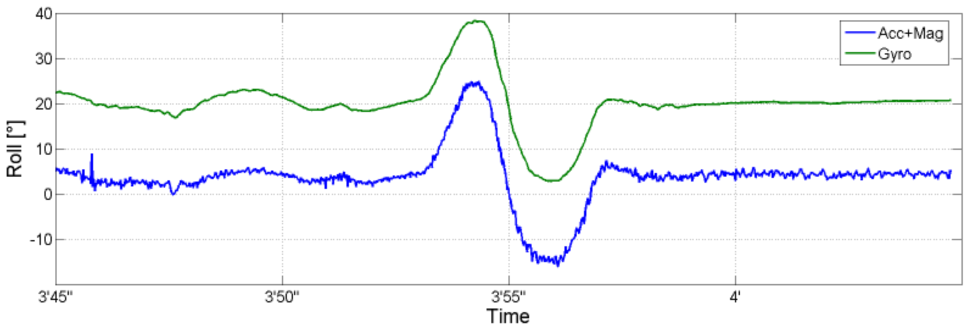
5.5 Experimental Setup and Results

In this section we will introduce the experimental analysis performed with the EXLs1 sensor node to evaluate its hardware characteristics and to test the different algorithms. The node was designed to study human movements, therefore we evaluated its performance when subject to different types of motion.

To efficiently test and compare orientation estimation algorithms, sensor data from the desired nodes was sent to a PC via the Bluetooth interface and logged for offline analysis. We first implemented in Matlab all the different algorithms and used the logged data to compare them in terms of orientation estimation accuracy. For this purpose we created an application for the collection and log of data from several nodes. In particular we created a Windows application to display and log the data coming from up to five nodes simultaneously, and an Android application, capable to collect data from up to seven nodes. Two screenshots of the applications can be pictured in Fig. 5.9. The Android application was used to test the sensors in a mobile scenario, where the user walked around wearing the sensors and carrying the phone, thus not having range constraints. Before all the operations, the sensor nodes used were calibrated using the procedure illustrated in Section 5.3.1.



(a)



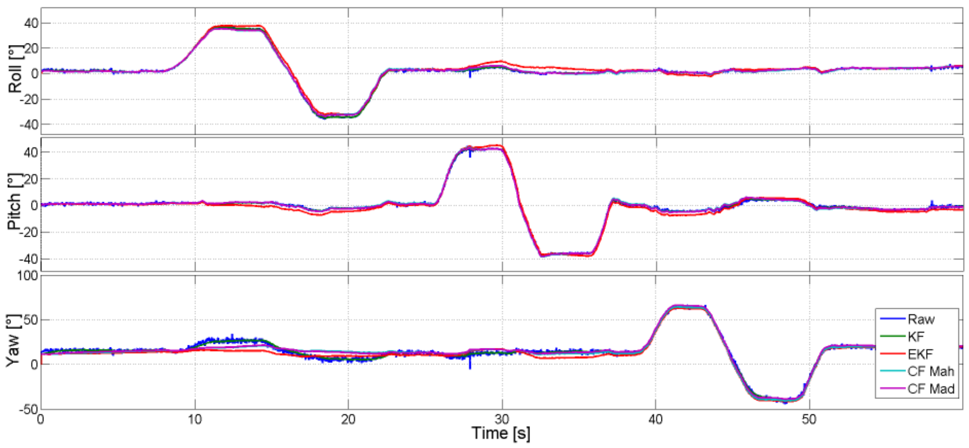
(b)

Figure 5.10: Comparison of the two algorithms to compute orientation from sensor data (a) and a zoom of the final part of the *roll* plot (b).

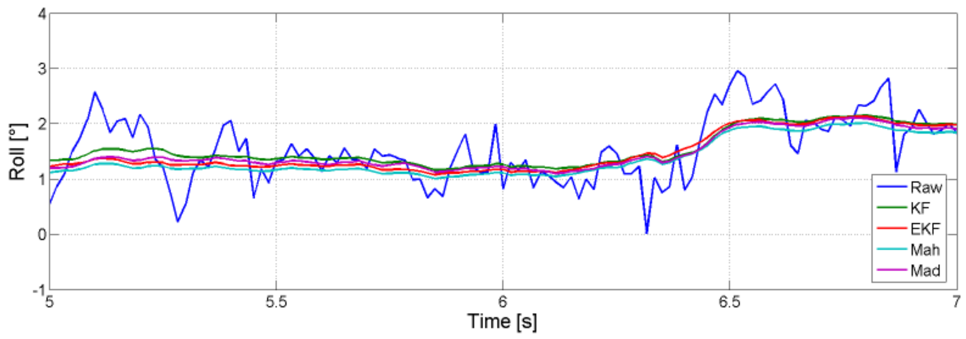
Orientation estimation

To test the different algorithms we collected several sessions of sensor data, while a user was manipulating the device by hand. All the proposed algorithms use quaternions to perform the needed computations. For an easy interpretation of the results we converted the output of the algorithms to Euler angles *roll*, *pitch* and *yaw*, corresponding respectively to rotations around x, y and z axes of the device.

In Figure 5.10 we can see a comparison between the two approaches to compute the orientation from sensor data. Here we compare the output of the trigonometric computation of orientation from accelerometer and magnetometer data



(a)



(b)

Figure 5.11: Comparison of different orientation estimation algorithms (a) and a zoom of the *roll* plot (b).

with the orientation computed integrating the angular rates from the gyroscope. During the experiment, the sensor node was held in hand by the user who performed small rotations around each axis, with different speeds. We can observe how the inertial output suffers from a high frequency noise, while the gyroscope integration leads to a drift of the estimated orientation. This result shows clearly a need for a sensor fusion approach to optimally combine the two estimates for a better final result. This can be better observed in Fig. 5.10(b), where the last portion of the previous figure is shown. During the 4 minutes of the experiment the gyroscope estimate accumulated and error of more than 15°.

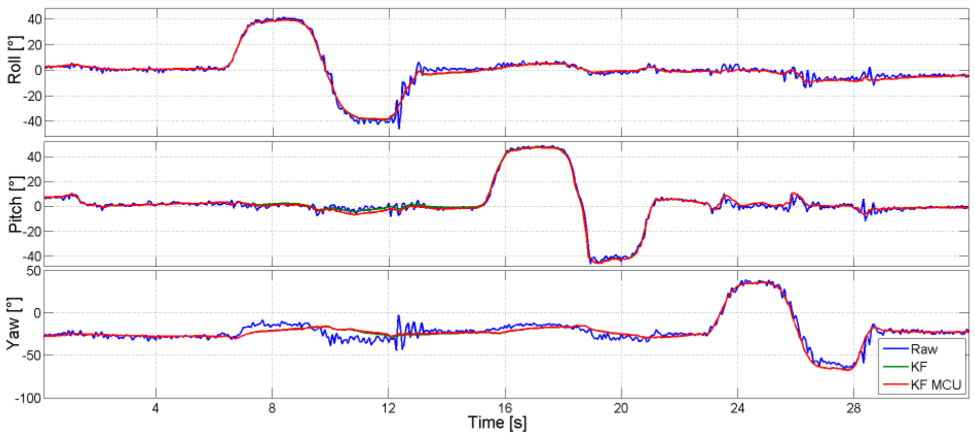
To achieve a better estimation of the orientation from our device, we tested the sensor fusion algorithms introduced in Section 5.3.3. For this purpose we first compared the algorithms in Matlab on a PC, using logged sensor data as input, then we implemented the diferent algorithms on the device, to compare their performance and computational costs. The result of this comparison can be seen in Figure 5.11. In particular we compared the Kalman filter (labeled as KF), the Extended KF (EKF), the complementary filter proposed by Mahony (CF Mah) and the one proposed by Madgwick (CF Mad). The estimate from the accelerometer and magnetometer was added to the plot for a direct comparison (labeled as RAW). We can observe how all the algorithms overcome the problems seen in the two individual estimations and achieve a comparable performance in terms of accuracy of orientation estimation. This can be further observed from the magnified portion of the plot shown in Fig. 5.11(b).

Embedded implementation

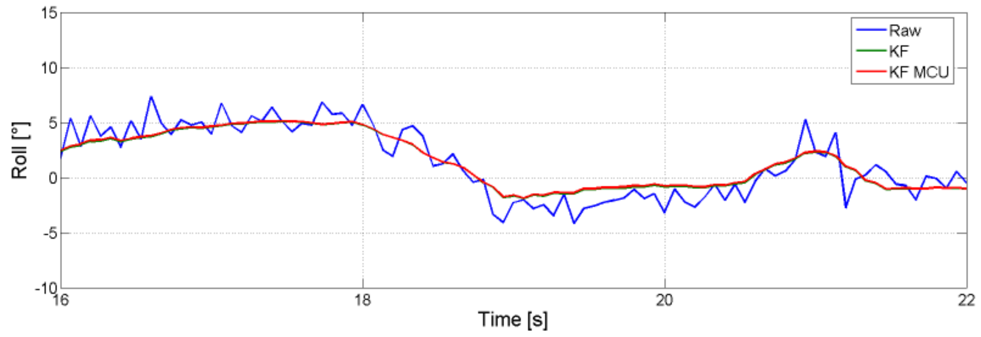
Next we implemented the sensor fusion algorithms for the ARM Coretex microcontroller embedded in the sensor node. This MCU has 32-bit precision and has no floating point unit. This means that all floating point operations are implemented in software using integer values, thus leading to longer computational times. On this platform we used the IEEE 754 standard to represent 32-bit single precision floating point values and the ARM compiler to handle the software implementation of the operations. We tested the precision achieved on the MCU against the Matlab implementation, which uses 64-bit double precision and was computed on the PC. For this purpose, both the sensor data and the computed quaternions were sent from the device to the PC, to allow the elaboration of the same stream of data on the two platforms. The results of this comparison are shown in Fig. 5.12 for the KF algorithm and in Fig. 5.13 for the Madgwick CF.

From the figures we can see how the MCU implementation of the two algorithms follows exactly the Matlab one. The two plots are bearly distinguishable and from a quantitative evaluation of the two implementations we found a mean deviation of less than $0.01°$, with peak differences always below $1°$. We observed the same results also for the other two implemented algorithms.

The MCU implementation of the orientation estimation techniques were also compared in terms of computational cost, measured as the number of CPU cycles



(a)

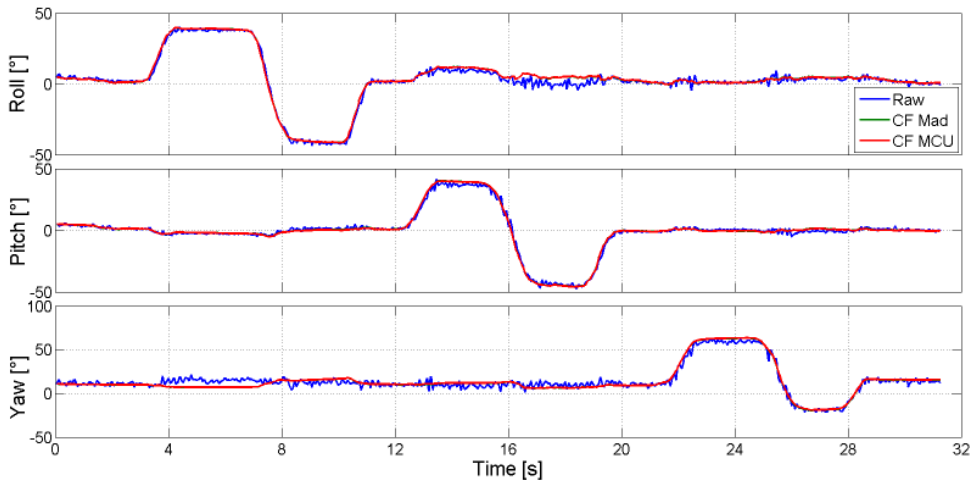


(b)

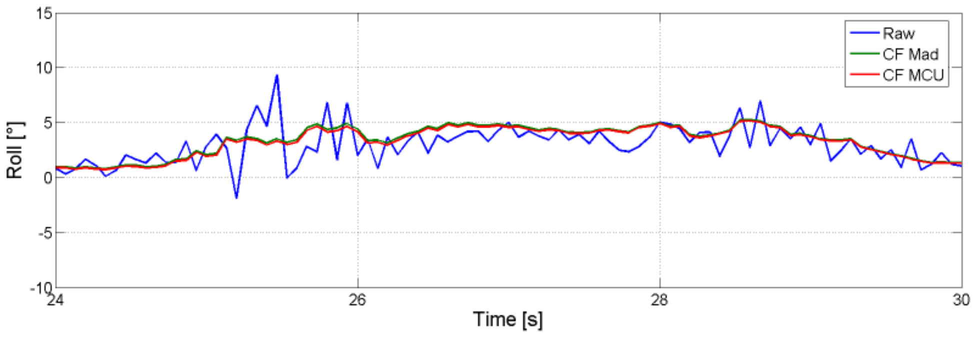
Figure 5.12: (a) Comparison of orientation estimation as computed by MATLAB (green) and by the microcontroller (red) for the KF. The raw MATLAB estimation was plotted as reference (blue). (b) Zoom of the *roll* plot.

needed to compute one iteration of the algorithm. The results of this comparison are shown in Table 5.1. We can see how the gyroscope integration is several times faster than the accelerometer and magnetometer one. This is due to the nature of the two algorithms, since the latter one needs the computation of trigonometric function $atan2$.

The different sensor fusion techniques showed almost identical performance in terms of orientation estimation, but they have very different computational costs. The EKF is the most computationally expensive technique, needing $72ms$ to perform every loop of the filter, while the two CF approaches have similar execution times of about $0.22ms$. The KF solution has an intermediate computational cost



(a)



(b)

Figure 5.13: (a) Comparison of orientation estimation as computed by MATLAB (green) and by the microcontroller (red) for the Madgwick CF. The raw MATLAB estimation was plotted as reference (blue). (b) Zoom of the *roll* plot.

needing $2.5ms$ to perform one iteration. The relatively high execution times of the Kalman filter approaches are due to the fact that they update the covariance matrix and the Kalman gain at every iteration. This operation leads to multiple matrix multiplications and to the inversion of a matrix, operating on matrices with dimensions equal to the dimension of the state vector. Moreover, the EKF solution dynamically estimates the bias of the inertial sensors, leading to a state vector of 9 elements and thus to operations on 9×9 matrices. In our analysis this increased computational cost is not justified by an effective increase in estimation accuracy, making this algorithm a poor choice for embedded platforms.

Table 5.1: Computational cost of the different algorithms (Times for MCU running at $72MHz$).

Algorithm	**CPU Cycles**	**Time [ms]**
Acc+Mag	73393	1.019
Gyro Int.	5224	0.073
KF	179115	2.489
EKF	5184020	72.000
CF Mah	16196	0.225
CF Mad	16294	0.226

Motion analysis

To test the device in a real-use scenario we used it to analyze the gait of a user. For this experiment two devices were secured on top of the shoes of the user, as can be seen in Figure 5.14. The user, a male 28 years old student performed a natural walk along a hallway, while a smartphone logged the data coming from the two sensors. On the collected data we applied the proposed techniques to estimate the orientation of the sensor during the motion.

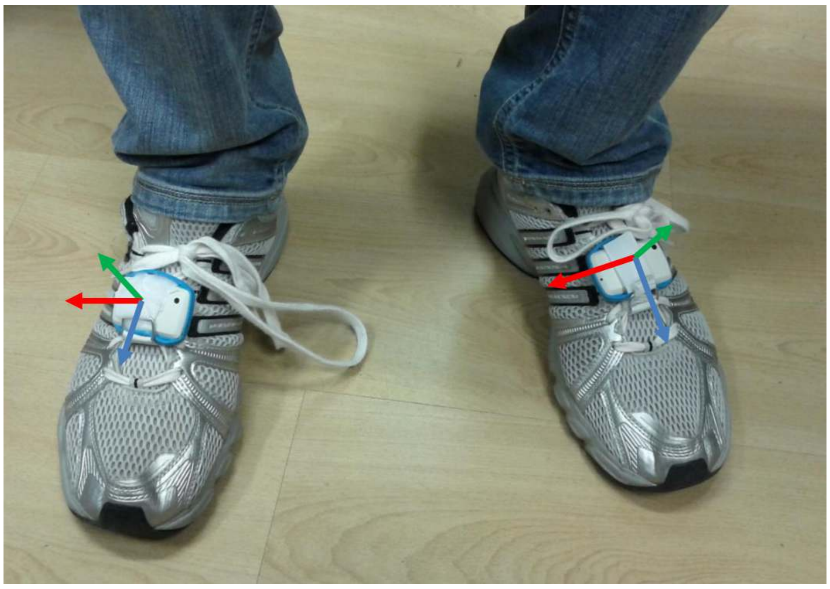
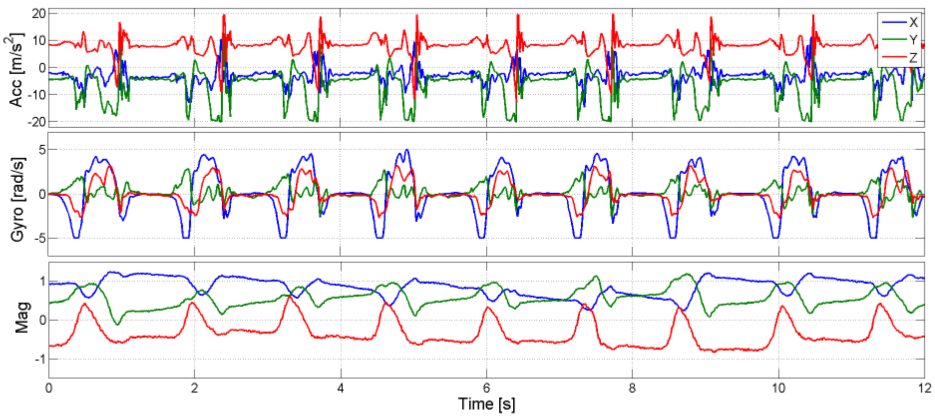
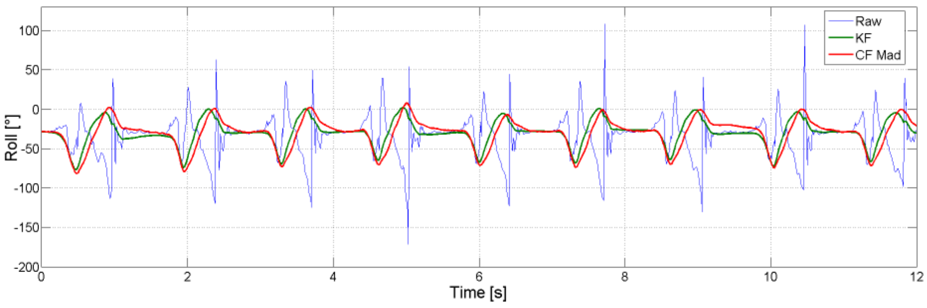


Figure 5.14: EXLs1 sensor nodes attached to the shoes. The body frames of the two sensors are aligned with x pointing to the user's right (in red in the figure), y forward (blue) and z up (green).



(a)



(b)

Figure 5.15: (a) Sampled sensor data and (b) roll angle as estimated by the KF and the CF Mad algorithms while the user was walking.

The results for the sensor mounted on the right shoe of the user are shown in Figure 5.15. Here we can observe (a) the log of the gathered data and (b) the resulting orientation of the *roll* angle as computed by the KF and the Madgwick CF. The results show how both of the filters compute correctly the orientation of the foot during the walk. This is a highly dynamic motion, during which the sensor node is subject to high inertial accelerations. This leads to high disturbances in the accelerometer-based orientation estimation, showed as a reference. Both sensor fusion algorithms overcome this problem by the fusion of the information provided by the gyroscope. We can further observe how the CF suffers from a small lag when compared to the KF orientation. This lag can be compensated by increasing the weighting coefficient of the filter, at the expense of the introduc-

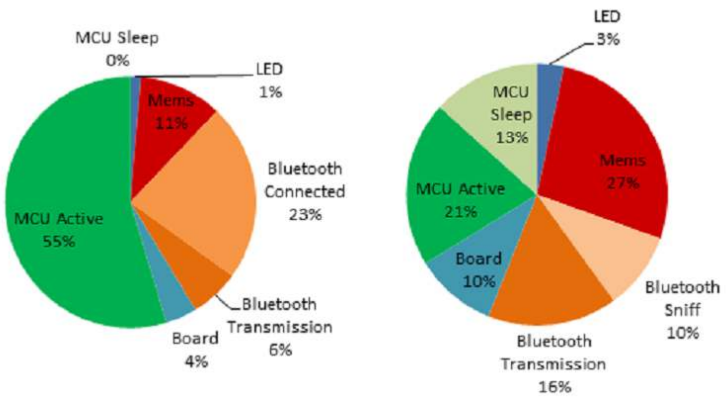


Figure 5.16: Power consumption breakdown associated with each task for the default case (left) and the optimized one (right).

tion of a major amount of sensor noise from the accelerometer. For an accurate evaluation of the two algorithms, an external independent estimation of the orientation would be needed, which can be implemented through a high precision vision system.

Power consumption

Sensor nodes rely on a battery for the power supply. In the current hardware configuration a 180mAh LiPo battery is used and, according to the energy policy used, it is possible to have different node lifetimes. If no policy is applied, a node can run up to 2 hours using a single charge when in continuous streaming mode, sampling the sensors at a 100Hz rate. In this case the MCU is set to the highest frequency available (72MHz) and guarantees the maximum computing performances. When it is not necessary to use the Bluetooth communication or when there is no need to perform data analysis on the node (e.g. in data logging mode), it is possible to put all the unused components in a sleep mode. When using appropriate sleep modes for the available peripherals and reducing the MCU clock frequency, the battery life increases up to 6 hours, reducing the average current consumption from 75mA to 29.8mA. The figure 5.16 illustrates the breakdown of the power consumption in the two cases, highlighting the percentage of the power consumed for each task and for the sleep states.

5.6 Conclusions

In this Chapter we presented the hardware and software development of a wearable sensor node for the tracking of human motion. The *EXLs1* sensor node embeds state of the art integrated inertial (accelerometer and gyroscope) and magnetic sensors, features an advanced ARM microcontroller and a Bluetooth transceiver. The device was designed to be small and versatile for an easy use in a broad range of applications, such can be human-computer interaction, motion capture or heathcare. The availability of the Bluetooth connection enables the device to communicate with a wide range of systems and allows the use of multiple nodes in a body sensor network.

To analyze the data from the on-board sensors, we compared several algorithms for orientation estimation. For a better estimate, sensor fusion algorithms such as the Kalman filter and the complementary filter are employed to optimally integrate the information from the on-board sensors. The use of several nodes to monitor articulated movements arises the issues of the simultaneous processing of the data streams, favoring a distributed solution where every node processes locally its sensor data. Thus, we implemented the proposed estimation techniques on the microcontroller embedded in the device, and tested their performance in terms of accuracy and orientation estimation. The results showed how the different algorithms have very similar estimation accuracy, while the Kalman filter-based solutions suffer from a considerably higher processing times. To achieve an efficient use of the limited energy provided by the battery integrated into the device, we carefully optimized the use of the available peripherals, taking advantage of power saving techniques.

Chapter 6

Energy Efficient Data Reconstruction for Wireless Sensor Networks

Wireless sensor networks (WSNs) are commonly recognized as one of the technological cornerstones of AmI. Agile, low-cost, ultra-low power networks of sensors can collect a huge amount of critical information from the environment. Using a biological analogy, a sensor network can be seen as the sensory system of the intelligent environment "organism". Sensor networks are irregular aggregations of communicating sensor-nodes, which collect and process information coming from on-board sensors, and they exchange part of this information with neighboring nodes or with nearby collection stations.

The exploration of WSNs and innovative technologies for their enhancement, from the data accuracy and power consumption viewpoints, is a natural evolution of the work presented in the first Chapters of this dissertation. Starting from single smart objects we explored body area networks, with a limited number of sensor nodes placed on the human body, and will now consider the case of WSNs composed by tens or thousands of nodes that sense the surrounding environment.

The recent evolution of sensing devices and the availability of new technologies for WSNs have triggered new research activities in the field of data gathering and compression. The goal or our work is to improve the performance/power consumption tradeoff: in a general WSN scenario this can be achieved by minimizing storage and communication costs to extend as much as possible the lifetime of the network, while maintaining the desired data accuracy. In this Chapter we will present two techniques capable to successfully recover the signal from highly incomplete sub-sampled versions: Compressive Sensing (CS) and a data-driven statistical model based on latent variables (LV). In this work, we focus on the comparison between these two approaches and their impact on the energy consumption of the network nodes. In particular, we evaluate their performance and power requirements and explore the impact of features such as the use of network wide correlations to improve the reconstruction accuracy.

6.1 Overview

Modern applications of WSNs typically require measuring several variables (such as temperature, humidity, light intensity, etc.) for extended periods of time over a large area. Examples of WSN applications include environmental monitoring [30], smart building [18], smart home solutions [66] or structural health monitoring [36]. To meet the application requirements, the design and deployment of a WSN has to carefully balance between two competing goals: (1) high spatio-temporal resolution to ensure the accuracy of the collected data, and (2) minimal energy consumption to maximize the network lifetime and limit node maintenance. The amount of data that a node collects and processes directly affects both its power consumption and the accuracy of the information obtained [53, 141].

Extending the uptime of a sensor node is an active topic of research in WSNs, especially when the nodes are deployed in difficult to access or remote locations. Common approaches try to enhance the battery life *directly* by harvesting energy from the environment and employing low-power hardware architectures [43, 47], or using improved wireless protocols and distributed computation for data processing [181]. More recently, researchers are optimizing the battery life *indirectly* by reducing the overall amount of sensed data [164]. Here, the data is selectively sampled according to a predetermined protocol, reducing the total amount of samples collected by the individual sensor nodes, thus minimizing the energy consumption. To maintain an acceptable amount of total measurements, the missing data is inferred according to statistical models that capture how the data evolves. In addition to enhancing the battery life, these approaches are also able to estimate any lost or corrupted data, making them a popular choice [164].

One of the most promising techniques capable to successfully recover the signal from highly incomplete sub-sampled versions is Compressive Sensing (CS) [25, 46, 63]. This technique is able to recover original signals from a smaller sub-sampled version obtained by skipping samples during the acquisition phase.

CS theory claims that if a signal can be compressed using classical transform coding techniques and its representation is sparse in some basis, then a small number of projections on random vectors contains enough information for approximate reconstruction [45]. Natural signals have usually a relatively low information content as measured by the sparsity of their spectrum [42], therefore the theory of CS suggests that randomized low-rate sampling may provide an efficient alternative to high-rate uniform sampling. This peculiar form of CS is a novel strategy to sample and process sparse signals at sub-Nyquist rate [119, 142].

In this work we propose a novel energy efficient, data-driven statistical model based on latent variables (LV) to estimate original signals from sub-sampled versions within a heterogeneous sensor network. Our approach extends the standard latent variable factorization model, which typically considers only dyadic interactions in data, to multivariate spatio-temporal data, by applying tensor decomposition techniques [101]. The key advantage of using a latent variable model is that it provides a compact representation of the gathered data that can be used to recover the missing samples. In order to perform well under extreme sampling conditions, we extend the standard technique to explicitly incorporate the spatial, temporal, and inter-sensor correlations.

To explore the efficiency of our approach, we analyzed two different scenarios. The first one is a structural health application for an aging building, employing low-power wireless sensor nodes. In this case the energy consumption of each sensor node is dominated by the power spent for the radio transmission of the gathered data. The second scenario is an environmental monitoring application, characterized by an energy model where the power required for data sampling contributes significantly to the overall energy consumption. Therefore, not only the amount of data sent through the wireless channel should be reduced, but also the amount of samples that a node collects should be minimized. Moreover, the two datasets exhibit completely different patterns and spatio-temporal resolutions in the gathered data, validating the general applicability of the proposed method.

This study focuses on the trade-off between the accuracy in recovering the missing data and the energy consumption when sensor nodes duty cycle to save energy. The proposed technique drastically reduces the amount of sampled data at each node, thus allowing the nodes to spend more time in a low-power sleep mode and save energy. The lower amount of sampled data implies a lower amount of data to transmit from the node to a central gathering station, reducing also the power consumptions associated with the radio communications.

We compared our technique with the CS approach and evaluated their impact on the energy consumption of the network nodes. The two techniques address the same problem from different theoretical bases, and hence a straightforward question is which one is most well suited as an energy-minimization technique for WSN. To the best of our knowledge this is the first work in which the two techniques are directly compared using the same dataset for evaluating the reconstruction quality.

For the tests and simulations presented we use real WSN deployments for both the node hardware characteristics and the gathered data. We evaluate the performance of the two techniques and explore the impact of features such as the use of network wide correlations to improve the reconstruction accuracy or the length of the data block to be reconstructed.

Our focus is on optimizing node sensing rates at different nodes to reduce the overall energy consumption. The communication network implements state-of-the-art low power architecture and routing protocols to get better energy efficiency. The reconstruction task is performed at the data collection center, thus reducing the complexity and energy consumption of the battery powered sensor nodes. The central node is connected to a power source and thus does not have energy constraints.

6.2 Related Work

The problem of data gathering, compression and signal reconstruction in WSNs is well explored in literature. Even though the majority of the works deal with reconstruction algorithms and mathematical aspects, practical aspects and low power implementation problems are lately gaining interest.

6.2.1 Theoretical approaches

As seen in the previous section, CS builds on several works like [45, 63] which show that if a signal can be compressed using classical transform coding techniques and its representation is sparse in some basis then a small number of projections on random vectors contain enough information for approximate reconstruction.

When CS is used in a scenario where several sensors acquire data from the same environment, we can think that the sensed data has a certain kind of shared information that can be exploited to perform a better reconstruction.

The most known technique used for exploiting the inter and intra-correlation among several nodes in a WSN is the Distributed CS (DCS) introduced in [29]. In these works authors analyze three different sparsity models (JSM-1, JSM-2, JSM-3) to describe most of the signals ensemble; and for each sparsity model they present a different reconstruction algorithm. Among other works dealing with joint sparse recovery we can cite [107] or Kronecker CS introduced in [62]. Differently from these works, in this paper we want to focus on a special technique for the reconstruction of jointly sparse solutions known as Multiple Measurement Vector (MMV) based on $\ell_{2,1}$-regularization that is introduced in [59].

Latent variables and decomposition-based techniques have also been proposed in literature, with their most notable applications in collaborative filtering and recommender systems [155, 177]. A standard way to learn a set of latent variables for a two dimensional dataset is to apply a matrix factorization technique. A recent review of matrix factorization techniques with applications to recommender systems can be found in [102]. To achieve a better latent variables model and to allow the prediction of the missing entries in the original data matrix, standard matrix factorization techniques have been extended to incorporate temporal dynamics and thus better capture the temporal evolution of the data [103, 166]. When dealing with multivariate datasets, which can be represented by $n-$dimensional data arrays, tensor factorization techniques can be exploited to further improve the performance of the models [95]. In our work, we applied a tensor factorization approach to WSNs, taking advantage of the three-dimensional nature of the data gathered from a heterogeneous sensor network. We further extended this approach, incorporating for each data dimension a correlation model learned from the data itself, to enhance the capabilities to reconstruct the missing data.

6.2.2 WSN-related practical implementations

The general problem of using CS in WSNs is investigated in several works like [139], where the authors analyze synthetic and real signals against several common transformations to evaluate the reconstruction performance. In [113] the measurement matrix is created jointly with data routing policies, trying to preserve a good reconstruction quality. Also in [115] the authors try to improve the reconstruction by reordering the input data to achieve a better compressibility. The main focus of these works is to investigate the signal reconstruction problem but what the authors think is missing is a consideration about how the usage of CS impacts on the power consumption. While there is no doubt that CS is a powerful technique for data size reduction and compression, its usage and impact on network lifetime, when real hardware and COTS nodes are used, are still marginally addressed in literature.

The work in [118] is one of the first papers trying to address the issue of energy consumption for data compression, dealing with the problem to generate a good measurement matrix using as low energy as possible. In this work the research is focused on Wireless Body Sensor Networks (WBAN) for real-time energy-efficient ECG compression. This is a quite different research field with respect to WSNs where the presence of several nodes sensing the same environment permits to exploit the distributed nature of the signals to improve the quality of recovery.

Several works like [109] or [93] deal with the use of CS when the data is gathered from different joint sources, using DCS to improve the recovery quality. In [117] DCS and Principal Component Analysis are used to reconstruct spatially and temporally correlated signals in a sensor network but, once again, the contribution of the power consumption for compression in the network lifetime is neglected.

In this paper, we deal with CS when the signals are sampled at sub-Nyquist frequency resembling a technique which is in literature referred to as analog CS. The name derives from the fact that the subsampling is performed at ADC level, dropping samples during the acquisition and analog-to-digital conversion stage. One notable example of this technique is in [180] where the effects of circuit imperfections in the analog compressive sensing architectures are discussed. In the framework proposed in this work, samples are not discarded by analogue circuits but are not sampled at all, saving the energy for waking up the node.

In literature, other works investigate the problem when the samples are discarded by the device performing the sensing rather than the ADC. For example, in [23] and [73] the analysis on energy consumption is totally neglected and the recovery is strictly related to the specific applications described. Differently from environmental signals, used in our work, the signals coming from the oximeter present a much higher temporal correlation, presenting small variations in their temporal evolution, facilitating their reconstruction.

Matrix factorization learning of latent variables has been used for recovering missing data in sensor networks in [166], where temporal correlations found in the dataset are used to infer the missing variables. Tensor decomposition techniques have been applied on WSNs in [138], where the learned models are used to find the damages in a structural health monitoring application. The previously presented algorithms only consider homogeneous sensor streams, dealing with one sensor at a time, and do not consider the energy costs across the network. Instead, our approach, focuses on the multivariate nature of the collected data, and it expands the tensor factorization techniques by employing spatio-temporal and intra-sensor correlations for more robust and better results than the existing methods. The energy consumption when sub-sampling is, instead, considered in [50] in which authors use a sparse generated matrix adjusting the sampling rate to maintain an acceptable reconstruction performance while minimizing the energy consumption. Differently from our work they do not consider inter- and intra-correlation among signals, and do not use any group-sparsity enhancing algorithm to perform a better recovery.

6.3 Compressive sampling and groups sparsity

Considered a continuous signal $x(t)$ of duration T, $x(n)$, $1 \leq n \leq N$ is its discrete version. The Nyquist theorem states that in order to perfectly capture the information of the signal $x(t)$, having a bandwidth of $B_{\text{nyq}}/2$ Hz, we must sample the signal at its Nyquist rate of B_{nyq} samples per second. In formula:

$$x(n) = x(t)|_{t=nT_s} \tag{6.1}$$

where $T_s \leq 1/B_{\text{nyq}}$ and $NT_s \leq T$. The sampled signal $x(n)$ is represented by an N-dimensional vector of real numbers $\mathbf{x}$.

If the vector $\mathbf{x}$ is sparse then CS is able to recover this finite-dimensional vector $\mathbf{x} \in \mathbb{R}^N$ from a very limited number of measurements of the original signal $x(t)$. The sparsity of a signal $\boldsymbol{\alpha}$ is usually indicated as the ℓ_0-norm of the signal where the ℓ_p-norm $\| \cdot \|_p$ is defined as:

$$\|\boldsymbol{\alpha}\|_p = \left(\sum_{i=0}^{N-1} |\alpha_i|^p \right)^{1/p} \tag{6.2}$$

with $\boldsymbol{\alpha} \in \mathbb{R}^N$.

If the signal $\mathbf{x}$ is sparse then there exists some $N \times N$ basis or dictionary $\boldsymbol{\Psi} \in \mathbb{R}^{N \times N}$ such that there is an N-dimensional vector $\boldsymbol{\alpha}$ implying $\mathbf{x} = \boldsymbol{\Psi}\boldsymbol{\alpha}$ and $\|\boldsymbol{\alpha}\|_0 \leq K$ with $K \ll N$.

CS theory demonstrates that it is possible to compress this kind of sparse signals using a second different measurement matrix $\boldsymbol{\Phi} \in \mathbb{R}^{M \times N}$ with $M \ll N$. The compression can be written as $\mathbf{y} = \boldsymbol{\Phi}\mathbf{x}$ where $\mathbf{y}$ is the M-dimensional measurements vector. While $\boldsymbol{\Psi}$ is usually defined by the signals characteristics, $\boldsymbol{\Phi}$ has to be designed so that M is much smaller than N.

Having the measurements vector $\mathbf{y}$, the recovery of the original signal $\mathbf{x}$ can be obtained by the inverse of the measurement problem

$$\mathbf{y} = \boldsymbol{\Theta}\boldsymbol{\alpha} = \boldsymbol{\Phi}\boldsymbol{\Psi}\boldsymbol{\alpha} \tag{6.3}$$

Even though the inversion is not an easy task since the matrix $\boldsymbol{\Theta} \in \mathbb{R}^{M\times N}$ is rectangular with $M \ll N$, the fact that $\mathbf{x}$ is sparse can relax the problem by opening the way to the use of optimization-based reconstruction or iterative support-guessing reconstruction.

The most common optimization-based method here reported for the sake of clarity is the *basis pursuit* (BP) [132] method that searches for the most sparse solution for which the $\|\boldsymbol{\alpha}\|_1$ is minimum:

$$\hat{\boldsymbol{\alpha}} = \arg\min \|\boldsymbol{\alpha}\|_1 \text{ s.t. } \mathbf{y} = \boldsymbol{\Theta}\boldsymbol{\alpha} = \boldsymbol{\Phi}\boldsymbol{\Psi}\boldsymbol{\alpha} \tag{6.4}$$

CS theory proves that if the two matrices $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ are incoherent (elements of the matrix $\boldsymbol{\Phi}$ are not sparsely represented in the basis $\boldsymbol{\Psi}$) and the original signal $\mathbf{x}$ is compressible or sparse we can recover $\boldsymbol{\alpha}$ with high probability.

In order to further enhance the recoverability, recent studies propose to take into account additional information about the underlying structure of the solutions [51]. When the signals to compress and recover are obtained from sensors deployed close to each other in the environment, we can expect that the ensemble of these signals presents an underlying joint structure. This characteristic can be exploited to further compress the data, without a loss in reconstruction accuracy. In practice, this class of solutions are known to have certain "group sparsity" structure. This means that the solution has a natural grouping of its components, and the components within a group are likely to be either all zeros or all non-zeros. Encoding the group sparsity structure can reduce the degrees of freedom in the solution, thereby leading to better recovery performance.

Having an ensemble of J signals we can denote each signal with $\mathbf{x}_j \in \mathbb{R}^N$ with $j \in \{1, 2, \dots, J\}$. For each signal $\mathbf{x}_j$ in the ensemble we have a sparsifying basis $\boldsymbol{\Psi} \in \mathbb{R}^{N\times N}$ and a measurement matrix $\boldsymbol{\Phi}_j \in \mathbb{R}^{M\times N}$ such that as before $\mathbf{y}_j = \boldsymbol{\Phi}_j\mathbf{x}_j$ with $M_j \ll N$ and $\mathbf{x}_j = \boldsymbol{\Psi}\boldsymbol{\alpha}_j$.

The reconstruction of jointly sparse solutions, also known as the multiple measurement vector (MMV) problem, has its origin in sensor array signal processing and recently has received much interest as an extension of the single sparse solution recovery in compressive sensing. The recovery problem can be formulated as:

$$\begin{aligned} \min_{\tilde{\boldsymbol{\alpha}}} \quad & \|\tilde{\boldsymbol{\alpha}}\|_{w,2,1} := \sum_{i=1}^{n} w_i \|\tilde{\boldsymbol{\alpha}}_i\|_2 \\ \text{s.t.} \quad & \tilde{\boldsymbol{\Theta}}\tilde{\boldsymbol{\alpha}} = \tilde{\mathbf{Y}} \end{aligned} \tag{6.5}$$

where $\tilde{\mathbf{Y}} = \left[\mathbf{y}_1^T \mathbf{y}_2^T \ldots \mathbf{y}_J^T\right]$, $\tilde{\boldsymbol{\alpha}} = \left[\boldsymbol{\alpha}_1^T \boldsymbol{\alpha}_2^T \ldots \boldsymbol{\alpha}_J^T\right]$, w_i is the weight and $\tilde{\boldsymbol{\Theta}} \in \mathbb{R}^{JM \times JN}$ is a matrix having on the diagonal matrices $\boldsymbol{\Theta}_j = \boldsymbol{\Phi}_j \boldsymbol{\Psi}$ for $j \in \{1, 2, \ldots, J\}$.

6.3.1 CS and sub-Nyquist sampling

As discussed in the previous section, to successfully recover the original signal from its sampled version, the samples are taken regularly on a time axis at a given rate that is not less that the Nyquist one. With respect to CS this requirement means that the measurement matrix $\boldsymbol{\Phi}$ is a dense matrix (usually a i.i.d. Gaussian matrix).

A particular form of CS, usually referred as analog CS, relies on random sampling and aims to produce a number of measurements fewer than with Nyquist sampling, still enabling the reconstruction of the original signal.

While analog CS is usually performed by means of specialized hardware encoders, this is also a suitable technique to be performed on WSNs nodes, opportunely skipping samples during acquisition phase.

From a mathematical point of view the problem is still the same as the problem in equation 6.3; what is different is the form of the measurement matrix $\boldsymbol{\Phi}$ that is not a dense matrix but it is a sparse measurement matrix.

More precisely if $\mathbf{B}$ is an M-dimensional vector where each element is a unique random entry between 1 and N then the matrix $\boldsymbol{\Phi}$ in the analog CS is a sparse $M \times N$ measurement matrix which is composed by an all-zero vector on each row and a "1" at the location given by the i-th element of $\mathbf{B}$. This is a very simple measurement matrix, energetically cheap to generate, store and it permits a huge reduction in the duty-cycling of the nodes.

6.4 Latent variables for data reconstruction

Latent variable based factorization is a simple yet powerful framework for modeling data, and has been successfully applied in several application domains [102]. The main idea behind this framework is to model the large number of observed variables (the *observed data*) in terms of a much smaller number of unobserved variables (the *latent variables*). The latent variables are learned from the observed data and are used to estimate the missing samples, modeling complex interactions between the observed variables through simple interactions between the latent variables.

More specifically, given some multivariate data that is collected by a heterogeneous WSN in a large field over time, we can naturally organize it in a three dimensional data array (or a 3-*tensor*, as shown in Figure 6.1 left). Each of the three dimensions corresponds to a different variate of a particular measurement (e.g. the time, the location and the sensor type associated with each reading). Once the data is organized in this way, we can now associate a low-dimensional *latent* variable with each unique location, time slice and sensor type. We can thus model a particular observation (at a given location, time and type) as a noisy combination of the associated latent variables. In many scenarios, a multiplicative combination of these latent variables is able to capture intricate dependencies in the data [17, 101]. The goal then is to learn a good set of latent variables (that is, find a factorization) that can efficiently represent our observed data.

6.4.1 Modeling details

We now define our model more formally. For each unique time instance t, sensor type s, and node location n, we associate a unique K-dimensional vector a_t, b_s and c_n respectively. These unobserved vectors are called the latent factors or variables, and are assumed to control the location-, time- and sensor-specific interactions present in the observed data.

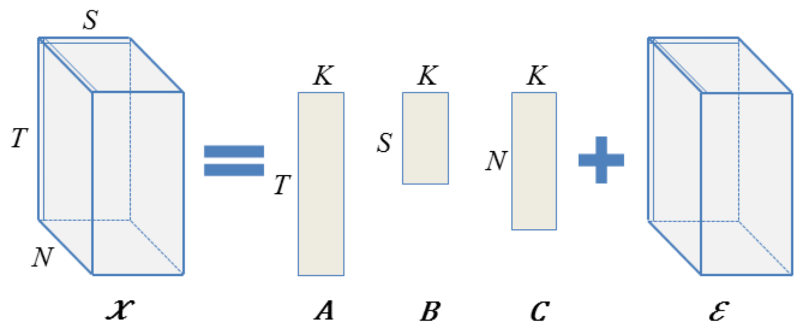


Figure 6.1: Tensor factorization model. Left: A $[T \times S \times N]$ tensor representation of WSN data from S different sensor types collected at N different locations at T different times. Each entry in the tensor is modeled as a combination the associated latent (unobserved) variables (A, B and C) plus noise (ε).

Then, given a $[T \times S \times N]$ tensor $\mathcal{X}$ of sensor readings from S different sensor types collected at N different nodes and T different time instances, with possible missing entries, we model $\mathcal{X}$ as follows. We assume that each reading x_{tsn} (reading at time t, for sensor type s, at node location n) is a noisy realization of the underlying true reading that is obtained by the interaction of the time specific latent variable a_t, with the sensor specific latent variable b_s and with the location specific variable c_n. That is,

$$x_{tsn} = \sum_{k=1}^{K} a_{tk} b_{sk} c_{nk} + \varepsilon, \tag{6.6}$$

where ε is modeled as independent zero-mean Gaussian noise ($\varepsilon \sim \mathcal{N}(0, \sigma^2)$). Observe that under this model once all the latent variables are known, one can recover the true readings of all sensors at all locations and times. Thus the goal is to find the most predictive set of vectors a_t, b_s and c_n for all $t = 1, \ldots, T$, $s = 1, \ldots, S$, $n = 1, \ldots, N$. Such a representation models the entire data of size $[T \cdot N \cdot S]$ by just $[K \cdot (T + N + S)]$ modeling parameters. The choice of the free parameter K provides a key trade-off: a large K increases the number of modeling parameters and thus can help model the observed data exactly. But this lacks the capability on predicting unobserved/missing data due to overfitting. A small

K, on the other hand, escapes the overfitting problem, but the corresponding model lacks sufficient richness to capture salient data trends. The exact choice of a good K is typically application dependent and is derived empirically.

6.4.2 Learning the latent variables

Finding the optimal set of K-dimensional latent variables given the observations is equivalent of factorizing the given tensor into three matrices each of rank at most K [101]. Thus, assuming that all the data is known (that is, every entry in the tensor is observed), we can find the latent factors by employing the CanDecomp/ParaFac (CP) tensor factorization. This is simply a higher-order generalization of the matrix singular value decomposition (SVD) [101], and decomposes a generic third order tensor in three matrix factors A, B, and C. By restricting the ranks of each of the matrix factors to at most K, yields the best rank K approximation. Algorithmically, the matrix factors are typically found by an alternating least squares approach (ALS) [17], which iteratively optimizes for one matrix factor at a time, while keeping the other two fixed.

This technique can be generalized to work with tensors that have missing entries. Since sensor nodes can periodically go offline due to duty-cycling or run out of energy (preventing all sensors on a node from collecting any data for an extended period of time), we need to extend our basic model to deal with data missing from multiple sensors or nodes, resulting in entries and rows of missing data in the collected tensor. In order to do well in this regime we extend the basic tensor factorization model to explicitly incorporate spatial, temporal and sensor specific information from neighboring observations by explicitly learning and enforcing the corresponding correlations.

6.4.3 Incorporating correlations

To successfully interpolate the sensor interactions to contiguously missing blocks of data, we need to explicitly model spatial, temporal and sensor-specific trends within each of our latent variables a_t, b_s and c_n. Such trends ensure that the latent variables a_t and $a_{t'}$ (respectively b_s and $b_{s'}$, and c_n and $c_{n'}$) take similar values when times t and t' are "similar" (respectively sensors types s and s', and locations n and n'). Note that similarity can mean anything based on the context. For locations, it can mean that variables associated two locations that are close in distance should have similar characteristics, while for time, it can mean that variables associated with times that are same hour of the day or same day of the week should have similar characteristics. Here we will take a data driven approach to infer the best notion of similarity using correlations directly computed from the data.

The similarity constraints are modeled in the same way for all the three sets of latent variables, and here we illustrate the case for the a_t's. Since each a_t is a K-dimensional variable, let a_t^k denote its k^{th} coordinate. We model a_t^k (independently for each coordinate k) as

$$\begin{aligned} a_{:}^{k} &= \mu_a^k + \alpha_{:}^{k} \\ \alpha_{:}^{k} &\sim \mathcal{N}(0, \Sigma_a). \end{aligned} \tag{6.7}$$

Here $a_{:}^{k}$ represents the collection of all a_t's (across $t = 1, \ldots, T$) in the k^{th} coordinate and μ_a represents their mean value. The distributional constraint over $\alpha_{:}^{k}$ (as $\mathcal{N}(0, \Sigma_a)$) enforces the similarity constraints via the $T \times T$ covariance matrix Σ_a. By changing the t, t' entry of Σ_a we can encourage/discourage the corresponding a_t and $a_{t'}$ to take similar values – a high positive value at $\Sigma_a(t, t')$ encourages a positive correlation, a high negative value encourages negative correlation, while a value close to zero does not encourage any correlation.

To get the right similarity constraints Σ_a, Σ_b and Σ_c (for latent variables a_t, b_t and c_n), we compute the empirical correlations from the data. That is, for spatial similarity constraints we computed the averaged pairwise Pearson correlation coefficient between data from different pairs of locations (across sensors and times). We do the same to approximate inter-sensor and temporal similarities.

6.4.4 Parameter learning

We can learn the underlying latent variables in a probabilistic framework using a *maximum a posteriori* (MAP) estimate. In particular, let θ denote all the model parameters (i.e. $\theta = \{\{a_t\}, \{b_s\}, \{c_n\}, \sigma\}$), then the optimum choice of parameters θ_{MAP} given the data $\mathcal{X}$ is obtained by:

$$\begin{aligned}\theta_{MAP}(\mathcal{X}) &:= \underset{\theta}{\operatorname{argmax}}\ \underbrace{p(\mathcal{X} \mid \theta)}_{\text{likelihood}}\ \underbrace{p(\theta)}_{\text{prior}} \\ &= \underset{\theta}{\operatorname{argmax}} \sum_{t,s,n \in \text{observed}} \log p(x_{tsn} \mid a_t, b_s, c_n, \sigma) + \\ &\quad \sum_{k=1}^{K} \log p(a_:^k) + \sum_{k=1}^{K} \log p(b_:^k) + \sum_{k=1}^{K} \log p(c_:^k).\end{aligned} \tag{6.8}$$

The first term (the likelihood) takes the form of Eq. 6.6, and the other terms represent the priors for each latent variable and each one of them takes the form of Eq. 6.7. We take a uniform prior over σ, the standard deviation of the residuals in Eqn. 6.6 so it doesn't explicitly show in the equation.

This optimization does not have a closed form solution and standard gradient based techniques can be used to get a locally optimal solution. Here we can do an alternating hill-climb approach by optimizing the value of one variable while keeping all others fixed to get a good solution.

6.5 Hardware, network and power models

To evaluate the proposed techniques we use real-world data from two sensor networks, with different hardware configurations and spatio-temporal resolutions. The composition of the two networks is summarized in Table 6.1. The number of physical sensors can be smaller than the number of reported variables because some sensors record multiple variables of interest.

Our first data set comes from an environmental monitoring WSN from the California Irrigation Management Information System (CIMIS) [54]. It is a program of the California Department of Water Resources that manages a network of 232 automated weather stations displace across the state of California. Each station provides hourly readings of 12 different measurements from its embedded sensors.

Dataset	Nodes	Sensors per node	Variables Reported	T_S [min]
CIMIS	128	7	10	60
3ENCULT	23	3	3	10

Table 6.1: Analyzed datasets.

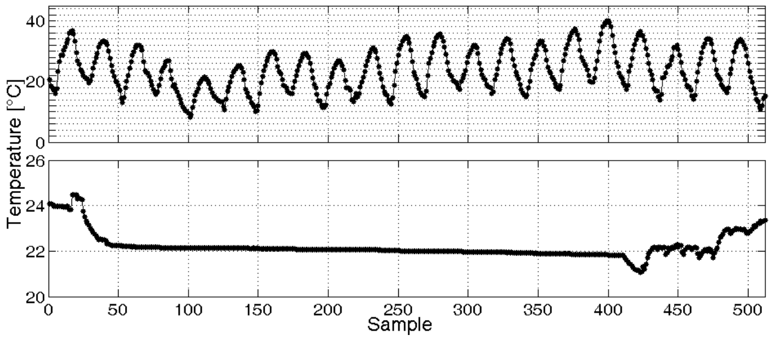


Figure 6.2: Comparison of the temperature signal collected from the CIMIS network (above) and the 3ENCULT network (below).

The second data set comes from a case study of the 3ENCULT European project [16]. This is a structural health monitoring application where a network of 23 low-power sensor nodes is deployed across the three floors of the historic building *Palazzina della Viola* at the University of Bologna.

The two data sets have different sampling periods (one hour for CIMIS and ten minutes for 3ENCULT) and different spatial coverage and distribution (state-wide coverage for CIMIS and indoor coverage of a 3-store building for 3ENCULT). Table 6.2 summarizes the variables reported by each station for the two datasets. Even if some variables are shared among the two datasets, the network characteristics, the nature of the captured signals and the resulting correlations are different because of the differences in the sensed environment. As an example, we can observe the Figure 6.2 where a block of 512 samples of temperature readings from the two data sets is reported.

Variables	**CIMIS**	**3ENCULT**
Solar Radiation $[W/m^2]$	X	X
Net Radiation $[W/m^2]$	X	
Soil Temperature $[°C]$	X	
Air Temperature $[°C]$	X	X
Pressure $[kPa]$	X	
Wind Speed $[m/s]$	X	
Wind Direction $[0 - 360°]$	X	
Precipitation $[mm]$	X	
Ref. ETo $[mm]$	X	
Rel. Humidity $[\%]$	X	X

Table 6.2: Sensor list for the different datasets.

6.5.1 Hardware

We used the hardware configuration of the nodes in the two data sets to estimate each node's power consumption. For the environmental monitoring stations in the CIMIS dataset we used the Vaisala WHT250 sensor station [161] which has detailed power consumption information available. This type of station is used at several locations in the CIMIS network.

The node employed in the 3ENCULT network is a wireless node by ST Microelectronics (STM32W108) that is a System on Chip with a $24GHz$ IEEE802.15.4-compliant transceiver integrated on die. The CPU is a 32-bit $24MHz$ ARM Cortex-M3 equipped with $128KB$ of Flash memory and $8KB$ of RAM. The set of sensors considered is composed by a Sensirion SHT21 (temperature and humidity sensors) and a BH1715 Light Sensor. Timing in performing the operations used in the power model are obtained either using the values reported in the datasheet or measured using a GPIO trigger connected to an oscilloscope. Data on power consumption of the various subsystems are not reported for lack of space but the reader can refer to the datasheets of the components for further reference [7, 14, 15, 161].

6.5.2 Network model

In a sensor network we can consider that each node samples the signals for a period of time T, called acquisition period, ideally gathering $N = T/f_s$ number of samples at a f_s sampling frequency, before sending the data towards the collecting point. If each node adopts a sub-sampling policy with an under-sampling ratio ρ then the number of samples actually gathered by the node is $M = \rho N$.

The under-sampling pattern is locally generated by each node using its own id and the timestamp as seed for randomization. In the random sampling pattern the inter-measurements intervals are always multiple of the minimum sampling period $T_k = T/N$.

6.5.3 Power model

We introduce an architecture level power model to evaluate the energy consumption of the node when the subsampling parameters are changed. Using this power model with data from real hardware and measurements, we can easily evaluate how changing the parameters influences the energy consumption and the lifetime of the network.

Starting from the assumption reported in the previous section, the average energy consumption in each period of duration T_k, for a sub-sampling factor ρ, is:

$$\begin{aligned} E_k =& \rho\left(E_{\text{setup}} + E_{\text{sampl}} + E_{\text{store}}\right) + E_{\text{sleep}} + \\ & N^{-1}\left(E_{\text{nv}} + E_{\text{send}}\right) \end{aligned} \tag{6.9}$$

where E_{sleep} is the energy spent in sleep mode, E_{setup} is the energy used for waking up and setting up the device, E_{sample} is the energy for sampling each sensors, E_{send} is the energy used to send the acquired data, E_{store} is the energy to store the acquired sample in non volatile memory and E_{nv} is the energy spent during the recovery of the data from non volatile memory.

Expanding each term we have:

$$\begin{aligned}
E_k = \rho(&T_{\text{setup}} \cdot (P_{\text{mcu}} + P_{\text{soff}} + P_{\text{toff}}) + \\
&T_{\text{sample}} \cdot (P_{\text{sample}} + P_{\text{sactive}} + P_{\text{toff}}) + \\
&T_{\text{store}} \cdot (P_{\text{soff}} + P_{\text{toff}} + P_{\text{store}})) + \\
&T_{\text{sleep}} \cdot (P_{\text{sleep}} + P_{\text{soff}} + P_{\text{toff}}) + \\
&N^{-1}(T_{\text{nv}} \cdot (P_{\text{store}} + P_{\text{soff}} + P_{\text{toff}}) + \\
&T_{\text{send}} \cdot (P_{\text{comm}} + P_{\text{soff}} + P_{\text{send}}))
\end{aligned} \tag{6.10}$$

where $T_{\text{sleep}}, T_{\text{setup}}, T_{\text{sample}}, T_{\text{send}}, T_{\text{store}}, T_{\text{nv}}$ are the duration of each respective phase. P_{sleep} is the power consumed in sleep mode, P_{soff} is the power absorbed from sensors when sleeping, P_{toff} is the power consumption of the transceiver when the node is in sleep mode. P_{mcu} is the power consumed by the MCU, P_{sample} is the power spent for data acquisition, P_{sactive} is the power consumed by sensors, P_{comm} is the power consumption for filling the transceiver output buffer and finally P_{send} is the power for sending data.

For the CIMIS dataset the values for the power consumption and timings were extracted from the datasheets, while for the 3ENCULT dataset they were measured on real hardware and reported in Table 6.3.

6.6 Results

6.6.1 Sensor power consumption

We used the hardware configuration of the nodes in the two data sets to estimate each node's power consumption. For an environmental monitoring station we used the Vaisala WHT250 sensor station [161] which has detailed power consumption information available. Since the WHT250 does not embed any radio, we assume that the radio communication is performed through a commercially available Zigbee wireless transceiver (XBee PRO [60]).

The energy consumption of this type of sensor station is summarized in Table 6.4. This data is obtained from the components' data sheets, considering

T_{sample}	600 [s]	Sampling Period
T_{setup}	5e-4 [s]	Setup time
T_{store}	5.25e-5 [s/16bit]	Time to store data in NVM
V_{batt}	3.3 [V]	Battery voltage
A_{sleep}	1.3e-6 [A]	Sleep current
A_{mcu}	7.5e-3 [A]	Current consumption in idle
A_{sample}	1.1e-3 [A]	Current consumption for ADCs
A_{store}	7.5e-3 [A]	Current consumption when saving in NVM
A_{comm}	1e-6 [A]	Current for filling the transceiver buffer
F	24 [MHz]	Microcontroller frequency
A_{send}	3.1e-2 [A]	Current consumption for transmission
A_{toff}	10e-6 [A]	Sleep current of the transceiver
S_{tx}	150 [kbps]	Transmission throughput
$T_{\text{setup\_radio}}$	5e-3 [s]	Transceiver setup time
B_{pkt}	127 [byte]	Packet size
B_{header}	10 [byte]	Header size
$A_{\text{sactive,SHT21}}$	3e-4 [A]	(SHT21) Current consumption
$A_{\text{soff,SHT21}}$	1.5e-7 [A]	(SHT21) Sleep current
$T_{\text{sampl,SHT21}}$	2e-5 [s]	(SHT21) Sampling time
$A_{\text{sactive,BH1715}}$	150e-6 [A]	(BH1715) Current consumption
$A_{\text{soff,BH1715}}$	0.01e-6 [A]	(BH1715) Sleep current
$T_{\text{sampl,BH1715}}$	1e-5 [s]	(BH1715) Sampling time

Table 6.3: Characteristics of the device taken as reference in the power model.

Sensor or State	Power [mW]	Time [s]	Energy [mJ]
Sleep (all node)	1.2	-	-
CPU Active	42.9	-	-
Wind speed & direction	20	60	1200
Pressure	9.6	5	48
Temperature	9.6	5	48
Humidity	9.6	5	48
Rain	0.84	10	8.4
Xbee - Data transmission	158.4	0.05	8

Table 6.4: Power consumption of the different components of an environmental station. The energy consumption refers to a single sampling event or a single packet transmission.

the average consumption for each operation, and using the World Meteorological Organization (WMO) specifications for environmental data gathering [176]. The wind measurements (speed and direction) need to be reported in 1 minute increments, thus costing a lot of energy. Pressure, temperature and humidity sensors have the same power consumption ($9.6mW$) as they are part of the same module, but each sensor has to be read individually. With an 1 hour sampling period, the energy consumed at each interval (E_{SAMPLE}) is $5.62J$, 75% of which is spent in the sleep mode. Of the active energy, 96% is consumed for the sampling of the sensors, 3.5% for data transmission and 0.5% by the CPU. Our model considers all the contributions even if the sampling energy is the dominating component in this scenario.

For the 3ENCULT sensor network, the energy consumption results are summarized in Table 6.5. In this case the radio transmission represents the main source of energy consumption. There is $3.6mJ$ of energy per each 10 minutes sampling period, of which almost 85% is for the sleep mode. Of the active power, 88% is consumed by the radio transmission of the acquired packets, 10% by the CPU and 2% by the sensors.

In both cases the influence of the energy spent in the sleep mode increases with an increase of the sampling period, since the node spends more time waiting for the next interval. With longer periods, even aggressive duty cycling policies

Sensor or State	Power [mW]	Time [ms]	Energy [mJ]
Sleep (all node)	0.578	-	-
CPU Active	24.7	-	-
Temperature	0.495	0.002	0.001
Humidity	0.495	0.002	0.001
Light	0.495	0.002	0.001
Data transmission	102.3	0.05	5.115

Table 6.5: Power consumption of the different components of the W24TH sensor node.

will have a smaller benefit in terms of energy saving, since they reduce only the active energy spent for sampling and data transmission.

6.6.2 Data preprocessing

The proposed statistical model treats data from each heterogeneous sensor equally. Since raw data from different sensor types are at widely different scales (e.g. the temperature ranges from about 10 to $40°C$, while relative humidity measurements range from 0 to 100%), we preprocess all the variables to ensure they have zero mean and unit variance. This normalization brings all measurements to the same scale and allows us to apply our multivariate tensor factorization technique. The normalized prediction can easily be translated back into the original scale by rescaling and adding back the mean value. The reconstruction error is evaluated in the original scale for each variable, adopting the Normalized Root Mean Squared Error ($NRMSE$) defined as:

$$NRMSE = \frac{\sqrt{E[(\hat{\boldsymbol{x}} - \boldsymbol{x})^2]}}{x_{max} - x_{min}},$$

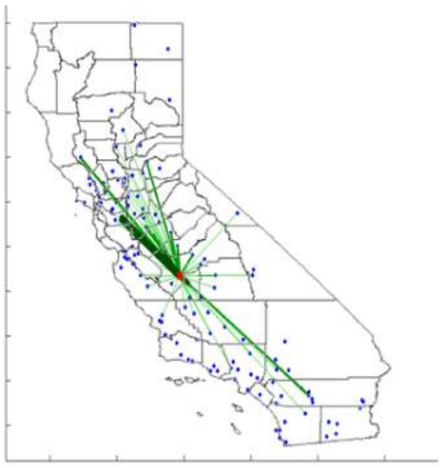
where $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ are respectively the actual and predicted data for each variable, x_{max} and x_{min} are the maximum and minimum values for the same variable.

6.6.3 Estimating spatial, temporal and inter-sensor correlations

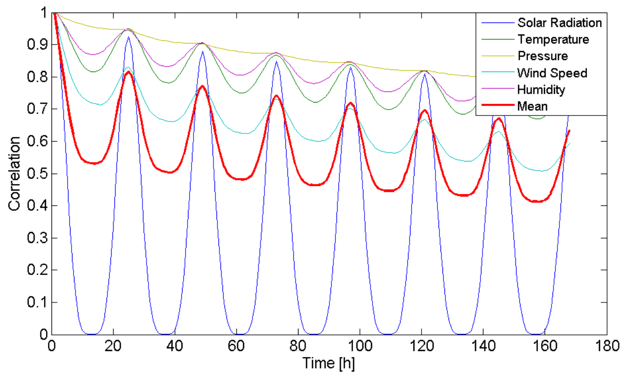
Here we present the results of the correlation analysis as discussed in Section 6.4.3. Figure 6.3 illustrates the case of the CIMIS dataset. Figure 6.3(a) shows the spatial correlation between the readings from one sensor node (marked in red) to all the other nodes, plotted over a map of the node locations. The thickness of the line that connects two nodes in the plot shows the strength of the correlation. For this analysis we averaged the readings between the different sensors at the various locations. From this figure we can see how node correlations are not strictly proportional to the physical distance between different nodes. This is primarily because since the nodes are located several hundreds of miles away from each other, the climatic similarity (mountainous regions vs. desserts) has a greater influence on the node similarity, rather than the raw distance. A similar correlation is computed for all the network nodes. We use these correlations to seed our covariance matrix Σ_b to enforce spatial similarities between the nodes (cf. Section 6.4.3). Similar results are obtained for the 3ENCULT dataset, where the indoor nature of the deployment emphasizes even more how node correlations are not strictly proportional to the distance between nodes.

Figure 6.3(b) presents the temporal correlation between the samples in the time series. The reported values are averaged among the different sensors and node locations. Observe that most variables show a strong periodic behavior with a 24 hour period. In the 3ENCULT case this periodic behavior is mitigated by external factors such as room occupancy and human activity, resulting in a rapidly decaying correlation curve. The computed correlation is used as the covariance matrix Σ_a to enforce the observed temporal similarity on the latent variables a_t.

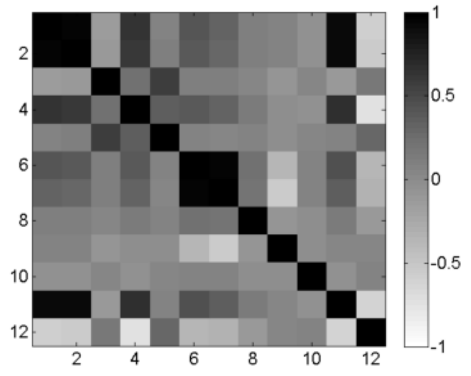
The inter-sensor correlations are presented in Figure 6.3(c) (the axises represent the variable ID as per Table 6.2). Note how values from different sensors can be inversely correlated. Observe that variables such as solar radiation, net radiation, air temperature and evapotranspiration (variable IDs 1, 2, 3, and 11 respectively) are directly correlated with each other, and are inversely correlated with the atmospheric pressure (variable ID 12).



(a)



(b)



(c)

Figure 6.3: The spatial, temporal and inter-sensor correlations for the CIMIS dataset. (a) Spatial correlation of a fixed node with all the other nodes over a state-wide map. Thick lines indicate strong correlation between the location pairs, while thin lines indicate weak correlations. (b) Temporal correlations along a week of sampled data. (c) Inter-sensor correlations. The axises represent the sensor IDs as ordered as in Table 6.2.

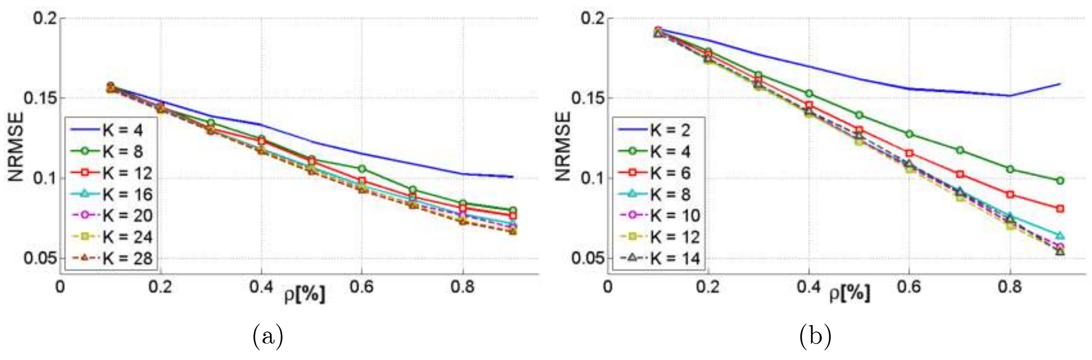


Figure 6.4: Comparison of tensor factorization reconstruction of data for different values of K: (a) CIMIS dataset and (b) 3ENCULT

6.6.4 Tensor factorization

Here we evaluate the effectiveness of our tensor-based latent variable factorization technique to model our datasets. In our approach, the data collected from the sensor network is organized in a $[T \times S \times N]$ tensor $\mathcal{X}$ and is factored in three matrix factors A, B and C of size $[T \times K]$, $[S \times K]$ and $[N \times K]$. This leads to a model with $K(T + S + N)$ parameters for a dataset with $T \cdot S \cdot N$ entries.

The latent variables dimensionality, K, regulates the complexity of the model. To evaluate the impact of K, different models with increasing complexity (increasing values of K) were learned from the data, using the standard CP tensor factorization technique. The whole dataset is reconstructed using the learned model and the $NRMSE$ between the real data and the reconstructed one is evaluated. This operation is repeated for different fractions of sampled data ρ used to learn the model. The $NRMSE$ shows how well the collected data can be summarized by a few latent variables, thus exploring the compression capabilities of our approach. The results for the two datasets are illustrated in Figure 6.4. In both cases we used blocks of 512 samples for the temporal duration, which were sub-sampled to test the algorithm. We can see a saturation effect on the quality of the reconstruction, with the increase of K. The exact choice of this parameter is application dependent and depends on the dimension of the network and the number of sensors. In our case K was set to 24 in the CIMIS and 12 in the 3ENCULT case.

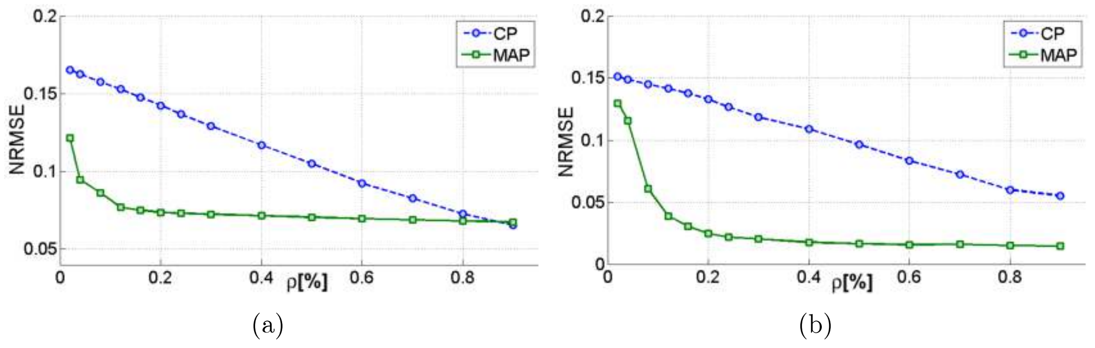


Figure 6.5: Node duty cycling effects on the reconstruction error as computed from the standard tensor factorization approach (CP) and our technique (MAP): (a) CIMIS dataset and (b) 3ENCULT.

6.6.5 Data recovery

In order to save energy, sensor nodes can apply aggressive duty cycling strategies, collecting just a portion of the original data. When applying a duty cycling policy, a node avoids sampling its sensors, resulting in an entire rows of missing entries in the data tensor. To analyze the ability of the proposed method to reconstruct the missing samples, we removed increasing percentages of rows from the collected data. The remaining data (*training data*) was used to learn our latent variable model, while the removed entries (*testing data*) were used to evaluate the recovering capabilities of the algorithm.

Figure 6.5 shows the reconstruction error of the missing data ($NRMSE$) as a function of the fraction of the sampled data (ρ) for the two analyzed datasets. We compare the two implementations of the tensor factorization algorithm: (1) the standard factorization technique (CP) that does not include correlations from the data, and (2) our approach that does incorporate the temporal, spatial, and inter-sensor correlation, where the variables are learned using the MAP estimation. In both cases we use a block of 512 samples from the whole network and set the dimension of the latent variables as indicated in the previous section. The mean $NRMSE$ across all the variables is reported on the *testing data* for the two datasets. For the CIMIS case the results show that our approach achieves a reconstruction error below 7%, with up to 80% of the missing samples. With the

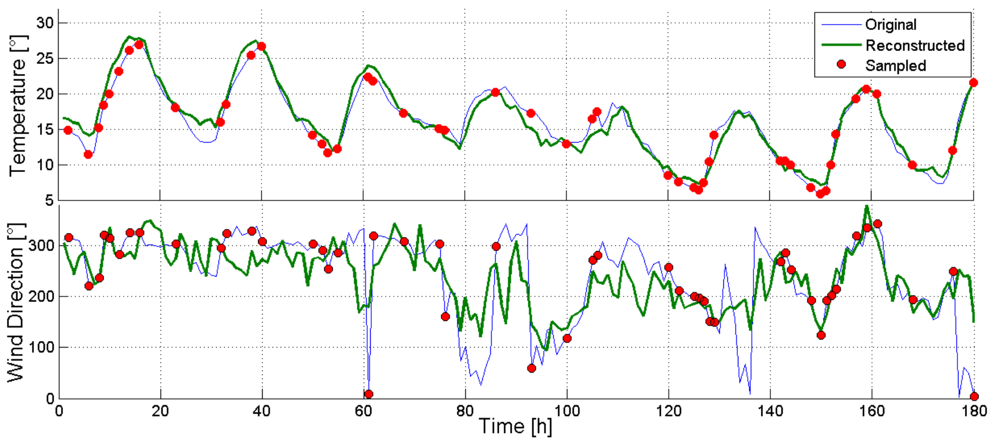


Figure 6.6: Example of reconstructed data with 80% of missing samples for temperature (above) and wind direction (below) from the CIMIS dataset.

same sampling policy for the 3ENCULT dataset we achieve a mean reconstruction accuracy of 2.5%. Our technique consistently outperforms the standard tensor factorization approach.

The results outlined above are the average reconstruction errors among all sensors. If we look at the individual sensors, we see that the single variables exhibit different recovering proprieties. The ones characterized by strong correlation proprieties and smooth temporal transitions, such as temperature and solar radiation, are better reconstructed compared to the ones that have high variance or little periodicity. For example, in Figure 6.6 we present an example of reconstructed data for two variables of the CIMIS dataset: one with smooth and periodical behavior (temperature, with a 2% error) and the other with high variance (wind direction, with a 23% error). For both examples we used 20% of the original data to learn the latent variable models. This is the cause of the higher mean error for the CIMIS dataset, since the 3ENCULT one does not incorporate variables difficult to reconstruct as the wind ones.

6.6.6 Energy saving

Using the energy characterizations of the nodes from Tables 6.4 and 6.5, we can estimate the average amount of energy savings associated with different sampling

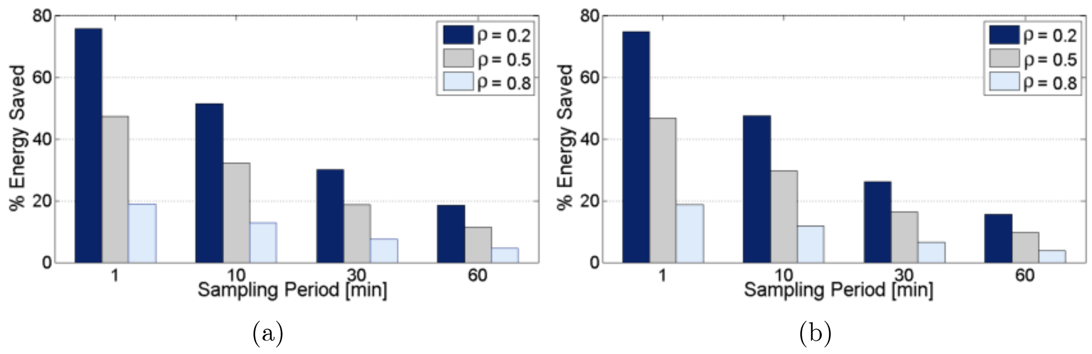


Figure 6.7: Energy saving (%) when applying a give sampling policy for different sampling periods: (a) CIMIS dataset and (b) 3ENCULT.

policies. Figure 6.7 shows energy savings for different combinations of sampling periods and duty cycling rates in the two case studies. Different sampling policies are compared to the case when all the data is sampled. Even with the different hardware characteristics and the energy consumption profiles of the two scenarios, we have a similar result, showing that our algorithm can get large energy savings in a wide range of applications. Aggressively duty cycling on a dataset that samples the environment every minute yields significantly higher energy savings (76%) than the one which only samples every hour (20%). This is expected because with lower sampling frequencies, the sensor nodes spend most of the time in a low power sleep state. Thus, the energy consumption within the sleep state dominates and the overall power consumption is less influenced by the sampling policy.

6.6.7 Comparison with compressive sensing

In this section we compare the reconstruction performance of CS and the latent variable (LV) based statistical model against the 3ENCULT deployment, considering data coming from temperature, humidity and light sensors. We want to investigate whether a better reconstruction technique does exist among those here proposed and how the sub-sampling parameters affect such recovery quality.

The compression phase is the same for both frameworks: each node samples

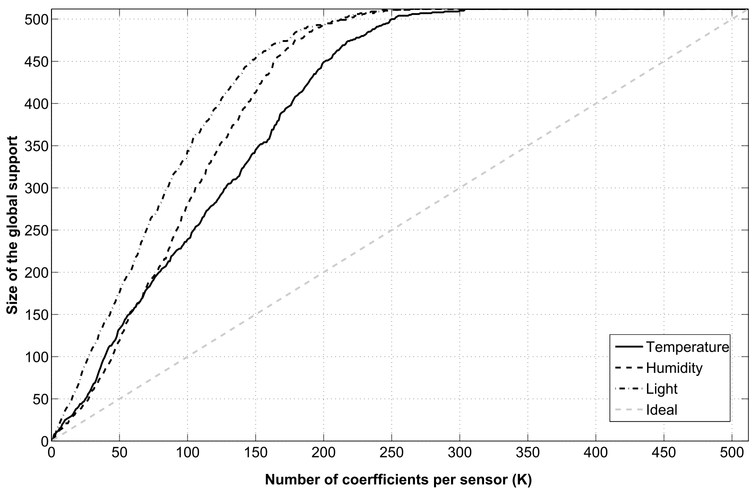


Figure 6.8: Number of DCT coefficients necessary to include the K largest coefficients for each signal. ($N = 512$)

the signals of interest gathering a sub-set M of the needed samples ($M = \rho N$), with $0 < \rho < 1$. After the acquisition period $T = NT_k$ the gathered data is sent to the collecting sink through the network. The sampling time T_k in the following simulations is set to $600s$ and the results are averaged over 100 trials. Each trial is characterized by a different sampling pattern and a different considered portion of the signal. The reconstruction phase is fairly different and determines the recovery quality of the original signal. For CS the DCT matrix is used as sparsifying matrix that is already been demonstrated being a good sparsifying matrix for natural signals [41].

In the first simulation we reconstruct the original signals from a sub-sampled version without exploiting any inter- or intra-correlation among them, just averaging the reconstruction quality over all the signals, with a signal length of $N = 512$. For the latent variables approach here we used the standard CP tensor factorization technique, without the contribution of any correlations in the data. The comparison is carried evaluating the signal to noise ratio (SNR) defined as:

$$\text{SNR}_{\text{dB}} = 20 \cdot \log_{10} \frac{\|\mathbf{x}\|_2}{\|\mathbf{x} - \hat{\mathbf{x}}\|_2} \tag{6.11}$$

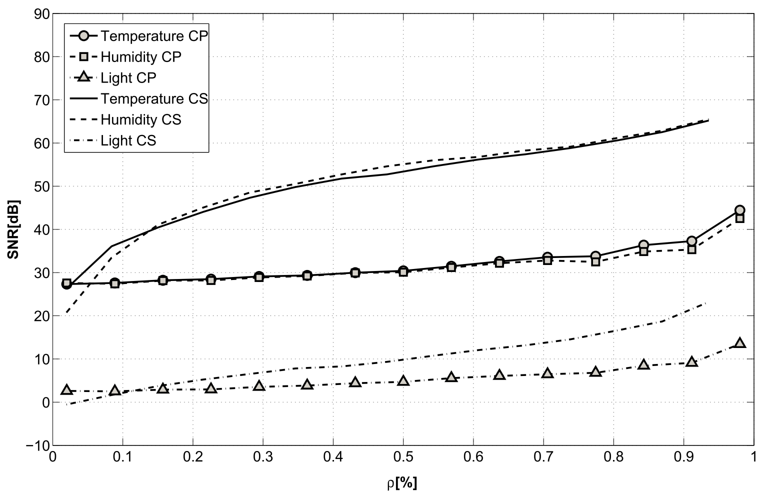


Figure 6.9: Recovery comparison between CS and latent variable (LV) method when reconstructing the original signals from sub-sampled version averaging the reconstruction quality over all the nodes ($N = 512$)

where $\mathbf{x}$ is the original signal and $\hat{\mathbf{x}}$ is its recovered version. We show the average SNR across all the network nodes.

From the plot, shown in Fig. 6.9, we can infer how different the performance is for the two techniques: while the reconstruction performance for LV is pretty stable varying the sub-sampling factor ρ, CS is much more affected by the compression factor. Recovery with CS achieves a better reconstruction almost for every sub-sampling factor with respect to the latent variable based technique.

Both the frameworks seem to be greatly affected by the nature of the signal to reconstruct. In particular from the plot we can infer how the recovery of light signals is difficult for the two proposed techniques. This is due to the nature of the light signal that is recorded inside the building. While for temperature and humidity the gathered signals are continuous signals and smoothly affected by the human presence, the light signal is highly irregular and highly influenced by the artificial lighting in the single rooms. Moreover, some of the nodes are placed in the basement where the light level is under the noise threshold of the light sensors, providing extremely noisy data.

To evaluate whether it is possible to exploit the correlations existing among sensors and nodes to improve the reconstruction, we performed the recovery

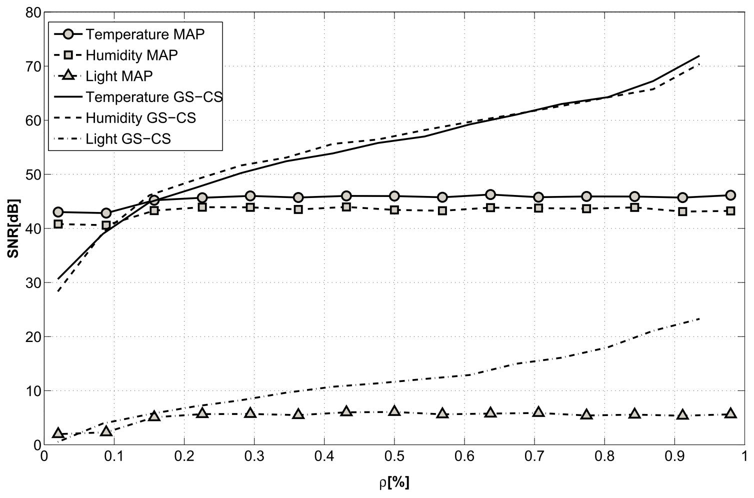


Figure 6.10: Recovery comparison between GC-CS and MAP. The recovery is obtained exploiting the correlations among sensors and nodes. $(N = 512)$

against the same dataset using for CS the group sparse optimization (GS-CS) exploiting the joint sparsity of the signals and for LV the maximum a posteriori optimization (LV-MAP) introduced in Eq. 6.9. Results are reported in Fig. 6.10. While the performance for CS remains almost the same, the LV-MAP method guarantees a significant improvement in reconstruction resulting better than CS for small values of ρ, especially in relation to humidity and temperature signals.

The behavior of CS could be explained looking at Fig. 6.8. Here we show how the union over all signals of the K best DCT basis vectors per signal has a size definitely greater than K. Practically this means that GS-CS is able to exploit the inter-nodes correlation only at a small extent since the shared information among different nodes is limited and the recovery algorithm is not able to exploit this information to improve the recovery quality.

According to the model in Section 6.5.2 the simulations are performed with a sampling frequency $f_s = 1/600$[Hz], and since the length of the data is $N = 512$ this brings in a delay in the data delivery towards the data collector of 3.5 days. Thus the size of the recovered signal spans across 3.5 days. In practice having high values of N means that we have to wait a longer time to proceed with data recovery. Therefore we want now to investigate how the length of the block of data gathered by sensors affects the two frameworks and whether a correlation

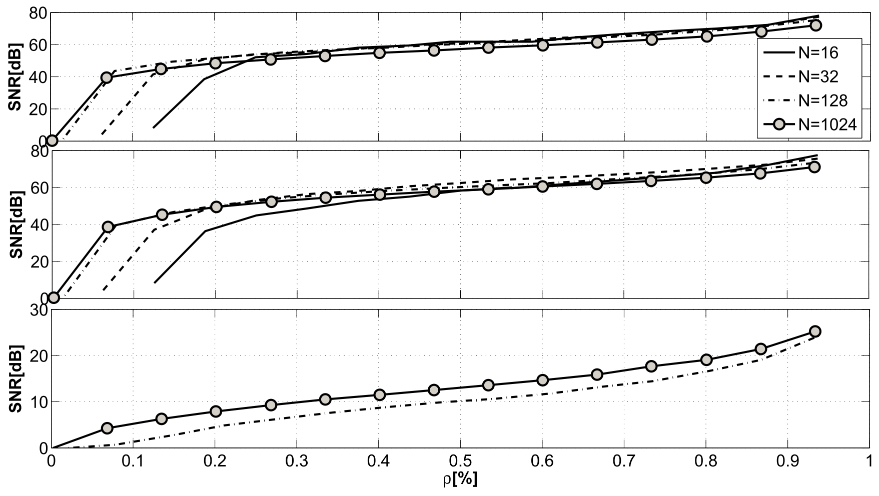


Figure 6.11: From top to bottom: temperature, humidity, light. GS-CS reconstruction quality for the different signals varying the sub-sampling factor ρ and using the signal length N as parameter.)

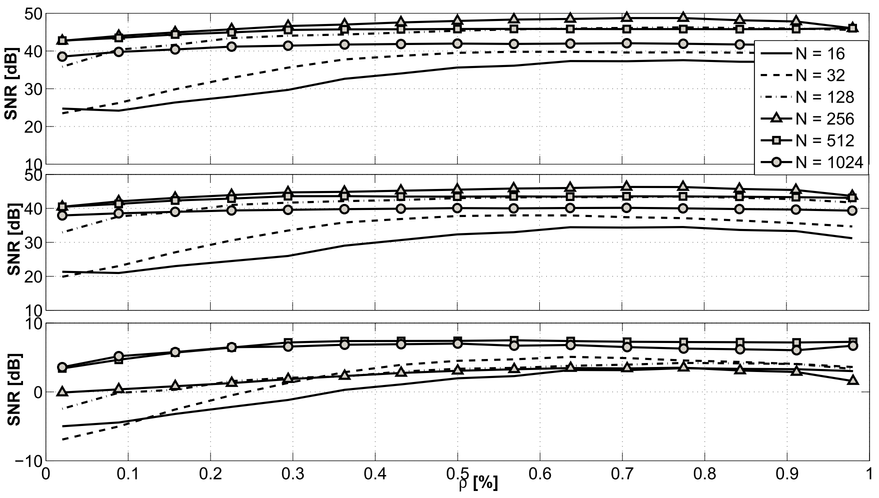
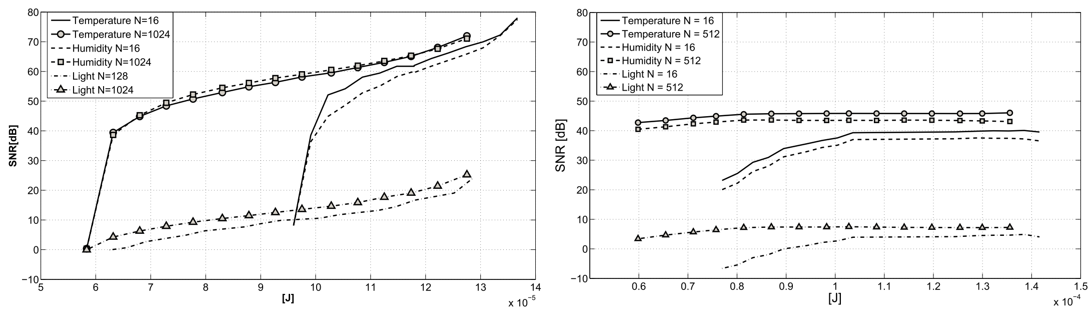


Figure 6.12: From top to bottom: temperature, humidity, light. LV-MAP reconstruction quality for the different signals varying the sub-sampling factor ρ and using the signal length N as parameter.)



(a) (b)

Figure 6.13: Reconstruction quality vs. averaged per cycle energy consumption varying the parameter N for the signals of interest for the considered techniques: (a) CS and (a) LV.

between recovery performance and the N parameter does exist for GS-CS and LV-MAP.

In Fig. 6.11 the results for GS-CS when N is changed are presented. From the plot we can infer how the length of the signal N does not greatly affect the reconstruction quality for all the signals taken into consideration. Rather we can see how the influence of the parameter N (and then of the delay in the data collection) is only visible for small values of the sub-sampling factor ρ. Differently from temperature and humidity, the light signal presents a peculiar behavior showing an increased reconstruction quality with the increasing in the number of acquired samples.

The same results for the LV-MAP approach are presented in Fig. 6.12. The difference in the reconstruction error for the various values of N is more evident than in the GS-CS case. With small values of N we registered difficulties to reconstruct the desired signals. The best recovery performance is achieved when considering 256-512 samples at a time, identifying the optimal trade-off between delay and reconstruction accuracy, since larger blocks of data present again a loss of accuracy.

For a delay smaller than $N = 128$ the reconstruction of the light signal is not feasible in both cases, since the majority of the samples gathered are zeros due to the lack of light at night.

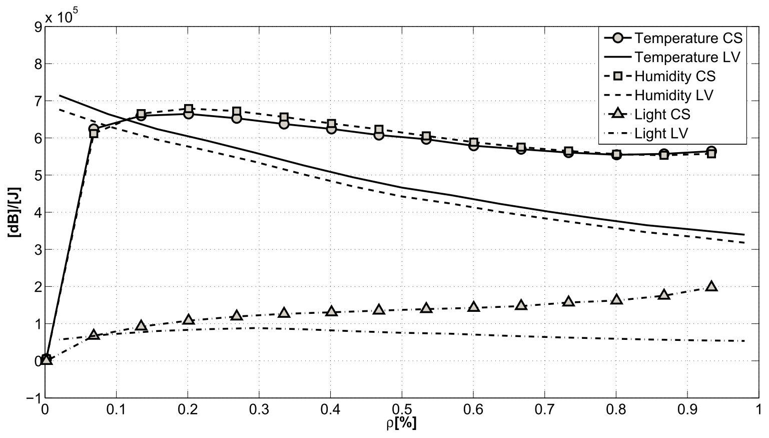


Figure 6.14: Ratio between the recovery quality and energy spent in compression varying the sub-sampling factor ρ for the two approaches

Having evaluated the recovery performance and the influence of the gathering delay on reconstruction, it is interesting to investigate the power consumption involved with compression according to the power model in section 6.5.3. In Fig. 6.13 we report the reconstruction quality against the energy consumption for one acquisition cycle. The plot clearly shows how a trade-off between energy consumption and transmission delay does exist in GS-CS case (Fig.6.13(a)). Higher values of N, thus higher delays in transmission, are able to guarantee a better reconstruction quality with definitely less energy than the $N = 16$ case (all the other cases are not considered in the plot since they are between these two boundaries). The light signal is a special case but we can draw the same conclusions as before. The LV-MAP case (Fig.6.13(b)) presents a similar behavior, but with a less accentuated increase in energy efficiency corresponding to the increase in data size N. When comparing the two graphs, we can observe how the GS-CS case exhibits a slightly higher energy efficiency, allowing a higher reconstruction quality when considering the same energy consumptions as the LV-MAP case. Only for extremely sub-sampled signals the LV-MAP approach results better, having a major benefit from the explicit correlation models incorporated in the data reconstruction. In both cases, we are able to obtain a better accuracy (or the same reconstruction quality with less energy) if we are willing to wait for an higher number of gathered samples before proceeding with the reconstruction.

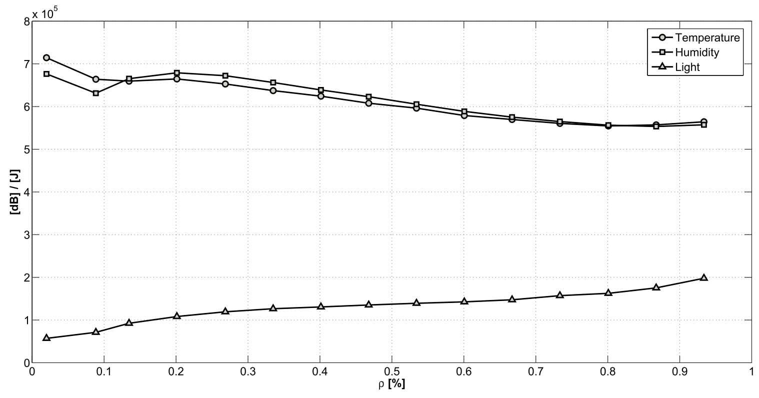


Figure 6.15: Combination of the two approaches, showing the best achievable ratio between accuracy and energy consumption for every sub-sampling factor ρ.

The same conclusions are explicated in Fig. 6.14 where the ratio between the reconstruction quality and the consumed energy is plotted against the sub-sampling factor ρ. Here we can see a direct comparison of the two techniques for the case with the best reconstruction performance ($N = 512$). Again we can see how the GS-CS case has a higher ratio when compared to the LV-MAP case, for almost all the sub-sampling policies. Only when dealing with a really small amount of sampled data (20%), the LV-MAP case shows a better performance. This result can be a guide for WSN developers, suggesting the adoption of the LV method only when a really aggressive power saving technique is needed. The plots of the two techniques are combined in Fig. 6.15, where the best ratio between the reconstruction accuracy and the consumed energy is plotted against the sub-sampling factor. Here we have the combination of the two approaches and show the best achievable results for every sub-sampling factor. This graph can be used as a design specification to choose the best technique depending on the application.

6.7 Conclusions

In this Chapter we presented and compared two promising techniques for energy efficient data gathering and reconstruction in wireless sensor networks. We introduced a latent variable model for energy efficient data collection and extended the approach of the well known compressive sensing framework. Both approaches try to successfully recover the desired signal from a highly incomplete sub-sampled version, obtained opportunely skipping samples during the acquisition phase. The two techniques exploit redundancies and correlations present in the gathered data to achieve a better reconstruction accuracy with a smaller number of collected samples and thus with a lower energy consumption.

Our approach allows us to employ aggressive duty cycling strategies on the sensor nodes for better energy conservation. Each node is able to save energy by sampling just a portion of the desired data. The available data is then used to reconstruct the missing samples. The proposed techniques do not require any kind of data processing on the sensor node, which only adopts the desired duty cycling policy and sends the collected samples.

We introduced an energy model for the sensor nodes and analyzed data from a real sensor network deployment. Experimental results show that the proposed latent variable statistical approach can maintain a low mean reconstruction error, below 7%, with up to 80% of missing samples. This permits energy savings ranging from 20% to 80%.

The comparison between the two techniques showed how the use of the data from the whole network and its correlations improved the recovery performance in both cases, when compared to the standard approaches where individual nodes and signals are considered. The CS approach usually achieves better reconstruction accuracy, with the exception of cases when really aggressive sub-sampling policies are used. This leads also to a better energy efficiency of the CS method.

Chapter 7

Conclusions

The vision of Ambient Intelligence, developed more than a decade ago, envisions a future of the information society where small, unobtrusive and ubiquitous electronic devices will augment the environment, making it sensitive and responsive to the presence of people and their activities. A decade after its definition, the AmI vision is still not as a widespread concept as it was intended and has not penetrated our lives according to the expectations. The vision, instead, contributed to a decade of scientific and technological progress, and to a wider debate on the present and the future of our society. There are still formidable challenges to be tackled by the scientific community and there is no guarantee we will solve all of them, but the more progress we make on some, the closer we will be to a stage where some of the initial aims are adopted.

In this context, the work introduced in this dissertation focused on the hardware and software design of embedded systems and sensor networks. Our approach has always the user as the main focus of the work: the digital ecosystem pervading the environment should take advantage of the technological progress to adapt to the needs and the desires of the user.

From the hardware perspective, we developed different embedded devices, employing state of the art solutions with advanced sensing and processing capabilities. The hardware of the devices were accompanied by the development of innovative algorithms to efficiently process the sensor data and extract useful information. The goal of our approach was to optimize the tradeoff between the accuracy and the power consumptions, since energy efficiency is one of the key features for embedded devices, especially when battery-powered.

The first device introduced was the *SMCube*, a smart object equipped with an accelerometer and used as a tangible interface. The ability to process on-board the sensor information and enable gestural recognition capabilities is an advantage for the device, improving its battery lifetime, the overall system scalability and the handling of multiple or moving devices. This dissertation presented and characterized an implementation of the Hidden Markov Model (HMM) forward algorithm suitable for the class of low-power, low-cost MCU embedded into the *SMCube*. HMMs are state of the art algorithms for gesture and speech recognition, and enable the adoption of natural user interaction paradigms. The characterization of our algorithm in both single and multiuser scenarios demonstrates the effectiveness of the approach and that the use of fixed point data representation results in recognition ratios comparable to the floating point case when using more than 16 bits.

Following the *SMCube* experience, we developed a wireless low-cost pen-like device for a fast and interactive reverse engineering framework named FIRES. The system enables real-time acquisition and manipulation of complex geometrical shapes through the *SmartPen*, the interactive input device. The FIRES framework utilizes the *SmartPen* as the primary interaction tool to support the reconstruction and editing of virtual 3D models of objects in a CAD application. The device embeds an Inertial Measurement Unit (IMU) and its sensor data is fused with the output of a low-cost stereo computer vision system, in order to accurately track the position and orientation of the device. Beside being a versatile free-hand interaction tool, the device can be used to draw the style-lines of the object to be modeled, which are then used to reconstruct its surface in the virtual environment.

Continuing the evolution of hardware and software solutions, we improved the design of the *SmartPen* to develop a network of miniaturized wearable IMUs for the analysis of human motion. The *EXLs1* sensor node embeds state of the art integrated inertial (accelerometer and gyroscope) and magnetic sensors, features an advanced ARM microcontroller and a Bluetooth transceiver. It was designed to be small and versatile for an easy use in a broad range of applications, such can be human-computer interaction, motion capture or heathcare. The availability of the Bluetooth connection enables the device to communicate with a wide range of systems and allows the use of multiple nodes in a body sensor network. We analyzed several orientation estimation techniques, employing sensor fusion algorithms to process the sampled sensor data and compute useful features. The proposed estimation techniques were implemented on the microcontroller embedded in the device, and tested comparing their performance in terms of accuracy and orientation estimation. To achieve an efficient use of the limited energy provided by the battery integrated into the device, we carefully optimized the use of the available peripherals, taking advantage of power saving techniques.

The exploration of Wireless Sensor Networks (WSNs) and innovative technologies for their enhancement, from the data accuracy and power consumption viewpoints, is a natural evolution of the work presented in the first part of this dissertation. Starting from single smart objects we explored body area networks, with a limited number of sensor nodes placed on the human body, and finally we considered the case of WSNs composed by tens or thousands of nodes that sense the surrounding environment. The goal or our work was to improve the performance/power consumption tradeoff: in a general WSN scenario this can be achieved by minimizing storage and communication costs to extend as much as possible the lifetime of the network, while maintaining the desired data accuracy.

In the last Chapter of this dissertation two promising techniques for energy efficient data gathering and reconstruction in WSNs were presented. We introduced a latent variable model for energy efficient data collection and extended the approach of the well known compressive sensing framework. The two techniques exploit redundancies and correlations present in the gathered data to achieve a better reconstruction accuracy with a smaller number of collected samples and thus with a lower energy consumption. The comparison between the two techniques showed how the use of the data from the whole network and its correlations improved the recovery performance in both cases, when compared to the standard approaches where individual nodes and signals are considered. The proposed CS approach usually achieves better reconstruction accuracy, with the exception of cases when really aggressive sub-sampling policies are used.

Bibliography

[1] Atmel ATmega8. http://www.atmel.com/devices/atmega8.aspx. [Online; accessed Jan-2012]. 40

[2] HMM Toolbox for Matlab. http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html. [Online; accessed Jan-2012]. 48

[3] Microsoft Kinect. http://www.microsoft.com/en-us/kinectforwindows/. [Online; accessed Jan-2012]. 61

[4] STM LIS3LV02 Acceleormeter. http://www.st.com/web/catalog/sense_power/FM89/SC444/PF127514. [Online; accessed Jan-2012]. 41

[5] Bluegiga WT12 Bluetooth trasciever. http://www.bluegiga.com/WT12_Class_2_Bluetooth_Module. [Online; accessed Jan-2012]. 40

[6] Anoto. www.anoto.com. [Online; accessed Dec-2012]. 61

[7] Datasheet bh1715. http://www.alldatasheet.com/view.jsp?Searchword=BH1715. Accessed: 10/12/2012. 136

[8] cvBlob Library. https://code.google.com/p/cvblob/. [Online; accessed Dec-2012]. 72

[9] Earth's Magnetic Field Calculator. http://www.ngdc.noaa.gov/geomag/magfield.shtml. [Online; accessed Dec-2012]. 94

[10] Manoscribe 3D Digitalizer. www.microscribe.ghost3d.com. [Online; accessed Dec-2012]. 60

[11] Minoru 3D. http://www.minoru3d.com/. [Online; accessed Dec-2012]. 59, 69

[12] OTM Techologies. www.otmtech.com/. [Online; accessed Dec-2012]. 61

[13] Sensable PHANTOM. www.sensable.com/industries-application-development.htm. [Online; accessed Dec-2012]. 60

[14] Datasheet sht21. http://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/Humidity/Sensirion_Humidity_SHT21_Datasheet_V3.pdf. Accessed: 10/12/2012. 136

[15] Stm32w108cb datasheet. http://www.st.com/internet/mcu/product/245381.jsp. Accessed: 10/12/2012. 136

[16] 3ENCULT. http://www.3encult.eu/en/casestudies/default.html, 2012. 135

[17] E. Acar and B. Yener. Unsupervised multiway data analysis: A literature survey. *IEEE Transactions on Knowledge and Data Engineering*, 21(1): 6–20, 2009. 130, 132

[18] Y. Agarwal, B. Balaji, R. Gupta, J. Lyles, M. Wei, and T. Weng. Occupancy-driven energy management for smart building automation. In *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, BuildSys '10, pages 1–6, 2010. 121

[19] A. Ahmadi, D.D. Rowlands, and D.A. James. Investigating the translational and rotational motion of the swing using accelerometers for athlete skill assessment. In *Sensors, 2006. 5th IEEE Conference on*, pages 980–983. IEEE, 2006. 85

[20] Oliver Amft, Holger Junker, and Gerhard Troster. Detection of eating and drinking arm gestures using inertial body-worn sensors. In *Proceedings of the Ninth IEEE International Symposium on Wearable Computers*, ISWC '05, pages 160–163, 2005. 33

[21] Roman Amstutz, Oliver Amft, Brian French, Asim Smailagic, Dan Siewiorek, and Gerhard Troster. Performance analysis of an hmm-based gesture recognition using a wristwatch device. In *Proceedings of the 2009 International Conference on Computational Science and Engineering - Volume 02*, CSE '09, pages 303–309, 2009. 28, 33

[22] Eric R Bachmann. Inertial and magnetic tracking of limb segment orientation for inserting humans into synthetic environments. Technical report, DTIC Document, 2000. 100, 101

[23] P.K. Baheti and H. Garudadri. An ultra low power pulse oximeter sensor based on compressed sensing. In *Wearable and Implantable Body Sensor Networks, 2009. BSN 2009. Sixth International Workshop on*, pages 144 –148, june 2009. 126

[24] Gonzalo Bailador, Daniel Roggen, Gerhard Tröster, and Gracián Triviño. Real time gesture recognition using continuous time recurrent neural networks. In *Proceedings of the ICST 2nd international conference on Body area networks*, BodyNets '07, pages 15:1–15:8, 2007. 32

[25] M. Balouchestani. Low-power wireless sensor network with compressed sensing theory. In *Fly by Wireless Workshop (FBW), 2011 4th Annual Caneus*, pages 1 –4, june 2011. doi: 10.1109/FBW.2011.5965565. 122

[26] Stefano Baraldi, Alberto Del Bimbo, Lea Landucci, and Alessandro Valli. wikitable: finger driven interaction for collaborative knowledge-building workspaces. In *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop*, CVPRW '06, 2006. 30

[27] Stefano Baraldi, Alberto Del Bimbo, Lea Landucci, Nicola Torpei, Omar Cafini, Elisabetta Farella, Augusto Pieracci, and Luca Benini. Introducing tangerine: a tangible interactive natural environment. In *Proceedings of the 15th international conference on Multimedia*, MULTIMEDIA '07, pages 831–834, 2007. 27, 42

[28] M. Barczyk, M. Jost, D.R. Kastelan, A.F. Lynch, and K.D. Listmann. An experimental validation of magnetometer integration into a gps-aided

helicopter uav navigation system. In *American Control Conference (ACC)*, pages 4439–4444, 2010. 93

[29] Dror Baron, Marco F. Duarte, Michael B. Wakin, Shriram Sarvotham, and Richard G. Baraniuk. Distributed compressive sensing. In *Sensor, Signal and Information Processing (SenSIP) Workshop*, 2008. 124

[30] G. Barrenetxea, F. Ingelrest, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange. Sensorscope: Out-of-the-box environmental monitoring. In *International Conference on Information Processing in Sensor Networks*, pages 332–343, 2008. 121

[31] Roger Bartlett. *Introduction to sports biomechanics: Analysing human movement patterns*. Routledge, 2007. 85

[32] C. Beccari, G. Casciola, and L. Romani. Interpolatory surface subdivision based on geometry-driven parameterizations. In *Minisymposium on New Trends in Numerical Approximation and Applications*, SIMAI08, 2008. 77

[33] C.V. Beccari, E. Farella, A. Liverani, S. Morigi, and M. Rucci. A fast interactive reverse-engineering system. *Computer-Aided Design*, 42(10):860 – 873, 2010. 59, 68

[34] J. Bohn, V. Coroama, M. Langheinrich, F. Mattern, and M. Rohs. Social, Economic, and Ethical Implications of Ambient Intelligence and Ubiquitous Computing. In W. Weber, J. Rabaey, and E. Aarts, editors, *Ambient Intelligence*. Springer, April 2005. 11

[35] Doug A. Bowman, Sabine Coquillart, Bernd Froehlich, Michitaka Hirose, Yoshifumi Kitamura, Kiyoshi Kiyokawa, and Wolfgang Stuerzlinger. 3D User Interfaces: New Directions and Perspectives. *IEEE Comput. Graph. Appl.*, 28(6):20–36, November 2008. URL `http://dx.doi.org/10.1109/MCG.2008.109`. 60

[36] D. Boyle, M. Magno, B. O'Flynn, D. Brunelli, E. Popovici, and L. Benini. Towards persistent structural health monitoring through sustainable wireless sensor networks. In *Intelligent Sensors, Sensor Networks and Informa-*

tion Processing (ISSNIP), 2011 Seventh International Conference on, pages 323 –328, dec. 2011. 121

[37] Carmen MN Brigante, Nunzio Abbate, Adriano Basile, Alessandro Carmelo Faulisi, and Salvatore Sessa. Towards miniaturization of a mems-based wearable motion capture system. *Industrial Electronics, IEEE Transactions on*, 58(8):3234–3241, 2011. 87

[38] Miguel Bruns Alonso and V. Keyson. Musiccube: a physical experience with digital music. *Personal Ubiquitous Comput.*, 10(2-3). 31, 32

[39] C. Buratti, R. D'Errico, M. Maman, F. Martelli, R. Rosini, and R. Verdone. Design of a body area network for medical applications: the wiserban project. In *Proceedings of the 4th International Symposium on Applied Sciences in Biomedical and Communication Technologies*, page 164. ACM, 2011. 85

[40] Omar Cafini, Piero Zappi, Elisabetta Farella, Luca Benini, Stefano Baraldi, Nicola Torpei, Lea Landucci, and Alberto Del Bimbo. Tangerine smcube: a smart device for human computer interaction. *Proceedings of 3rd IEEE European Conference on Smart Sensing and Context (EuroSSC)*, October 2008. 27, 28

[41] C. Caione, D. Brunelli, and L. Benini. Compressive sensing optimization over zigbee networks. In *Industrial Embedded Systems (SIES), 2010 International Symposium on*, pages 36 –44, july 2010. doi: 10.1109/SIES.2010.5551380. 148

[42] C. Caione, D. Brunelli, and L. Benini. Distributed compressive sampling for lifetime optimization in dense wireless sensor networks. *IEEE Transactions on Industrial Informatics*, 8(1):30–40, 2012. 122

[43] B.H. Calhoun, D.C. Daly, Naveen Verma, D.F. Finchelstein, D.D. Wentzloff, A. Wang, Seong-Hwan Cho, and A.P. Chandrakasan. Design considerations for ultra-low energy wireless microsensor nodes. *IEEE Transactions on Computers*, 54(6):727–40, 2005. 121

[44] James Calusdian, Xiaoping Yun, and Eric Bachmann. Adaptive-gain complementary filter of inertial and magnetic data for orientation estimation. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1916–1922. IEEE, 2011. 100

[45] E.J. Candes and M.B. Wakin. An introduction to compressive sampling. *Signal Processing Magazine, IEEE*, 25(2):21 –30, march 2008. ISSN 1053-5888. doi: 10.1109/MSP.2007.914731. 122, 124

[46] E.J. Candes and M.B. Wakin. An introduction to compressive sampling. *Signal Processing Magazine, IEEE*, 25(2):21 –30, march 2008. 122

[47] D. Carli, D. Brunelli, L. Benini, and M. Ruggeri. An effective multi-source energy harvester for low power applications. In *Design, Automation Test in Europe Conference Exhibition*, pages 1–6, 2011. 121

[48] E. Catmull and J. Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10(6):350 – 355, 1978. 76

[49] Graeme S. Chambers, Svetha Venkatesh, Geoff A. W. West, and Hung H. Bui. Segmentation of intentional human gestures for sports video annotation. In *Proceedings of the 10th International Multimedia Modelling Conference*, MMM '04, pages 124–, 2004. 33

[50] Wei Chen and I.J. Wassell. Energy efficient signal acquisition via compressive sensing in wireless sensor networks. In *Wireless and Pervasive Computing (ISWPC), 2011 6th International Symposium on*, pages 1 –6, feb. 2011. 126

[51] Wei Chen, M.R.D. Rodrigues, and I.J. Wassell. Distributed compressive sensing reconstruction via common support discovery. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1 –5, june 2011. 128

[52] Sung-Do Choi, A.S. Lee, and Soo-Young Lee. On-line handwritten character recognition with 3d accelerometer. In *Information Acquisition, 2006 IEEE International Conference on*, pages 845 –850, aug. 2006. 61

[53] Suan Khai Chong, Mohamed Medhat Gaber, Shonali Krishnaswamy, and Seng Wai Loke. Transactions on large-scale data and knowledge-centered systems. 2011. 121

[54] CIMIS dataset. http://www.cimis.water.ca.gov, March 2012. 134

[55] Microsoft Corporation. Microsoft surface. URL http://www.microsoft.com/en-us/pixelsense/. 30

[56] Nintendo Corportaion. Wii. URL http://www.nintendo.com/wii. 31

[57] CuPiD EU Project. http://www.cupid-project.eu. 13

[58] K. Das, P. Diaz-Gutierrez, and M. Gopi. Sketching free-form surfaces using network of curves. In *Proceedings of eurographics workshop on sketch-based interfaces and modeling (SBIM'05)*, 2005. 60

[59] W. Deng, W. Yin, and Y. Zhang. Group sparse optimization by alternating direction method. In *Rice CAAM Report TR11-06*, 2011. 124

[60] Digi. Xbee pro zb. http://www.digi.com/xbee/, March 2012. 138

[61] C Doukas and I Maglogiannis. Advanced patient or elder fall detection based on movement and sound data, 2008. 84

[62] M.F. Duarte and R.G. Baraniuk. Kronecker compressive sensing. *Image Processing, IEEE Transactions on*, 21(2):494 –504, feb. 2012. ISSN 1057-7149. doi: 10.1109/TIP.2011.2165289. 124

[63] M.F. Duarte and Y.C. Eldar. Structured compressed sensing: From theory to applications. *Signal Processing, IEEE Transactions on*, 59(9):4053–4085, sept. 2011. 122, 124

[64] K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten, and J.-C. Burgelman. Scenarios for ambient intelligence in 2010, 2001. URL www.cordis.lu/ist/istag. 2, 9

[65] eSense EU Project. http://www.ist-esense.org/. 12

[66] E. Farella, M. Falavigna, and B. Ricco. Aware and smart environments: The casattenta project. In *Advances in sensors and Interfaces, 2009. IWASI 2009. 3rd International Workshop on*, pages 2 –6, june 2009. 121

[67] Elisabetta Farella, Augusto Pieracci, Davide Brunelli, Luca Benini, Bruno Ricco, and Andrea Acquaviva. Design and implementation of wimoca node for a body area wireless sensor network. In *Proceedings of the 2005 Systems Communications*, ICW '05, pages 342–347, 2005. 30

[68] Elisabetta Farella, Mirko Falavigna, and Bruno Riccò. Aware and smart environments: The casattenta project. *Microelectron. J.*, 41(11):697–702, November 2010. ISSN 0026-2692. 12

[69] Jay Farrell. *Aided navigation: GPS with high rate sensors*. McGraw-Hill New York, NY, USA:, 2008. 85, 90, 93, 95, 97, 100

[70] C.C. Foster and G.H. Elkaim. Extension of a two-step calibration methodology to include nonorthogonal sensor axes. *IEEE Trans. on Aerospace and Electronic Systems*, 44(3). 92

[71] Michael Friedewald, Olivier Da Costa, Yves Punie, Petteri Alahuhta, and Sirkka Heinonen. Perspectives of ambient intelligence in the home environment. *Telemat. Inf.*, 22(3):221–238, August 2005. 12

[72] Bernd Frohlich, Jan Hochstrate, Alexander Kulik, and Anke Huckauf. On 3D Input Devices. *IEEE Comput. Graph. Appl.*, 26(2):15–19, March 2006. URL `http://dx.doi.org/10.1109/MCG.2006.45`. 60

[73] H. Garudadri, P.K. Baheti, S. Majumdar, C. Lauer, F. Masse and, J. van de Molengraft, and J. Penders. Artifacts mitigation in ambulatory ecg telemetry. In *e-Health Networking Applications and Services (Healthcom), 2010 12th IEEE International Conference on*, pages 338 –344, july 2010. 126

[74] Siome Klein Goldenstein. A gentle introduction to predictive filters. *Revista de Informatica Teorica e Aplicada (RITA) XI*, 1:61–89, 2004. 64

[75] Mohinder S Grewal and Angus P Andrews. *Kalman filtering: theory and practice using MATLAB*. Wiley-IEEE press, 2011. 85

[76] D L Hall and J Llinas. An introduction to multisensor data fusion. *Proceedings of the IEEE*, 85(1):6–23, 1997. 62

[77] Holger Harms, Oliver Amft, Rene Winkler, Johannes Schumm, Martin Kusserow, and Gerhard Troester. Ethos: Miniature orientation sensor for wearable human motion analysis. In *Sensors, 2010 IEEE*, pages 1037–1042. IEEE, 2010. 86

[78] Jin He, Huaming Li, and Jindong Tan. Real-time daily activity classification with wireless sensor networks using hidden markov model. In *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, pages 3192–3195. IEEE, 2007. 85

[79] Michael Healy, Thomas Newe, and Elfed Lewis. Wireless sensor node hardware: A review. In *Sensors, 2008 IEEE*, pages 621–624. IEEE, 2008. 85

[80] Seongkook Heo, Jaehyun Han, Sangwon Choi, Seunghwan Lee, Geehyuk Lee, Hyong-Euk Lee, SangHyun Kim, Won-Chul Bang, DoKyoon Kim, and ChangYeong Kim. Ircube tracker: an optical 6-dof tracker based on led directivity. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 577–586, 2011. 61

[81] Frank G. Hofmann, Peter Heyer, and Günter Hommel. Velocity profile based recognition of dynamic gestures with discrete hidden markov models. In *Proceedings of the International Gesture Workshop on Gesture and Sign Language in Human-Computer Interaction*, pages 81–95, 1998. 33

[82] Chun-Rong Huang, Chu-Song Chen, and Pau-Choo Chung. Tangible photorealistic virtual museum. *IEEE Comput. Graph. Appl.*, 25(1):15–17, January 2005. 30

[83] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: a sketching interface for 3D freeform design. In *International Conference on Computer Graphics and Interactive Techniques*, 2007. 60

[84] iRoom Project. `http://perso.limsi.fr/Individu/bellik/yacine/doku.php?id=iroom:en:home`. 12

[85] Hiroshi Ishii. The tangible user interface and its evolution. *Commun. ACM*, 51(6):32–36. 29

[86] Hiroshi Ishii and Brygg Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, CHI '97, pages 234–241, 1997. 29

[87] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 559–568, 2011. 61

[88] Kugsang Jeong Kugsang Jeong, Jongho Won Jongho Won, and Changseok Bae Changseok Bae. User activity recognition and logging in distributed intelligent gadgets. In *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 683–686, 2008. 32

[89] Michael Johnson, Michael Healy, Pepijn van de Ven, Martin J Hayes, John Nelson, Thomas Newe, and Elfed Lewis. A comparative review of wireless sensor network mote technologies. In *Sensors, 2009 IEEE*, pages 1439–1442. IEEE, 2009. 85

[90] Sergi Jordà, Günter Geiger, Marcos Alonso, and Martin Kaltenbrunner. The reactable: exploring the synergy between live music performance and tabletop tangible interfaces. In *Proceedings of the 1st international conference on Tangible and embedded interaction*, TEI '07, pages 139–146, 2007. 30

[91] Rudolph Emil Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 82 (Series D):35–45, 1960. 64, 85, 97

[92] Martin Kaltenbrunner and Ross Bencina. reactivision: a computer-vision framework for table-based tangible interaction. In *Proceedings of the 1st*

international conference on Tangible and embedded interaction, TEI '07, pages 69–74, 2007. 32

[93] Li-Wei Kang and Chun-Shien Lu. Distributed compressive video sensing. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 1169 –1172, april 2009. 125

[94] L. B Kara and K. Shimada. Sketch-based 3d-shape creation for industrial styling design. *IEEE Computer Graphics and Applications*, pages 60–71, 2007. 60

[95] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 79–86, 2010. 125

[96] O. A Karpenko and J. F Hughes. SmoothSketch: 3D free-form shapes from complex sketches. *Proceedings of ACM SIGGRAPH 2006*, 25(3):589–598, 2006. 60

[97] Juha Kela, Panu Korpipaa, Jani Mantyjarvi, Sanna Kallio, Giuseppe Savino, Luca Jozzo, and Di Marca. Accelerometer-based gesture control for a design environment. *Personal Ubiquitous Comput.*, 10(5):285–299. 28, 32

[98] Eamonn J. Keogh, Selina Chu, David Hart, and Michael J. Pazzani. An online algorithm for segmenting time series. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, ICDM '01, pages 289–296, 2001. 33

[99] Laehyun Kim, Hyunchul Cho, Sehyung Park, and Manchul Han. A tangible user interface with multimodal feedback. In *Proceedings of the 12th international conference on Human-computer interaction: intelligent multimodal interaction environments*, HCI'07, pages 94–103, 2007. 31

[100] Ming Hsiao Ko, Geoff West, Svetha Venkatesh, and Mohan Kumar. Using dynamic time warping for online temporal fusion in multisensor systems. *Inf. Fusion*, 9(3):370–388. 32

[101] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009. 122, 130, 132

[102] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Transactions on Computers*, 42(8):30–37, 2009. 124, 130

[103] Yehuda Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, April 2010. 125

[104] Matthias Kranz, Dominik Schmidt, Paul Holleis, and Albrecht Schmidt. A display cube as a tangible user interface. adj. In *Proc. of UbiComp 2005*. Springer, 2005. 31

[105] A. Kus, E. Unver, and A. Taylor. A comparative study of 3D scanning in engineering, product and transport design and fashion design education. *Computer Applications in Engineering Education*, 17(3):263–271, 2009. 58

[106] Christopher Lee and Yangsheng Xu. *Online, interactive learning of gestures for human/robot interfaces*, volume 4, pages 2982–2987. 1996. 28

[107] Kiryung Lee, Y. Bresler, and M. Junge. Subspace methods for joint sparse recovery. *Information Theory, IEEE Transactions on*, 58(6):3613 –3641, June 2012. 124

[108] Chunyuan Liao and Françcois Guimbretièere. Evaluating and understanding the usability of a pen-based command system for interactive paper. *ACM Trans. Comput.-Hum. Interact.*, 19(1), May 2012. 61

[109] Haiying Liu, Yunsong Li, Song Xiao, and Chengke Wu. Distributed compressive hyperspectral image sensing. In *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2010 Sixth International Conference on*, pages 607 –610, oct. 2010. doi: 10.1109/IIHMSP.2010.154. 125

[110] Konrad Lorincz, David J. Malan, Thaddeus R. F. Fulford-Jones, Alan Nawoj, Antony Clavel, Victor Shnayder, Geoffrey Mainland, Matt Welsh, and Steve Moulton. Sensor networks for emergency response: Challenges and opportunities. *IEEE Pervasive Computing*, 3(4):16–23. 84

[111] Konrad Lorincz, Bor-rong Chen, Geoffrey Werner Challen, Atanu Roy Chowdhury, Shyamal Patel, Paolo Bonato, Matt Welsh, et al. Mercury: a wearable sensor network platform for high-fidelity motion analysis. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 183–196. ACM, 2009. 86

[112] HJ Luinge and PH Veltink. Measuring orientation of human body segments using miniature gyroscopes and accelerometers. *Medical and Biological Engineering and Computing*, 43(2):273–282, 2005. 85

[113] Chong Luo, Feng Wu, Jun Sun, and Chang Wen Chen. Efficient measurement generation and pervasive sparsity for compressive data gathering. *Wireless Communications, IEEE Transactions on*, 9(12):3728 –3738, december 2010. ISSN 1536-1276. doi: 10.1109/TWC.2010.092810.100063. 125

[114] Sebastian O H Madgwick, Andrew J L Harrison, and Andrew Vaidyanathan. Estimation of imu and marg orientation using a gradient descent algorithm. *IEEE International Conference on Rehabilitation Robotics proceedings*, 2011. 73, 85, 101

[115] Mohammadreza Mahmudimanesh, Abdelmajid Khelil, and Neeraj Suri. Reordering for better compressibility: Efficient spatial sampling in wireless sensor networks. In *Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC), 2010 IEEE International Conference on*, pages 50 –57, june 2010. doi: 10.1109/SUTC.2010.30. 125

[116] Robert Mahony, Tarek Hamel, and J-M Pflimlin. Nonlinear complementary filters on the special orthogonal group. *Automatic Control, IEEE Transactions on*, 53(5):1203–1218, 2008. 85, 101

[117] A. Makhzani and S. Valaee. Reconstruction of a generalized joint sparsity model using principal component analysis. In *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2011 4th IEEE International Workshop on*, pages 269 –272, dec. 2011. doi: 10.1109/CAMSAP.2011.6136001. 125

[118] H. Mamaghanian, N. Khaled, D. Atienza, and P. Vandergheynst. Compressed sensing for real-time energy-efficient ecg compression on wireless body sensor nodes. *Biomedical Engineering, IEEE Transactions on*, 58(9): 2456 –2466, sept. 2011. ISSN 0018-9294. doi: 10.1109/TBME.2011.2156795. 125

[119] H. Mamaghanian, N. Khaled, D. Atienza, and P. Vandergheynst. Design and exploration of low-power analog to information conversion based on compressed sensing. *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, PP(99):1 –9, 2012. ISSN 2156-3357. doi: 10.1109/JETCAS.2012.2220253. 122

[120] Danilo P. Mandic, Dragan Obradovic, Anthony Kuh, Tülay Adali, Udo Trutschel, Martin Golz, Philippe De Wilde, Javier Barria, Anthony Constantinides, and Jonathon Chambers. Data fusion for modern engineering applications: an overview. In *Proceedings of the 15th international conference on Artificial neural networks: formal models and their applications - Volume Part II*, ICANN'05, pages 715–721, 2005. 63

[121] Steve Mann. ̈smart clothing ̈: wearable multimedia computing and ̈personal imaging ̈to restore the technological balance between people and their environments. In *Proceedings of the fourth ACM international conference on Multimedia*, MULTIMEDIA '96, pages 163–174, 1996. 29

[122] V M Mantyla, J Mantyjarvi, T Seppanen, and E Tuulari. Hand gesture recognition of a mobile device user, 2000. 32

[123] Philippe Martin and Erwan Salaün. Design and implementation of a low-cost observer-based attitude and heading reference system. *Control Engineering Practice*, 18(7):712–722, 2010. 86, 101

[124] A. Mazalek, M. Reynolds, and G. Davenport. TViews: An Extensible Architecture for Multiuser Digital Media Tables. *Computer Graphics and Applications, IEEE*, 26(5):47–55, 2006. 30

[125] Bojan Milosevic, Elisabetta Farella, and Luca Benini. Continuous gesture recognition for resource constrained smart objects. *UBICOMM10: The Fourth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, pages pp.391–396, October 2010. 29

[126] Emiliano Miluzzo, Nicholas D. Lane, Kristóf Fodor, Ronald Peterson, Hong Lu, Mirco Musolesi, Shane B. Eisenman, Xiao Zheng, and Andrew T. Campbell. Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, SenSys '08, pages 337–350, 2008. 85

[127] MIT Oxygen Project. `http://oxygen.csail.mit.edu/`. 11

[128] J. Mitani, H. Suzuki, and F. Kimura. 3D sketch: Sketch-based model reconstruction and rendering. In *Seventh IFIP WG 5.2 Workshop on Geometric Modeling*, 2000. 60

[129] S. Mitra and T. Acharya. Gesture recognition: A survey. *Trans. Sys. Man Cyber Part C*, 37(3):311–324. 28

[130] S. Morigi and M. Rucci. Reconstructing surfaces from sketched 3d irregular curve networks. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling*, SBIM '11, pages 39–46, 2011. 59

[131] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa. Fibermesh: designing freeform surfaces with 3D curves. volume 26, 2007. 60

[132] Deanna Needell and Roman Vershynin. Uniform uncertainty principle and signal recovery via&#x00a0;regularized orthogonal matching pursuit. *Found. Comput. Math.*, 9(3):317–334. 128

[133] Sejin Oh and Woontack Woo. Tangible media control system for intuitive interactions with multimedia contents. *IEICE - Trans. Inf. Syst.*, pages 53–61. 32

[134] A Olivares, G Olivares, F Mula, JM Górriz, and J Ramírez. Wagyromag: Wireless sensor network for monitoring and processing human body movement in healthcare applications. *Journal of Systems Architecture*, 57(10): 905–915, 2011. 83, 86

[135] C OMalley and D Stanton Fraser. Literature review in learning with tangible technologies. *Technology*, 11(2004):48, 2010. URL http://opus.bath.ac.uk/9506/. 29

[136] Kyoung Shin Park, Hyun-Sang Cho, Jaewon Lim, Yongjoo Cho, Seungmook Kang, and Soyon Park. Learning cooperation in a tangible moyangsung. In *Proceedings of the 2nd international conference on Virtual reality*, ICVR'07, pages 689–698, 2007. 30

[137] PERSIST EU Project. http://www.ict-persist.eu/. 12

[138] Miguel A. Prada, Janne Toivola, Jyrki Kullaa, and Jaakko Hollmn. Three-way analysis of structural health monitoring data. *Neurocomputing*, 80(0): 119 – 128, 2012. 126

[139] G. Quer, R. Masiero, D. Munaretto, M. Rossi, J. Widmer, and M. Zorzi. On the interplay between routing and signal representation for compressive sensing in wireless sensor networks. In *Information Theory and Applications Workshop, 2009*, pages 206 –215, feb. 2009. doi: 10.1109/ITA.2009.5044947. 125

[140] Lawrence R. Rabiner. Readings in speech recognition. chapter A tutorial on hidden Markov models and selected applications in speech recognition, pages 267–296. 1990. 32, 34, 37

[141] V. Raghunathan, S. Ganeriwal, and M. Srivastava. Emerging techniques for long lived wireless sensor networks. *IEEE Comm. Magazine*, 44(4):108–114, 2006. 121

[142] J. Ranieri, R. Rovatti, and G. Setti. Compressive sensing of localized signals: Application to analog-to-information conversion. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 3513 –3516, 30 2010-june 2 2010. doi: 10.1109/ISCAS.2010.5537820. 122

[143] M. Rauterberg, T. Mauch, and R. Stebler. The digital playing desk: a case study for augmented reality. In *In: Proceedings of International Workshop on Robot and Human Communication ROMAN96*, pages 410–415, 1996. 29

[144] Hongliang Ren, Denis Rank, Martin Merdes, Jan Stallkamp, and Peter Kazanzides. Multisensor data fusion in an integrated tracking system for endoscopic surgery. *Trans. Info. Tech. Biomed.*, 16(1):106–111, January 2012. 62

[145] Daniel Roetenberg, Henk Luinge, and Per Slycke. Xsens mvn: full 6dof human motion tracking using miniature inertial sensors. *Xsens Motion Technologies BV, Tech. Rep*, 2009. 83, 87

[146] Bharatula N.B. Stäger M. Lukowicz P. Tröster G. Roggen, D. From sensors to miniature networked sensor buttons. In *Proc. 3rd Int. Conf. on Networked Sensing Systems - INSS 2006*, pages 119–122, June 2006. 32

[147] Fariba Sadri. Ambient intelligence: A survey. *ACM Comput. Surv.*, 43(4): 36:1–36:66, October 2011. 11

[148] D. Sampaio, L.P. Reis, and R. Rodrigues. A survey on ambient intelligence projects. In *Information Systems and Technologies (CISTI), 2012 7th Iberian Conference on*, pages 1 –6, june 2012. 11

[149] H Martin Schepers, Daniel Roetenberg, and Peter H Veltink. Ambulatory human motion tracking by fusion of inertial and magnetic sensing with adaptive actuation. *Medical and Biological Engineering and Computing*, 48 (1):27–37, 2010. 85

[150] Tal Shany, Stephen J Redmond, Michael R Narayanan, and Nigel H Lovell. Sensors-based wearable systems for monitoring of human movement and falls. *Sensors Journal, IEEE*, 12(3):658–670, 2012. 85

[151] SOFIA EU Project. http://www.sofia-community.org/. 13

[152] Sony. Playstation move. URL http://us.playstation.com/ps3/playstation-move/. 31

[153] Alan N Steinberg, Christopher L Bowman, and Franklin E White. Revisions to the jdl data fusion model. *Proceedings of SPIE*, 3719(1):430–441, 1999. 62

[154] Thomas Stiefmeier, Georg Ogris, Holger Junker, Paul Lukowicz, and Gerhard Tröster. Combining motion sensors and ultrasonic hands tracking for continuous activity recognition in a maintenance scenario. In *10th IEEE International Symposium on Wearable Computers*, October 2006. 28, 32

[155] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009, January 2009. 124

[156] Muhammad Tahir, Gilles Bailly, and Eric Lecolinet. Aremote: A tangible interface for selecting tv channels. In *Proceedings of the 17th International Conference on Artificial Reality and Telexistence*, ICAT '07, pages 298–299, 2007. 32

[157] T.M. Takala, P. Rauhamaa, and T. Takala. Survey of 3DUI applications and development challenges. In *3D User Interfaces (3DUI), 2012 IEEE Symposium on*, pages 89 –96, march 2012. 60

[158] Pingbo Tang, Daniel Huber, Burcu Akinci, Robert Lipman, and Alan Lytle. Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques. *Automation in Construction*, 19(7):829 – 843, 2010. 58

[159] Emmanuel Tapia, Stephen Intille, and Kent Larson. Activity recognition in the home using simple and ubiquitous sensors. *Pervasive Computing*, pages 158–175, 2004. 85

[160] Kyoko Ueda, Atsushi Kosaka, Ryoichi Watanabe, Yoshinori Takeuchi, Takao Onoye, Yuichi Itoh, Yoshifumi Kitamura, and Fumio Kishino. m-activecube: multimedia extension of spatial tangible user interface. In *Proceedings of the Second international conference on Biologically Inspired Approaches to Advanced Information Technology*, BioADIT'06, pages 363–370, 2006. 32, 33

[161] Vaisala. Wxt250. `http://www.vaisala.com/en/products/multiweathersensors/Pages/WXT520.aspx`, Mar 2012. 136, 138

[162] Rudolph Van Der Merwe and Eric A Wan. Sigma-point kalman filters for integrated navigation. In *Proceedings of the 60th Annual Meeting of the Institute of Navigation (ION)*, pages 641–654, 2004. 85

[163] Tamas Varady, Ralph R Martin, and Jordan Cox. Reverse engineering of geometric models - an introduction. *Computer-Aided Design*, 29(4):255 – 268, 1997. 58

[164] Venugopal V. Veeravalli and Pramod K. Varshney. Distributed inference in wireless sensor networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 370(1958):100–117, 2012. 121, 122

[165] Janneke Verhaegh, Willem Fontijn, and Aljosja Jacobs. On the benefits of tangible interfaces for educational games. In *Proceedings of the 2008 Second IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning*, DIGITEL '08, pages 141–145, 2008. 29

[166] N. Verma, P. Zappi, and T. Rosing. Latent variables based data estimation for sensing applications. In *IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pages 335–340, 2011. 125, 126

[167] Jack Volder. The cordic computing technique. In *Papers presented at the the March 3-5, 1959, western joint computer conference*, IRE-AIEE-ACM '59 (Western), pages 257–261, 1959. 47

[168] Jamie A. Ward, Paul Lukowicz, Gerhard Troster, and Thad E. Starner. Activity recognition of assembly tasks using body-worn microphones and accelerometers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(10):1553–1567. 28

[169] Ryoichi Watanabe, Yuichi Itoh, Yoshifumi Kitamura, Fumio Kishino, and Hideo Kikuchi. Distributed autonomous interface using activecube for interactive multimedia contents. In *Proceedings of the 2005 international conference on Augmented tele-existence*, ICAT '05, pages 22–29, 2005. 32

[170] Thibaut Weise, Thomas Wismer, Bastian Leibe, and Luc Van Gool. Online loop closure for real-time interactive 3d scanning. *Comput. Vis. Image Underst.*, 115(5):635–648. 61

[171] Mark Weiser. The computer for the 21st century. *Scientific American*, 265: 94–104, 1991. 9, 29

[172] Greg Welch and Eric Foxlin. Motion tracking: No silver bullet, but a respectable arsenal. *IEEE Comput. Graph. Appl.*, 22(6):24–38. 83

[173] Gregory F Welch. The use of the kalman filter for human motion tracking in virtual reality. *Presence: Teleoperators and Virtual Environments*, 18 (1):72–91, 2009. 83

[174] Wiibrew. Wiimote reverse engineering. URL `http://wiibrew.org/wiki/Wiimote`. 31

[175] Chadwick A. Wingrave, Brian Williamson, Paul D. Varcholik, Jeremy Rose, Andrew Miller, Emiko Charbonneau, Jared Bott, and Joseph J. LaViola Jr. The wiimote and beyond: Spatially convenient devices for 3d user interfaces. *IEEE Comput. Graph. Appl.*, 30(2):71–85. 31, 83

[176] World Meteorological Organization. Wind measurement and archival under the automated surface observing system (ASOS): User concerns and opportunity for improvement. `http://www.aoml.noaa.gov/hrd/Powell/ArchiveASOS.pdf`. 140

[177] K. Yoshii, M. Goto, K. Komatani, T. Ogata, and H.G. Okuno. An efficient hybrid music recommender system using an incrementally trainable probabilistic generative model. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(2):435 –447, feb. 2008. 124

[178] AD Young. Comparison of orientation filter algorithms for realtime wireless inertial posture tracking. In *Wearable and Implantable Body Sensor Networks, 2009. BSN 2009. Sixth International Workshop on*, pages 59–64. IEEE, 2009. 86

[179] AD Young, MJ Ling, and DK Arvind. Orient-2: A realtime wireless posture tracking system using local orientation estimation. In *Proceedings of the 4th workshop on Embedded networked sensors*, pages 53–57. ACM, 2007. 86

[180] Zhuizhuan Yu and S. Hoyos. Digitally assisted analog compressive sensing. In *Circuits and Systems Workshop,(DCAS), 2009 IEEE Dallas*, pages 1 –4, oct. 2009. 126

[181] Piero Zappi, Clemens Lombriser, Thomas Stiefmeier, Elisabetta Farella, Daniel Roggen, Luca Benini, and Gerhard Trster. Activity recognition from on-body sensors: Accuracy-power trade-off by dynamic sensor selection. In *European Conference on Wireless Sensor Networks*, pages 17–33, 2008. 121

[182] Piero Zappi, Bojan Milosevic, Elisabetta Farella, and Luca Benini. Hidden markov model based gesture recognition on low-cost, low-power tangible user interfaces. *Entertainment Computing*, 1(2):75 – 84, 2009. 28, 38

[183] R. C Zeleznik, K. P Herndon, and J. F Hughes. SKETCH: an interface for sketching 3D scenes. In *International Conference on Computer Graphics and Interactive Techniques*, 2006. 60

[184] Ludwig Zeller and Lasse Scherffig. Cubebrowser: a cognitive adapter to explore media databases. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '09, pages 2619–2622, 2009. 32

[185] Bofeng Zhang, Susu Jiang, Daming Wei, Michael Marschollek, and Wu Zhang. State of the art in gait analysis using wearable sensors for

healthcare applications. In *Computer and Information Science (ICIS), 2012 IEEE/ACIS 11th International Conference on*, pages 213–218. IEEE, 2012. 85