Analysing Crowd Behaviours using Mobile Sensing

Kleomenis Katevas

Queen Mary University of London

This dissertation is submitted for the degree of Doctor of Philosophy

Abstract

Researchers have examined crowd behaviour in the past by employing a variety of methods including ethnographic studies, computer vision techniques and manual annotation-based data analysis. However, because of the resources to collect, process and analyse data, it remains difficult to obtain large data sets for study. Mobile phones offer easier means for data collection that is easy to analyse and can preserve the user's privacy. The aim of this thesis is to identify and model different qualities of social interactions inside crowds using mobile sensing technology. This Ph.D. research makes three main contributions centred around the mobile sensing and crowd sensing area.

Firstly, an open-source licensed mobile sensing framework is developed, named SensingKit, that is capable of collecting mobile sensor data from iOS and Android devices, supporting most sensors available in modern smartphones. The framework has been evaluated in a case study that investigates the pedestrian gait synchronisation phenomenon.

Secondly, a novel algorithm based on graph theory is proposed capable of detecting stationary social interactions within crowds. It uses sensor data available in a modern smartphone device, such as the Bluetooth Smart (BLE) sensor, as an indication of user proximity, and accelerometer sensor, as an indication of each user's motion state.

Finally, a machine learning model is introduced that uses multi-modal mobile sensor data extracted from Bluetooth Smart, accelerometer and gyroscope sensors. The validation was performed using a relatively large dataset with 24 participants, where they were asked to socialise with each other for 45 minutes. By using supervised machine learning based on gradient-boosted trees, a performance increase of 26.7% was achieved over a proximity-based approach. Such model can be beneficial to the design and implementation of in-the-wild crowd behavioural analysis, design of influence strategies, and algorithms for crowd reconfiguration.

Declaration

I, Kleomenis Katevas, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party's copyright or other Intellectual Property Right, or contain any confidential material. I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis. I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

The copyright of this thesis rests with the author; no quotation from it or information derived from it may be published without the prior written consent of the author.

Kleomenis Katevas
November 20, 2018

Dedicated to Evi,
for her love and continuous support
throughout this journey.

Acknowledgements

First and foremost, I would like to thank my two supervisors for their valuable guidance and support through all these years. I thank Laurissa Tokarchuk for her phenomenal support, wise guidance, for helping me grow in the scientific community and for always encouraging me to try something new and different. I thank Hamed Haddadi for his passion and excitement, insightful feedback, and for giving me the opportunity to collaborate with some of the top people in the field.

I am grateful to Martin Pielot, for being a mentor and friend to me during my two summer internships at Telefónica Research in Barcelona. Special thanks to my good friends and collaborators there, Ioannis Arapakis, Ilias Leontiadis and Joan Serrà, for teaching me so much about research.

I also thank the members of the Cognitive Science research group. Pat Healey for introducing me to the world of research. Katrin Hänsel, Daniel Gabana Arellano, Alessio Xompero, Mohammad Malekzadeh, Antonella Mazzoni, Lida Theodorou and Jeni Maleshkova, for being the best Ph.D. officemates and collaborators. Richard Clegg from the Networks research group for his ideas and insightful feedback and Akram Alomainy from Antennas & Electromagnetics research group for his inspiring discussions.

I sincerely thank my examiners, Mirco Musolesi and Ioannis Patras, for their constructive and insightful comments on my thesis during my viva.

Last, and most importantly, to my family and friends for their love and support all these years.

This work was supported by funding from the UK Defence Science & Technology Laboratory (DSTL).

Contents

1 Introduction **11**

1.1 Motivation and Scope 11
1.2 Objectives and Challenges 13
1.3 Novelty and Contribution 14
1.4 List of Publications 15
1.5 Report Structure 17

2 Background and Related Work **19**

2.1 Overview 19
2.2 Mobile Sensing 19
2.2.1 Available Sensors 20
2.2.2 Available Virtual Sensors 23
2.2.3 Sensor Fusion 24
2.2.4 Motion Co-Processors 25
2.2.5 Sensing in the Background 25
2.2.6 Continuous Sensing Platforms 26
2.3 Crowd Sensing 27
2.3.1 Detecting Mobility 28
2.3.2 Detecting Walking Social Interactions 28
2.3.3 Detecting Stationary Social Interactions 30
2.4 Summary 32

3 SensingKit – A Multi-Platform Mobile Sensing Framework **34**

3.1 Overview 34
3.2 SensingKit Platform 36
3.3 CrowdSense App 38
3.4 Evaluation of Battery Consumption 39
3.5 Pilot Study: Analysing Pedestrian Behaviour 41
3.5.1 Experiment 42
3.5.2 Results and Discussion 44

3.5.3 Conclusion . . . 47
3.6 Summary . . . 48

4 Towards Detecting Social Interactions **50**
4.1 Overview . . . 50
4.2 Validation Experiments . . . 51
4.3 Preliminary Study in Detecting F-formations . . . 55
4.3.1 Participants . . . 55
4.3.2 Sensor Data Collection . . . 56
4.3.3 Procedure . . . 56
4.3.4 Ground Truth . . . 58
4.3.5 Data Pre-Processing . . . 59
4.3.6 The DetecTN Algorithm . . . 60
4.3.7 Results . . . 61
4.4 Summary . . . 63

5 Detecting Social Interactions **65**
5.1 Overview . . . 65
5.2 Validation Experiments . . . 66
5.2.1 Evaluating the Effect of Proximity in the RSSI . . . 67
5.2.2 Evaluating the Effect of Body Orientation in the RSSI . . . 68
5.3 Data Set . . . 71
5.3.1 Participants . . . 71
5.3.2 Procedure . . . 72
5.3.3 Sensor Data Set . . . 74
5.4 Data Analysis . . . 74
5.4.1 Data Pre-processing . . . 75
5.4.2 Target Variable . . . 76
5.4.3 Ground Truth . . . 76
5.4.4 Normalised Inverted Proximity . . . 77
5.4.5 Feature Selection . . . 77
5.4.6 Evaluation Procedure . . . 80
5.4.7 Model Choice . . . 81
5.5 Results . . . 82
5.5.1 Baseline . . . 82
5.5.2 General Performance . . . 83
5.5.3 Normalise Predictions . . . 83
5.5.4 Performance Per Participant . . . 85
5.5.5 Sensor Importance . . . 85

5.5.6 Feature Importance . . . 86
5.5.7 Probabilistic Threshold Choice . . . 87
5.6 Summary . . . 88
5.6.1 Battery Consumption Implications . . . 89
5.6.2 Privacy Implications . . . 90
5.6.3 Limitations . . . 90

6 Conclusion and Future Work **91**
6.1 Conclusion . . . 91
6.2 Future Work . . . 92

Bibliography **94**

A QMUL Ethics Committee Approval **105**

B Featured Work **109**
B.1 SensingKit framework . . . 109
B.2 Walking in Sync work . . . 109
B.3 Group Detection Algorithm . . . 110

C Proximity Evaluation Plots **111**

D Model Tuning Performance **113**

E SensingKit Developer Instructions **115**
E.1 SensingKit for Android . . . 115
E.2 SensingKit for iOS . . . 117

F Source Code **120**

List of Figures

2.1 Illustration of a Sensor Fusion technique 25
2.2 Example of F-formations in real-world 30

3.1 SensingKit System Architecture 36
3.2 CrowdSense App for iOS 38
3.3 Battery consumption of SensingKit in iOS 39
3.4 Acceleration magnitude and autocorrelation 45
3.5 Pearson correlation matrix 46
3.6 Cross-correlation analysis 46
3.7 Pearson-correlation applied to windows of 1 sec 47

4.1 Bluetooth Smart RSSI box plot 53
4.2 Evaluation of distance estimation models 55
4.3 Floor plan of the Performance Lab at QMUL 57
4.4 ELAN manual annotations 59
4.5 Overview of the DetecTN algorithm 60
4.6 Heat map reporting the performance of the DetecTN algorithm 62

5.1 Measuring the effect of proximity on iBeacon™ signal. 68
5.2 Measuring the effect of body orientation on iBeacon™ signal 69
5.3 Heat map visualisation of the effect of body orientation in iBeacon™ signal. 70
5.4 Examples of different body orientations 71
5.5 Floor plan of the Performance Lab at QMUL 73
5.6 Performance of NPC and NIP baselines 83
5.7 General performance of XGBoost classifier 84
5.8 Performance of XGBoost classifier using normalisation 84
5.9 Performance of XGBoost classifier per participant 85
5.10 Sensor importance of XGBoost classifier 86

B.1 System Design of the demo presented in ACM MobiSys 2017. 110

C.1 Measuring the effect of proximity on iBeacon™ signal. 112

List of Tables

2.1 Continuous Sensing Platforms 27

3.1 SensingKit Supported Sensors 37
3.2 SensingKit evaluation: Device Specification 39
3.3 Battery consumption of SensingKit in iOS 40
3.4 Device Specification 43
3.5 Demographic Information 43
3.6 Groups Configuration 43

4.1 Participant demographic information. 56
4.2 Scenes with group formations tested during the experiment. 58
4.3 Confusion matrix reporting the performance of the DetecTN algorithm . . 62

5.1 List of the features used in the data analysis. 78
5.2 Feature importance of XGBoost classifier 87
5.3 Confusion matrix reporting the performance of the XGBoost classifier . . . 88
5.4 Confusion matrix reporting the performance of NIP 88

D.1 Parameter Tuning for Logistic Regression model 113
D.2 Parameter Tuning for Random Forest model 113
D.3 Parameter Tuning for XGBoost model 113
D.4 Detailed Parameter Tuning for XGBoost model 114

Chapter 1

Introduction

1.1 Motivation and Scope

It is well known that crowds share common behavioural patterns. For example, pedestrian flocks inside a crowd tend to move together as a unit; group members walk at the same speed, follow the same goals and when they become separated, they tend to reform their group [64]. Similar social interactions exist on stationary groups with two or more participants, known as F-formations. These groups orient their bodies is such ways that they have equal and direct access with each other [38].

Researchers have examined crowd behaviour in the past by employing a variety of methods including ethnographic studies [79, 19, 90], computer vision techniques [54, 87, 6] and manual annotation-based data analysis [26, 15]. Despite the evidential validity of the results, the scale and the sample were limited due to the resources required to compile, process and analyse the data. More precisely, the use of video cameras as a medium for data collection constitutes a major constraint. Camera surveillance infrastructure can be expensive and is not available everywhere, especially when the requirement is to monitor a spontaneous event such as a riot. Monitoring the video feed manually requires huge amount of man power and can be prone to human errors. Moreover, the use of computer vision systems can be computationally expensive as lots of processing power per video feed is required.

The emergence of mobile sensing technology provides a unique set of capabilities to overcome those limitations. While the first mobiles were primarily used as communication devices, modern smartphones are a ubiquitous part of people's lives and offer the ability to non-intrusively capture high quality sensor data such as the motion, orientation or location of the individual. In addition, modern smartphones come equipped with fast processing power, large memory and highly available wireless connectivity. With the existence of application distribution channels such as the Apple App Store and the Google Play Store, researchers can distribute applications and collect large-scale data in ways that previously

were not possible. This includes information about an individual, a group of people or entire communities such as crowds [43].

In the last few years there have been several attempts to examine human crowd behaviour automatically using mobile sensing technology. Wirz *et al.* [96] proposed an on-line method for detecting pedestrian flocks using GPS traces. Roggen *et al.* [82] developed a framework that only uses one on-body accelerometer sensor installed on each user's hip as a simulation of a phone placed in the user's trouser pocket. Kjærgaard *et al.* used a combination of various mobile sensors (*i.e.* accelerometer, compass and WiFi) in order to detect pedestrian flocks from crowds with an accuracy of up to 87%. All previously mentioned works only focus on detecting the social interactions of walking pedestrians, excluding stationary interactions (*i.e.* standing interactions between two or more participants) that are more likely to happen in planned events such as social gatherings or networking events.

Matic *et al.* [51] presented a solution for detecting stationary interactions using the smartphone's WiFi and magnetometer sensors, together with an external accelerometer sensor installed on each user's chest as an indication of the user's speech status. Montanari *et al.* [57] created a wearable device that uses near-infrared light to measure the user proximity and relative orientation for measuring interaction proxemics. Both works depend on external hardware, making the distribution of such technology difficult to be used in larger social gatherings.

Palaghias *et al.* [65] presented a system for identifying social interactions in real-world scenarios using sensors available in the user's smartphone (*i.e.* Bluetooth, accelerometer and magnetometer). In contrast to all previously mentioned works, this approach only uses the user's smartphone as a data source, with no other external hardware or cloud-based infrastructure required to be present. One of the main limitations of this work is that it has only been evaluated in an artificial environmental setting with eight participants performing one-to-one interactions, using human observations as ground truth. Moreover, the trained model would only work with users having the specific device model used in the study (HTC One S). Currently there is no research the author is aware of that uses a sensor based automated method to identify dynamic groups (*i.e.* social groups that their size changes over time) of variety of sizes that is not device dependent.

Considering the limitation of the presented related work, the motivation of this research is expressed by the following research questions:

1. *How can we detect stationary social interactions happening in planned events using mobile sensing technology?*

2. *Which sensors are the most appropriate for answering Research Question 1?*

Answering the research questions stated above will allow us to understand specific

characteristics and behaviours from groups within crowds in a fully automated way. Furthermore, it can enable the identification of socially influential individuals. This can be very useful in cases of emergency as pedestrians can receive personalised information of how to evacuate a building in the most efficient way.

1.2 Objectives and Challenges

The aim of this research is to identify and model different qualities of social interactions of groups inside crowds. This will be achieved by exploring different types of mobile sensors such as *accelerometer*, *gyroscope*, and *Bluetooth*. It is important to mention that this research will only focus on individuals aged from 18 to 60 and will exclude children and older people as the dynamics of such crowds can be significantly different. Moreover, it will focus on individuals living in Northern Europe (London, UK) as interpersonal distance, body orientation and touching behaviour can differ based on cultural background differences [80]. Finally, it will only focus on planned events such as exhibitions and conferences and will not explore unplanned events like riots or protests. The reason behind this decision is that it is unrealistic to organise an experiment that takes place in unplanned events because of its spontaneous nature. It is expected that people will be less likely to install an application that collects sensor data from their personal device while being on a riot or protest. It is however expected that this work will have implication to any group socially engaged in unplanned events as some of the crowd characteristics are similar.

To achieve the aim of this research, the following objectives have been defined:

1. Design and implement a mobile application that will allow the automated collection of mobile sensor data from crowds.

2. Conduct an initial experiment in order to understand the pedestrian behaviour and walking patterns while existing in a group.

3. Explore the sensors that are appropriate for detecting stationary social interactions through a controlled case study.

4. Conduct a larger study in a natural environment in order to identify ways of detecting stationary social interactions.

In order to achieve the objectives presented above, the following challenges need to be addressed:

1. One main challenge of this research is accessing the participants in an organised case study. Requesting access to people's mobile phone can be hard as it highly

depends on the people willingness to share their data. Concerns about the privacy of the data collection, or the battery life of their devices might discourage people from participating on this experiment. However according to Wirz *et al.* [95], people are open to share their data as long as they receive some benefits from it or if they realise that sharing such information is for their own good and safety. Thus, the benefits of such research need to be clearly communicated with the participants. Moreover, data collection should be anonymised before publishing the dataset for protecting the user's privacy.

2. Another important challenge in real-world experiments is establishing an accurate ground truth. Researchers have been using various methods to solve this, such as using annotations from human observers or post analysis of audio/video recordings. These methods can be time consuming, prone to human error and are hard to scale in larger environments. Other solutions are the use of experience sampling questionnaires, a method that is easier to scale in larger studies but is subjective to each user's responses.

3. The battery consumption of the mobile application during the sensor data collection is a factor that highly affects the success of this research. According to Lane *et al.* [43], "Mobile phones cannot be overloaded with continuous sensing commitments that undermine the performance of the phone". Since some of the sensors such as the GPS or the gyroscope require a lot of processing power, the battery life of the smartphones will be limited. Thus, the impact on the battery life of each sensor as well as the optimal choice for achieving the objectives of this thesis should be investigated.

1.3 Novelty and Contribution

According to the objectives presented in Section 1.2, the main contributions of this research are the following:

1. SensingKit, an open-source licensed mobile sensing framework suitable for capturing sensor data from smartphone devices. Unlike other existing platforms, explained in more details in Section 2.2.6, this framework is cross-platform and covers the majority of today's mobile operating systems (*i.e.* Apple's iOS and Google's Android). It uses the same Application Programming Interface (API) for both versions, making it easy for researchers to build cross-platform apps to support their studies. It also includes device proximity capabilities using features of the Bluetooth Low Energy (BLE) standard. Finally, all data timestamps are based on the device's CPU time

base register rather than the system's clock to avoid timing issues when the clock changes.

2. The DetecTN algorithm, a novel algorithm based on graph theory that is capable of detecting stationary social interactions within crowds. It uses sensor data available in a modern smartphone device, such as the Bluetooth Smart (BLE) sensor, as an indication of user proximity, and accelerometer sensor, as an indication of each user's motion state.

3. A probabilistic machine learning model that uses multi-modal mobile sensor data extracted from Bluetooth Smart, accelerometer and gyroscope sensors for detecting stationary interactions inside crowds. The validation was performed using a relatively large dataset with 24 participants socially interacting in a natural setting for 45 minutes. This rich data set will also be made available after the submission of this thesis.

1.4 List of Publications

During the course of this Ph.D. research, the following papers were published (in reverse-chronological order).

Publications related to this dissertation:

- *Finding Dory in the Crowd: Detecting Social Interactions using Multi-Modal Mobile Sensing.*
  Kleomenis Katevas, Katrin Hänzel, Richard Clegg, Ilias Leontiadis, Hamed Haddadi, Laurissa Tokarchuk.
  Work is under review.

- *Demo: Detecting Group Formations using iBeacon Technology* [36].
  Kleomenis Katevas, Laurissa Tokarchuk, Hamed Haddadi, Richard G. Clegg, Muhammad Irfan.
  ACM MobiSys 2017, Niagara Falls, NY, USA, June 2017.

- *Detecting Group Formations using iBeacon Technology* [33].
  Kleomenis Katevas, Hamed Haddadi, Laurissa Tokarchuk, Richard G. Clegg.
  4th International Workshop on Human Activity Sensing Corpus and Application (HASCA 2016) in conjunction with ACM UbiComp 2016, Heidelberg, Germany, September 2016.

- *SensingKit: Evaluating the Sensor Power Consumption in iOS Devices* [32].
  Kleomenis Katevas, Hamed Haddadi, Laurissa Tokarchuk.

12th International Conference on Intelligent Environments (IE'16), London, UK, September 2016.

- *Walking in Sync: Two is Company, Three's a Crowd* [30].
  Kleomenis Katevas, Hamed Haddadi, Laurissa Tokarchuk, Richard G. Clegg.
  2nd Workshop on Physical Analytics (WPA) in conjunction with ACM MobiSys 2015, Florence, Italy, May 2015.
  Brendan Murphy prize for presenting this work at the 2015 Next Generation Networking, Multi-Service Networks workshop, in July 2015.

- *Poster: SensingKit – A Multi-Platform Mobile Sensing Framework for Large-Scale Experiments* [28].
  Kleomenis Katevas, Hamed Haddadi, Laurissa Tokarchuk.
  Extended abstract, ACM MobiCom 2014, Maui, Hawaii, September 2014.
  Best poster award for presenting this work at the 2015 Marconi Society Symposium – "The Future Infrastructure of the Internet of Things", Paul Baran Young Scholars Poster Session, in October 2015.

Other publications published during the course of this research but are not part of the thesis:

- *Typical Phone Use Habits: Intense Use Does Not Predict Negative Well-Being* [37].
  Kleomenis Katevas, Ioannis Arapakis, Martin Pielot
  ACM MobileHCI '18, Barcelona, Spain, September 2018.

- *Continual Prediction of Notification Attendance: Classical Versus Deep Network Approaches* [34].
  Kleomenis Katevas, Ilias Leontiadis, Martin Pielot, Joan Serrà.
  arXiv preprint arXiv:1712.07120. 2017 Dec 19.

- *Beyond Interruptibility: Predicting Opportune Moments to Engage Mobile Phone Users* [68].
  Martin Pielot, Bruno Cardoso, Kleomenis Katevas, Joan Serrà, Aleksandar Matic, Nuria Oliver.
  ACM UbiComp 2017 (IMWUT), Maui, Hawaii, September 2017.

- *Practical Processing of Mobile Sensor Data for Continual Deep Learning Predictions* [35].
  Kleomenis Katevas, Ilias Leontiadis, Martin Pielot, Joan Serrà.
  1st International Workshop on Embedded and Mobile Deep Learning in conjunction with ACM MobiSys 2017, Niagara Falls, NY, USA, June 2017.

- *Robot Comedy Lab: Experimenting with the Social Dynamics of Live Performance* [31].
  Kleomenis Katevas, Patrick G.T. Healey, Matthew Tobias Harris.
  Frontiers in Psychology 6, 2015.

- *Robot Stand-up: Engineering a Comic Performance* [29].
  Kleomenis Katevas, Patrick G.T. Healey, Matthew Tobias Harris.
  Proceedings of the 2014 Workshop on Humanoid Robots and Creativity at the 2014 IEEE-RAS International Conference on Humanoid Robots (Humanoids 2014), Madrid, Spain, November 2014.

1.5 Report Structure

The rest of this report is organized as follows:

Chapter 2, *Background and Related Work*, presents a review of the literature of this research. It describes previous researches that have been carried out in the mobile sensing area as well as the difficulties faced when using mobile phones for sensor data gathering. Finally, it presents an overview of crowd sensing techniques and the related works that have been conducted in the research area of detecting walking and stationary social interactions.

Chapter 3, *SensingKit – A Multi-Platform Mobile Sensing Framework*, outlines the implementation and design details of SensingKit, an open-source framework for iOS and Android environments that is capable of accessing sensor data from mobile devices. It continuous with an evaluation of the power consumption of the framework while collecting data from various sensors, including motion, orientation, and location sensors. It also presents CrowdSense App, a cross-platform mobile sensing app that is capable of capturing all sensor data supported by SensingKit and save it into the device's memory in Comma-Separated Values (CSV) format. Finally, it concludes with a pilot study that investigates the phenomenon of gait synchronisation that is happening when groups of people walk together.

Chapter 4, *Towards Detecting Social Interactions*, presents the experimental design of a preliminary study that analysis stationary social interactions. It begins with an evaluation of three models for estimating the proximity between smartphone devices with the use of Bluetooth Smart sensor. It continues with the presentation of DetecTN, an algorithm for detecting stationary social interactions using Bluetooth and accelerometer sensor data. It concludes with an evaluation of the algorithm and presentation of the results and implications of this approach.

Chapter 5, *Detecting Social Interactions*, proposes a supervised machine learning model for detecting stationary interactions by analysing information extracted from multiple mobile sensors. It is an enhanced version of the DetecTN algorithm presented in

Chapter 4, evaluated in a relatively large dataset with 24 participants interacting for a total duration of 45 minutes.

Chapter 6, *Conclusion and Future Work*, presents a summary of all findings reported in the previous chapters and suggests the direction for future research.

Chapter 2

Background and Related Work

2.1 Overview

This chapter surveys the literature that is relevant to this Ph.D. research. It begins with a review of mobile sensing technology that focuses on *motion*, *orientation*, *location* and *proximity* sensing techniques. In addition, it outlines the difficulties and limitations of the described sensors. Finally, it discusses relevant research that has been conducted in the Crowd Sensing area, including the social engagement in moving or stationary crowds.

2.2 Mobile Sensing

Mobile sensing is the research area that uses mobile sensors collectively from an individual or across multiple scales to obtain information about the human characteristics and behaviour [43]. During the last few years, the research area of mobile sensing became more and more popular. The main reason behind this is that smartphones are now equipped with advanced mobile sensors that can measure our motion, location, orientation, or even external environmental properties such us the temperature or the humidity of the environment. The mobile platforms provide Application Programming Interfaces (APIs) to the developers, allowing access to most of these sensors. Another reason is that the devices now have advanced computational and storage capabilities, allowing us to do data processing (*e.g.* activity recognition, language processing) inside the smartphone, something that was not possible in the past years. Finally, using the application delivery channels such as the Apple App Store or the Google Play Sore, researchers can distribute applications to a large number of users and conduct studies in a much larger scale.

Lane *et al.* [43], in their thorough survey paper described different sensing scales that are possible through mobile sensing, named *Personal Sensing*, *Group Sensing* and *Community Sensing*. The Personal Sensing category represents the application that focuses on the individual, like activity recognition, exercise tracking or healthcare applications.

In Group Sensing we categorise applications that focus on groups of people with common interests (*e.g.* small communities). Finally, in Community Sensing we find applications that measure an entire community like a city (urban-scale sensing).

Knowing when to start the data collection from mobile sensors is also a tricky task in mobile sensing area. The most straight forward way of starting the data collection is when the user decides to enable the sensing application, a method called *Participatory Sensing* [88]. The advantage of using this method is that the user has the knowledge of when the sensing process began or ended and has the option of sharing the data or not [44]. In *Opportunistic Sensing* [10], the smartphone starts the sensing procedure automatically, based on triggers set by the developer. For instance, the device might use the ambient light sensor to identify when the user uses the smartphone in an outdoor environment, and only then start collecting GPS localisation data. The obvious benefit of using this method is improved battery efficiency, but less control in sharing the data.

2.2.1 Available Sensors

In this section, an overview is given of the most popular sensors in smartphones, its sensing capabilities as well as some of its current use in the research context. This section also provides information about the accessibility of these sensors in Android and iOS, two of the most popular mobile operating systems today.

Camera

Most mobile devices have at least one camera that is capable of recording images in real-time. It is mainly used for capturing video or photographs. In addition, researchers often use this sensor to provide Augmented Reality capabilities in mobile applications. Both Android and iOS provide extensive access to the cameras.

Microphone

Microphone is the sensor that collects audio from the environment by converting the sound into electrical signal. Some modern smartphones have a second microphone that is used to perform noise cancellation in noisy environments. Both Android and iOS provide access to the microphone.

Accelerometer

Accelerometer is a sensor that is widely used in the smartphone industry. It measures the force of acceleration caused either by the gravity or by the device's movement in three-axis (Surge, Heave and Sway). Since an accelerometer measures the total acceleration caused

both by the gravity and device's movement combined, a device at rest will show a force of approximately 1g upwards. In contrast, a device at free-fall will show a force of 0g.

The limitation of an accelerometer sensor is that it is noisy due to its dependency to the force of the gravity. Adding a low-pass filter to remove the noise greatly improves its performance but makes it less responsive due to the additional delay that is added.

Smartphones extensively use this sensor to calculate the gravity direction, find the device orientation and rotate the screen from portrait to landscape mode when needed.

Both Android and iOS platforms provide access to the accelerometer sensor. In iOS, developers can choose the preferred sampling rate from a range of 1 to 100Hz. However, Android platform only supports four sampling profiles:

- `SENSOR_DELAY_FASTEST`: Maximum sampling rate allowed on the device.
- `SENSOR_DELAY_GAME`: Sampling rate suitable for games.
- `SENSOR_DELAY_UI`: Sampling rate suitable for user-interface related tasks.
- `SENSOR_DELAY_NORMAL`: Default sampling rate suitable for screen orientation changes.

The sampling rate of each of these profiles varies from device to device. Furthermore, since a mobile device is not a real-time system, the configured rate is just a suggestion to the mobile device, with the actual sampling rate being unstable in both Android and iOS.

Gyroscope

A gyroscope is a sensor that is used to measure the device's rotation rate in three-axis (Pitch, Roll and Yaw). Even though the Pitch and Roll (but not the Yaw) of the device can be calculated with just an accelerometer sensor, it is very computational expensive as it requires trigonometric calculations. The gyroscope provides all three measurements in a less expensive and more accurate way.

Even though a gyroscope is not a noisy sensor as it doesn't depend on the gravity force, it suffers from a drift problem (also known as gyro bias drift) [83]. Every time a rotation is performed, there is a small error in the measurements that varies from sensor to sensor.

Both Android and iOS platforms provide access to the gyroscope sensor. In Android, the sampling rate can be set using the same sensor profiles described in Section 2.2.1

It is also worth mentioning that by combining the gyroscope with an accelerometer sensor, we can highly improve their accuracy and usability, a technique called *Sensor Fusion* that is discussed in Section 2.2.3.

Magnetometer

Magnetometer (also known as compass or magnetic field sensor) is the sensor that measures the strength of the magnetic field surrounding the device. It is usually a three-axis sensor in order to work on all possible device orientations. This sensor is also used in order to compute the actual orientation of the device relatively to the Magnetic North. A 3-axis accelerometer sensor is used to detect the device orientation related to the ground and compute the 2D orientation from the 3-axis of the magnetometer.

The main limitation of this sensor is that it requires a calibration before each usage. In addition, it is very sensitive to metal objects and magnetic fields. Compared to the gyroscope, it is not sensitive to the drift problem as it relies on the magnetic north for its measurements [83].

Both Android and iOS platforms provide access to the magnetometer sensor. In Android, the sampling rate can be set using the same sensor profiles described in Section 2.2.1. Magnetometer also benefits from the *Sensor Fusion* technique, later discussed in Section 2.2.3.

Location

A location sensor is available in most modern smartphones. Depending on the configuration of the sensor, it uses different sources to estimate the location coordinates of the device around the earth's surface, such as the WiFi or Cellular sensor reporting a low accuracy of up to 100 meters, or the Global Positioning System (GPS) reporting an increased accuracy of approximately 3 meters.

The limitation of these localisation sensors is that they only work in outdoor environments as the sensor needs a direct view with at least three satellites to estimate the location. These sensors are also well known for their power inefficiency, as later presented in Section 3.4.

Both Android and iOS provide access to this sensor.

Bluetooth

Bluetooth is a wireless technology standard that is used for short distance communication between devices. This sensor can also be used to find the proximity between devices, by measuring the Received Signal Strength Indicator (RSSI) of every Bluetooth sensor available in range. Depending on the device type and the version of Bluetooth, this range can be from 10 to 400 meters [7].

Since Bluetooth Core Specifications v4.0 it provides a Low Energy (BLE) feature, branded as Bluetooth Smart, that is low cost for consumers and power efficient, allowing a stand-alone device to last longer. Apple developed its own implementation of Bluetooth

Smart, branded as iBeacon™ and presented with iOS 7 in June 2013. Google presented a similar protocol titled Eddystone™ that is open-source and supported in both iOS and Android platforms.

Both iOS and Android provide complete access to the Bluetooth Classic sensor. While the improved Bluetooth Smart version is already available in most iOS devices today, Android only added full support on that sensor with the release of Android Lollipop (v5.0) in September 2014. Apple on the other side, while supporting Bluetooth Smart since iOS 6, they restrict broadcasting custom BLE packets by only supporting their own iBeacon™ protocol. Finally, iOS only allows devices to broadcast as an iBeacon™ while the app is running in the foreground but not in the background.

Ambient Light, Temperature, Humidity and Barometer

These environmental sensors can be used to monitor various environmental properties.

Most smartphones include an Ambient Light Sensor, as it is used by the mobile operating systems to adjust the brightness levels of the screen automatically. Even though Android provides an API for accessing this sensor, iOS doesn't. Only Android provides an API to access the temperature, humidity and barometer sensors when available on a device. Since iOS 8, there is an API to access the barometer Sensor on supported devices.

Proximity

Most smartphones operating with a touch screen use a proximity sensor that detects the presence of nearby objects. It is used by the mobile platform to deactivate the touch screen while the user holds the device next to his/her ear during a phone call. In that way, it avoids accidentally touches to the screen.

Android platform provides full access to this sensor, allowing the developers to know when an object is close to the device, as well as the proximity of the object in cm. In iOS the access is limited, as a developer can only ask the system to auto-deactivate the screen when an object is nearby.

2.2.2 Available Virtual Sensors

These sensors are not actual hardware sensors but can provide useful information to the researchers such as social activity or battery consumption of the device.

Call and Message logs

Even though call and message logs is not an actual mobile sensor, researchers have used these logs to measure the total communication of mobile users, as well as to measure the

communication taking place at specific hours of the day (*e.g.* late in the evening or early in the morning) [48, 37].

Only Android platform provides access to Call and Message logs.

Battery Status

Researchers often evaluate the battery status in order to measure the power consumption of a device while running an application or custom process.

Both iOS and Android platforms provide information about the battery level with 1% resolution. Additionally, Android provides information about the battery temperature in °C as well as the voltage in millivolts (mV).

2.2.3 Sensor Fusion

Sensor fusion is a method that analyses arrays of different sensors in order to produce information that is new or has improved accuracy.

A simple example of a sensor fusion technique applied in smartphones is the combination of the GPS sensor with the magnetometer. As shown before, the magnetometer can show the orientation of the device relatively to the Magnetic North. However, since the True North depends on the location of the device, by combining the location retrieved from the GPS sensor, the direction of the True North can be computed.

Both Android and iOS platforms perform sensor fusion internally, mainly for improving the sensor accuracy. Thus, not only they provide the raw data from the sensors, but also the calibrated data that have been processed with techniques similar to the one presented in Figure 2.1, a sensor fusion technique used by Android platform, as presented by David Sachs in the Google Tech Talk "Sensor Fusion on Android Devices: A Revolution in Motion Processing" [83].

By combining the data of an accelerometer, a gyroscope and a magnetometer, we can minimise the limitations of each sensor. Thus, the gyroscope will not have the well-known drift problem as it will use the accelerometer to fix its measurements. An accelerometer will have much more stable data, using the gyroscope as a filter. The magnetometer can know the vertical orientation of the sensor using the accelerometer and use the appropriate axis to calculate the direction. Finally, a separation of the Gravity and the Linear Acceleration is also possible, with both iOS and Android platforms providing these measurements separately, named *User Acceleration* and *Gravity*.

A common drawback of these sensor fusion techniques is that they require continuous sensor processing, having an impact on the device's battery. Moreover, the double integration that is required for separating gravity from accelerometer is a computational process that highly depends on accurate timing. These limitations overcome with the use

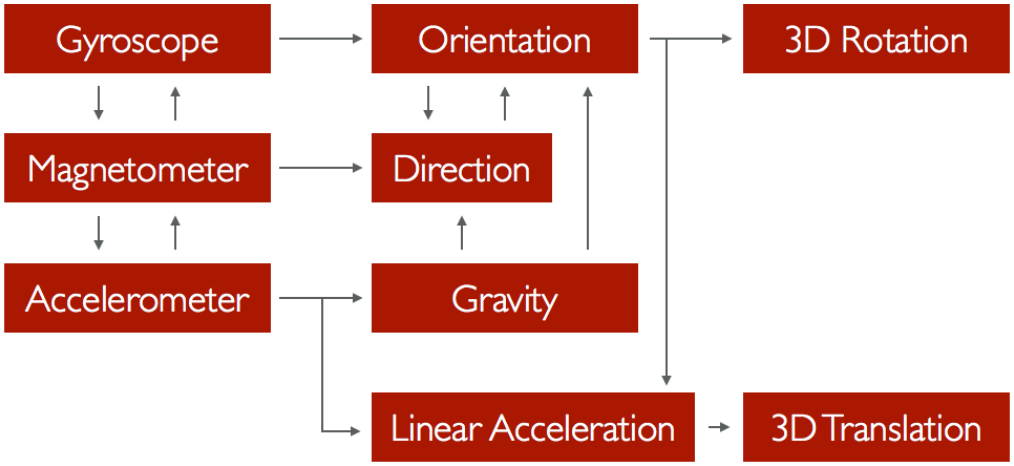


Figure 2.1: Illustration of a Sensor Fusion technique [83].

of motion co-processors.

2.2.4 Motion Co-Processors

Initially launched by Apple in the iOS ecosystem with the release of iPhone 5S, they are separate hardware processors that continuously analyse sensor data from accelerometer, gyroscope, magnetometer and barometer. Since the data process is happening entirely on hardware, it is more accurate and less power consuming. This module also performs motion activity recognition that constantly tracks the user's activity classified as *stationary*, *walking*, *running*, *driving* and *cycling*. Moreover, it tracks the steps taken as well as its pace and cadence. More importantly, an app doesn't need to be active in order to collect these data as all data collection and analysis is happening in the processor's internal memory. Developers can query the analysed data, either on demand (*e.g.* when the user opens the application) or sporadically (*e.g.* daily).

2.2.5 Sensing in the Background

Sensing while the device is not actively used (*i.e.* device is locked, or the data collection app is in the background) is an important requirement for collecting mobile sensor data in research studies as many works explore the daily phone use patterns of groups of people [37, 34, 68].

Past versions of Android mobile system (*i.e.* prior to Android Oreo) did not restrict a process running in the background for any task, as long as it did not overuse the resources of the device. Since Android Oreo (v8.0), Google included some Background Execution Limits[1] adding restrictions in the processes running in the background. More specifically,

[1]https://developer.android.com/about/versions/oreo/background

a window of some minutes (approximately 10) exist from the moment the app moves into the background in which it is allowed to execute background tasks. Similar restrictions were added to the apps querying the user's current location, limiting it to a few times per hour.

Apple iOS systems include similar restrictions, with processes only allowed to run in the background for up to 10 minutes until they are killed by the system. Some exceptions to these restrictions exist for specific services, including the *Location updates* service that allows you to access the device's location or proximity with nearby iBeacon™ sensors (*i.e.* a version of Bluetooth Smart suitable for location tracking). Other services are the *Audio and AirPlay* that allows you to access the microphone sensor in the background, or the *Picture in Picture* that allows you to play a video in a window mode while the user uses other apps. Note that while a background service is running, it is possible to perform additional operations, including sensor data collecting from additional sensors.

2.2.6 Continuous Sensing Platforms

Due to the research opportunities presented with the ubiquity of the smartphones, researchers developed continuous sensing platforms that automate the sensor data collection. Continuous sensing in the context of mobile systems is a technique that a process (usually in the form of a mobile application) automatically collects sensor data in the background, using the device's sensing capabilities. This data can be collected from actual hardware sensors (*e.g.* accelerometer, GPS, microphone, etc.), or virtual sensors (*e.g.* call and messaging logs, battery status, etc.). Researchers have been using these sensors to infer information about the user's location (*e.g.* indoor localisation using Bluetooth or WiFi [3]), motion activity (accelerometer and gyroscope sensor [91]) or context (*e.g.* indoors vs. outdoors using several sensors including light, microphone, proximity and cellular [77]). Continuous sensing platforms can collect data from multiple sensors, save it to the device's memory and transmit it to a remote repository for future analysis. Examples of such frameworks are MobiSens [98], EmotionSense [75], Funf [1] and AIRS [93].

MobiSens [98] is a complete sensor data collection system that consists of a mobile framework, capable of accessing mobile sensor data, and a server back-end responsible for data collection and pre-processing. This work is not available in open-source.

EmotionSense [75] is an open-source project from University of Cambridge that is focused on inferring the user's emotion state. It can be configured to recognise specific activities using a low sampling rate and only when the activity is recognised (*e.g.* when the user is walking), activate the desired sensors for data capturing in higher sampling rate. This is an application of opportunistic sensing [44], discussed in Section 2.2 that results in an improved power efficiency of the system.

Table 2.1: Continuous Sensing Platforms

Sensor	AIRS	EmotionSense	Funf	MobiSens	SensingKit
Accelerometer	No	Yes	Yes	Yes	Yes
Battery Status	Yes	Yes	Yes	Yes	Yes
Bluetooth	Yes	Yes	Yes	Yes	Yes
Bluetooth Smart	No	No	No	No	Yes
Call / Message Logs	Yes	Yes	Yes	No	No
Camera	No	Yes	No	No	No
Gyroscope	No	Yes	Yes	Yes	Yes
Location	Yes	Yes	Yes	Yes	Yes
Magnetometer	Yes	No	Yes	Yes	Yes
Microphone	Yes	Yes	No	Yes	Yes
Proximity	Yes	Yes	Yes	No	Yes
Sensor Fusion	No	Yes	No	No	Yes
WiFi	Yes	Yes	Yes	Yes	No
Open Source	LGPL 2.0	ISC	LGPL 3.0	No	LGPL 3.0
Supported Platforms	Android	Android	Android	Android	Android & iOS
Local Storage	Yes	Yes	Yes	Yes	Yes
Remote Storage	Yes	Yes	Yes	Yes	No

MIT Media Lab also presented their own continuous sensing platform called Funf [1]. It can capture sensor data either locally to the device's memory, remotely to a server or save them to a remote Dropbox account. It is worth mentioning that this library is not supported any more.

Android Remote Sensing (AIRS) [93] is one of the most complete sensing systems in terms of sensor support, with a wide range of internal mobile sensors being available such as microphone, magnetometer, GPS, WiFi, Bluetooth, battery status, call log and proximity. In addition, it supports external sensors connected to the device (*e.g.* heart rate monitors).

Last but not least, our new continuous sensing library titled *SensingKit* [28] supports both Android and iOS platforms and is available at https://www.sensingkit.org. It is discussed in Chapter 3.

Table 2.1 illustrates a comparison of the continuous sensing platforms described above.

2.3 Crowd Sensing

This section presents previous works that use mobile sensing technology to measure the dynamics of the crowd. More specifically, it presents some example works that detect the user mobility and continues with more related to this work research areas that focus on detecting the social interactions of moving and stationary groups.

2.3.1 Detecting Mobility

Calabrese *et al.* [9] investigated the crowd mobility using one-million anonymous cell-phone traces from an area of 15 × 15 km within Boston. The aim was to understand the relationship between events (*i.e.* event type) and origins of people (*i.e.* distance from home). Such approaches that investigate the crowd mobility with call-logs have the disadvantage of low accuracy when estimating the user location. Moreover, data are only accessible through mobile providers, requiring an active connection from the user (*i.e.* make a call, receive an SMS, maintain a data connection etc.).

Wirz *et al.* [94] used GPS location traces to measure the crowd dynamics and visualise them in real time through a web-interface. They measured the crowd density, velocity, turbulence and pressure in order to raise an alert when critical situations exist in crowded areas. The novelty of this research is that it does not require access to CCTVs and expensive vision analysis techniques. However, it depends on the availability of a GPS sensor on each pedestrian's smartphone.

Another work that uses GPS location and Bluetooth scan traces was conducted in Roskilde Festival in Denmark [89]. They captured spatio-temporal data from 155 participants, by distributing a participatory sensing mobile application that enables Bluetooth scanning in each device. Larsen *et al.* [45] conducted a similar study in the same festival. Instead of distributing an application, they installed 33 Bluetooth Scanners in strategic locations. They captured spatio-temporal traces from 5,127 people during the eight-day festival. Finally, they applied community detection algorithms to identify groups of people and their musical preferences. While these works are capable of detecting communities, the use of Bluetooth sensing alone as a mean of mobility is not capable of detecting the social interactions of each community in high granularity.

Virtual Compass from Banerjee *et al.* [4] is a peer-based relative positioning system capable of accurately detecting user mobility from mobile devices. It uses multiple radios (*i.e.* WiFi and Bluetooth Classic) to detect nearby devices and place then into 2-dimensional space. To reduce the error produced from the RSSI-based proximity estimation, it uses measurements from multiple devices simultaneously and applies an iterative algorithm that refines the estimated coordinates until the algorithm converges. They evaluated the system with 10 devices (including smartphones and laptops) that used WiFi and Bluetooth interfaces. Results reported an average error of 3.4 meters when using the Bluetooth sensor, 3.91 when using the WiFi sensor, and 1.41 when using both sensors.

2.3.2 Detecting Walking Social Interactions

Grouping people into clusters with the use of accelerometer data and gait analysis has recently been of interest due to availability of high-frequency 3-axis accelerometers in

smartphones and wearables. Roggen *et al.* [82] suggested a framework that uses an on-body accelerometer sensor installed on each user's hip as a simulation of a phone placed in the user's trouser pocket. By analysing the time-series accelerometer data, they were able to identify pedestrian flocks of up to three members.

Garcia-Ceja *et al.* [20] have used low-frequency accelerometer (5Hz) and WiFi data to detect when two individuals are walking together. They analysed the WiFi RSSI signal from the in-range access point as an indication of co-location, and the accelerometer signal as an indication of similarity in movement. In a case study with seven participants for seven working days, they reported a 77.2% precision and 90.2% recall performance.

Kjærgaard *et al.* [40] presented a work for identifying pedestrian flocks using smartphones equipped with accelerometer, WiFi and magnetometer sensors. Using sensor-fusion techniques, they developed a method that extracts specific features from sensor data such as variance, turn, indoor location etc., and then applied classification on every extracted feature. Even though they reported a high accuracy of 87%, the evaluation depended on pre-scripted scenarios of 16 participants. Furthermore, the use of the WiFi sensor makes this method only applicable in controlled environments with installed WiFi hotspots.

Wirz *et al.* [96] proposed an on-line method for detecting pedestrian flocks using GPS traces. Even though the results were very promising with an accuracy of 91%, the method only works on outdoor environments. Moreover, the GPS is considered one of the most power expensive sensors in mobile systems [32], having a noticeable impact on the device's battery life.

Computer vision has also been applied into the area of modelling the pedestrian behaviour. One example is the Social Force Model (SFM) that was initially proposed by Helbing and Molnar [23] as a method for simulating pedestrian behaviour. It is a mathematical model that describes the dynamics of an individual based on social forces produced by other people or obstacles around the space. Based on the theory of this model, each person is driven by several attractive and repulsive social forces that affects his/her trajectory in the space. A force can be attractive, towards another individual if these are within a group that know each-other, or repulsive, if they are strangers.

Even though the SFM was originally intent and is widely used in crowd simulations, Sochman and Hogg [87] presented an original work that used an inverted version of the SFM to detect walking social interactions. This work uses the pedestrian's coordinates as an input in order to calculate the trajectories of each individual. By applying an iterative algorithm based on the principle of the SFM, they tested its performance in real-world scenarios and outperformed other related works in the computer vision area. The disadvantage of this method is that it requires the absolute coordinates of each participant to be known, and thus, it is mostly applicable in automated computer vision techniques.

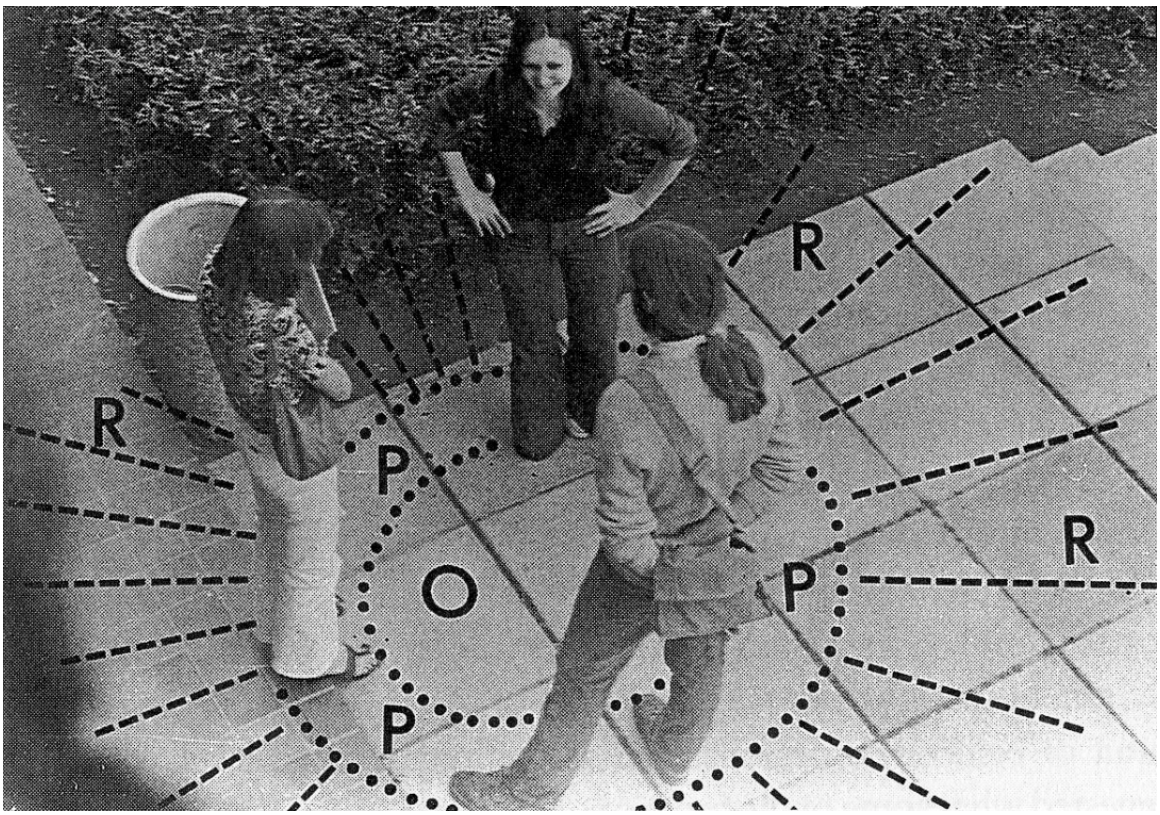


Figure 2.2: Example of F-formations in real-world. (Kendon, 1990)

More recently, Mazzon *et al.* [53] presented an enhanced version of this algorithm that uses a buffered graph-based tracker that improves the accuracy up to 13% in group detection.

All previously mentioned works only focus on the social interactions of walking pedestrians, excluding stationary interactions that are more likely to happen in planned events such as social gatherings or networking events.

2.3.3 Detecting Stationary Social Interactions

Kendon [38] was the first that introduced the idea of an F-formation to categorise the casual face-to-face conversations. He defined an F-formation as "two or more people cooperate together to maintain a space between them to which they all have direct and exclusive access" [38]. Each individual has a 'transactional segment' that others respect. The overlapping segment from two or more people is called 'O-space' and is the common space that people share during the conversation. Surrounding the O-space, the 'P-space' exists which is the area that the participants are location. Finally, the 'R-space' is the area that doesn't belong to the F-formation. Figure 2.2 shows an example of an F-formations in real-world [38].

An F-formation has some specific characteristics that make it a very interesting categorisation of social interactions:

- A distance always exists between participants.
- People normally respect the 'transactional segment' of each-other.
- Every time a new person joins the formation, everybody re-adjusts their position so that they all have direct access with each-other.
- People rarely cross the 'O-space' when leaving an F-formation.

Hung & Krose's [26] work used the term of F-formation to characterise a face-to-face interaction. In their study they used information about the proximity between people, as well as the body orientation to identify interactions with an accuracy of 92%. In a similar research, Cristani *et al.* [15] suggested a system that also detects F-formations using information about people's position and head orientation. They reported comparable results of 89% accuracy. Both works assume that the information of related proximity between participants or absolute position in the space, as well as the body or head orientation is known, either using manual annotations or computer vision techniques.

One of the first attempts to identify stationary, face-to-face interactions in an automated way is the *Sociometer* by Choudhury and Pentland [14], a wearable device that can be placed on each person's shoulder and identify other people wearing the same device using Infrared (IR) sensors. In addition, it is equipped with an accelerometer sensor to capture motion as well as a microphone to capture speech information. During the system evaluation, Sociometer was able to identify social interaction with an accuracy of 63.5% overall and 87.5% for conversations lasted for more than one minute.

Matic *et al.* [52] presented a solution based on the RSSI of the WiFi sensor as a way of estimating the proximity between people, and the embedded magnetometer to extract the standard deviation of the relative body orientation, as an indication of the position stability between participants. Finally, by placing an external accelerometer device into each user's chest they analysed the vibrations produced by the user's vocal chords and detected speech activity. They evaluated their approach in an office located study with 4 participants for 7 working days, achieving an performance of 89% true positives and 11% false negatives in detecting existing social interactions.

Montanari *et al.* [57] created a wearable device named *Protractor* that uses near-infrared light to monitor the user proximity with a mean error of 2.3 - 4.9 cm, and relative body-orientation with a 6° error 95% of the time. The device was evaluated in a group-collaborative task where 64 participants split into groups of four were asked to collaborate with each-other to build a construction made of spaghetti and plastic tape. Using a supervised machine learning approach based on Random Forest, they achieved an 84.9% accuracy when detecting the individual's task role (*i.e.* the verbal role of each participant) and 93.2% accuracy when identifying the task timeline (*i.e.* the building

phases of the collaborative task). Note that even though the evaluation of this work is focused on the social behaviour of an existing group that is interacting, Protractor could also be used to detect stationary interactions within crowds by using the estimated proximity and relative orientation between participants.

Palaghias *et al.* [65] presented a real-time system for recognizing social interactions in real-world scenarios. Using the RSSI of Bluetooth Classic radios and a 2-layer machine learning model, they classified the proximity between two devices into three interaction zones, based on the theory of Proxemics [22]: *public*, *social* and *personal*. In addition, they used an improved version of *uDirect* research [24] that utilises a combination of accelerometer and magnetometer sensors to estimate the user's facing direction with respect to the earth's coordinates. This work reported results of 81.40% accuracy for detecting social interactions, with no previous knowledge of the device's orientation inside the user's pocket. However, this work has been evaluated in a limited dataset with eight participants while an observer was keeping notes that were later used as ground truth. Moreover, it is only able to detect one-to-one social interactions using a specific device model (HTC One S) and has not been evaluated in scenarios of interactions with dynamic sizes.

2.4 Summary

Smartphones and their wide range of embedded sensors inspired a wealth of research opportunities. Researchers have developed continuous sensing platforms that automate the sensor data collection. However, these frameworks are currently limited to work on Android platform, limiting the sampling space of users participating in different studies. Moreover, they are not up-to-date with current sensing technologies such as the use of Bluetooth Smart (BLE) for proximity estimation or indoor localisation.

The availability of mobile sensing technology in modern smartphone devices enabled researchers to explore crowds in an automated way, without the need for additional wearable equipment or computer vision systems. Mobile sensing-based solutions are also easier and more cost efficient to deploy in unknown or new spaces as they only rely on the users' own hardware. Early systems that use mobile sensing report reasonable results, but primarily focus on detecting one-to-one social interactions. Furthermore, their evaluation was conducted in controlled only environments, a situation that only covers a subset of the formations that occur in a natural setting. Finally, they rely on pre-trained models that only work with specific devices.

The work presented in the following chapters will address the aforementioned gaps of the literature review: Firstly, SensingKit continuous sensing framework is presented in Chapter 3, an open-source framework for iOS and Android environments that is capable of accessing sensor data from modern mobile devices. Moreover, a method for detecting

stationary social interactions within crowds is presented in Chapters 4 and 5. This method is capable of detecting interactions in dynamic groups of variety of sizes and is not device dependent. In contrast to other related works, it has been evaluated in a natural, non-artificial social setting.

Chapter 3

SensingKit – A Multi-Platform Mobile Sensing Framework

3.1 Overview

The ubiquity of smartphones as well as the variety of their on-board sensors have enabled the automated acquisition of large-scale data, inspiring a wealth of research opportunities. Mobile operating systems such as Android and iOS provide application programming interfaces (APIs) to access these sensors. However, developing software that collect sensor data for each mobile sensor is a time-consuming task that involves multiple steps such as initialising the sensor, requesting a permission from the operating system to access it, configure its sampling rate, as well as implement a data collection solution that works as a background process even when the user does not actively use the device.

As previously discussed in Section 2.2, various mobile sensing frameworks have been designed that provide continuous sensing, like MobiSens [98], EmotionSense [75], Funf [1] and AIRS [93]. However, these platforms are currently limited to work on Android, or the currently discontinued Nokia Maemo platform, limiting the sampling space of users participating in different studies. Since Android and iOS are the two main players in the mobile ecosystem, there is a clear need for supporting continuous sensing in these two mobile environments. A new framework that has a shared API, identical to iOS and Android versions, could benefit the research community by allowing easy development and distribution of mobile apps for collecting and analysing mobile sensor data.

In 2014, an early prototype of SensingKit framework was released as part of this Ph.D. research [28]. SensingKit is a continuous sensing framework that, in contrast to other existing frameworks, it is compatible with both iOS and Android platforms. It supports most sensors available in a modern smartphone device, being capable of capturing motion, orientation, location, proximity between devices as well as environmental sensor data. It includes a shared, almost identical for iOS and Android platforms API, for accessing the

sensors of the device. Since the two operating systems are equipped with sensor fusion techniques, both raw measurements and fused data like Linear Acceleration and Gravity are supported. Furthermore, SensingKit can also be configured to capture user's natively-labelled activity in supported devices, classified as *Stationary*, *Walking*, *Running*, *Driving* and *Cycling*.

Beside the multi-platform characteristic, SensingKit has some unique features that are not available in other sensing libraries. It fully supports the Bluetooth Low Energy (BLE) specification, branded as Bluetooth Smart (v4.0), for capturing the proximity between devices or other Bluetooth Smart enabled beacons. This has significantly reduced power consumption and a higher sampling rate compared to the classic Bluetooth. At this moment, it supports Apple's iBeacons™, as well as the Google Eddystone™ beacons. These are protocols developed by Apple and Google respectively, that allow a device to broadcast its presence to nearby devices. The receiver can estimate the proximity of the beacon based on the Received Signal Strength Indicator (RSSI) combined with the broadcast *Measured Power* level, the beacon's signal strength measured in 1-meter distance. That feature makes beacon technology extremely useful for indoor localisation systems, allowing smartphones to estimate their approximate location in indoor environments where GPS signal is not available.

In order to avoid timing issues when the user, or even when the device itself changes the system time, the timing in the sensor measurements depends on the device's CPU time base register rather than the system's clock. The library also makes use of the devices motion co-processor for its motion activity recognition sensor, having only a minimum effect on the device's battery life. Finally, it utilises all sensor fusion technologies that are available into the two operating systems, providing calibrated and accurate sensor data.

This chapter presents the latest version of SensingKit framework (v0.5) for both iOS and Android platforms. The objective in this work is to provide an easy-to-use sensing framework that developers and researchers can use to provide continuous sensing in iOS and Android applications. Section 3.2 presents the System Architecture and technical details of the framework. Section 3.3 presents CrowdSense, a cross-platform mobile app designed for research experiments that utilises SensingKit and is capable of collecting sensor data into the device's memory. The battery consumption of SensingKit was evaluated, as reported in Section 3.4, by using CrowdSense installed on an iPhone 5S mobile phone. Further evaluation was conducted in a pilot study that investigates the subconscious phenomenon of gait synchronisation between individuals, reported in Section 3.5. A summary of this chapter is presented in Section 3.6.

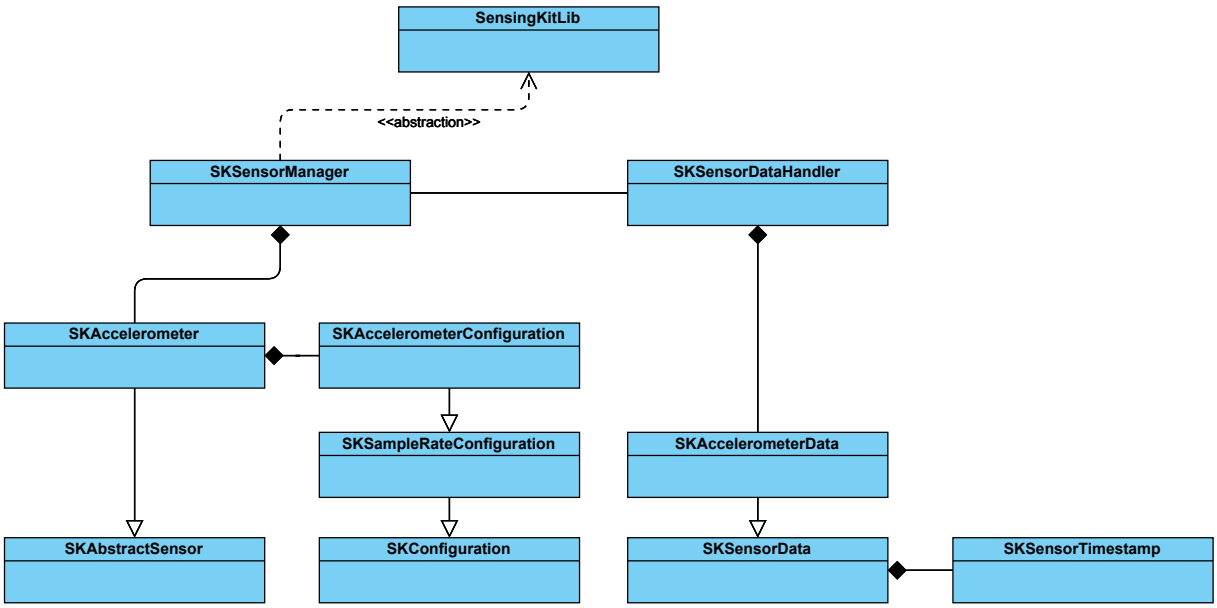


Figure 3.1: SensingKit System Architecture. Accelerometer sensor was used as an example for Sensor, SensorData and SensorConfiguration classes.

3.2 SensingKit Platform

SensingKit is a modular mobile framework developed in the native programming language of each platform (Java for the Android and Objective-C for the iOS version). It supports mobile devices running iOS 9 and Android Jelly Bean (v4.1) and above. At this moment, that corresponds to over 95% of all iOS and over 99% of all Android devices available today[1].

Figure 3.1 gives an overview of the system architecture in a Unified Modelling Language (UML) class diagram. For every sensing category, a sensing module (*e.g.* SKAccelerometer) exists in SensingKit, as well as a corresponded configuration (*e.g.* SKAccelerometerConfiguration) and a data object (*e.g.* SKAccelerometerData). Each sensor module provides access to the corresponding sensor inside the device whereas a configuration object initialises the sensor with custom configuration (*e.g.* custom sample rate or accuracy). When new sensor data is available, a data object is generated that represents the sensor data in CSV or JSON format. SensingKitLib is the interface that developers need to use in order to check for sensor availability inside the device, initialise and configure a sensor, provide the block function that will be called each time new sensor data is available, and finally start or stop continuous sensing operations. SKSensorManager is the module that implements SensingKitLib interface and performs all required operations to the sensor modules such as memory allocation and deallocation, sensor configuration etc. Due to the modular design of this library, it is easy to develop a new module and extend its sensing capabilities. Table 3.1 presents the available sensing modules of the

[1]As reported by Apple App Store and Google Play Store on April 22, 2018.

Table 3.1: SensingKit Supported Sensors

Sensor	Apple iOS	Google Android
Accelerometer	Yes	Yes
Altimeter	Yes	Yes
Ambient Temperature	-**	Yes
Battery Status	Yes	Yes
Bluetooth® Classic	-	Scanning only
Eddystone™ Proximity	Scanning only***	Yes
Gravity	Yes*	Yes
Gyroscope	Yes	Yes
Heading	Yes	No
Humidity	-**	Yes
iBeacon™ Proximity	Yes	Yes
Light	-**	Yes
Linear Acceleration	Yes*	Yes
Location	Yes	Yes
Magnetometer	Yes	Yes
Microphone	Yes	Yes
Motion Activity	Yes	Yes
Pedometer	Yes	Yes
Rotation	Yes*	Yes
Screen Status	No	Yes

*In SensingKit-iOS, these sensors are part of the *Device Motion* sensor.

**These sensors are either not available or access is not allowed on iOS due to Apple's restrictions.

***Broadcasting an Eddystone™ beacon signal is not allowed on iOS due to Apple's restrictions.

framework.

For proximity sensing, SensingKit uses Bluetooth Smart (4.0) proximity profile that allows to broadcast a device's presence, scan for other devices and most important, estimate the distance between them using the Received Signal Strength Indicator (RSSI). Bluetooth Smart is only fully supported in Android Lollipop (v5.0) mobile operating system. Android Jelly Bean (v4.3) devices are only limited to scan and connect to other devices (*Observer and Central mode*) and not to advertise its presence to the nearby devices (*Peripheral mode*). Due to Apple restrictions applied to the iOS operating system, it is not possible to broadcast a Bluetooth Smart signal with custom service data blocks, making it impossible to broadcast an Eddystone™ signal from an iOS device.

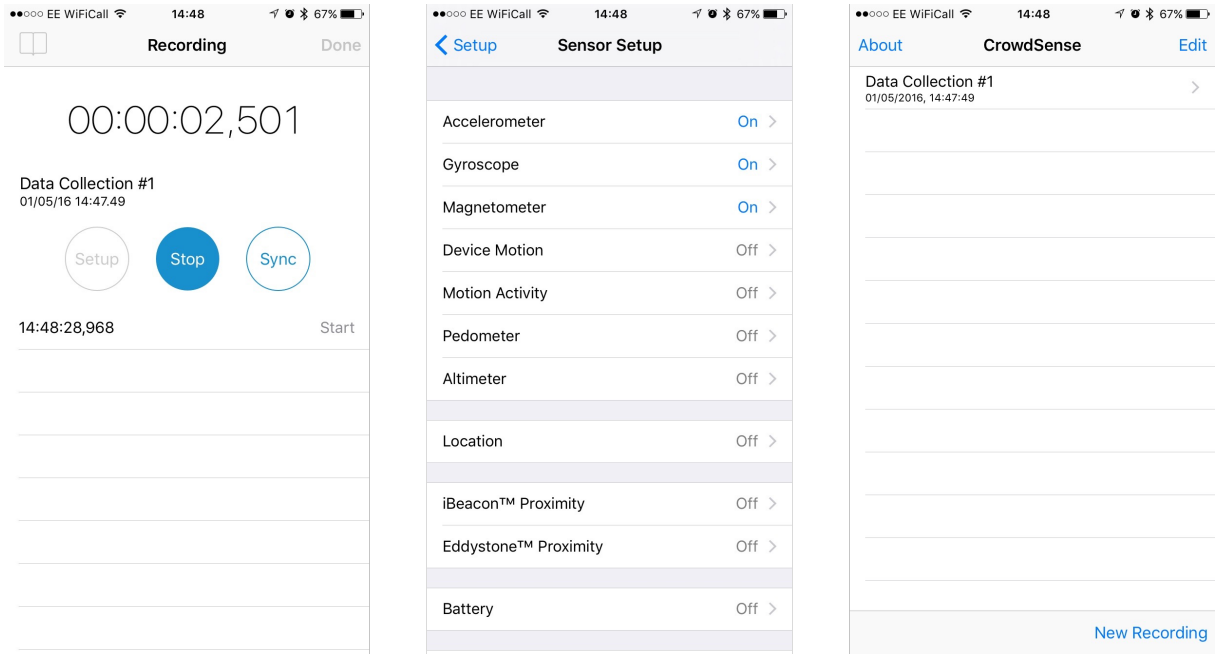


Figure 3.2: CrowdSense App for iOS

3.3 CrowdSense App

CrowdSense is a cross-platform mobile app for iOS and Android applications based on SensingKit. It was built with the aim to support the experiments of this Ph.D. research. It supports all sensors available in SensingKit library, when these are available and supported by the device. It can be run in all iOS smartphones with version iOS 9 or greater, and all Android smartphones with version Jelly Bean (v4.1) or greater. The application runs as a background service, collects the sensor data and save them into the device's memory in CSV format.

Since iOS is quite restrictive when running an app in the background, CrowdSense App for iOS will only collect data in the background if specific sensors that allow this behaviour are enabled. These sensors are the *location* and the *microphone* sensors, with the appropriate requested permissions accepted by the user. When at least one of those sensors is collecting data, the app will continue collecting data from any other enabled sensor.

Since Android Oreo (v8.0), Google added similar restrictions on what apps can do while running in the background, only allowing a process to run for a few minutes before the system kills it automatically. Since this research is more focused on the iOS platform as a tool for experimentation and data collection, the Android version of this app is not as advanced as the iOS, lacking support for several sensors and features. Moreover, the restriction of data collection while the app is in the background has not been fully addressed for devices with Android Oreo or greater. A new release of CrowdSense App for Android is under development and planned to be released in the close future.

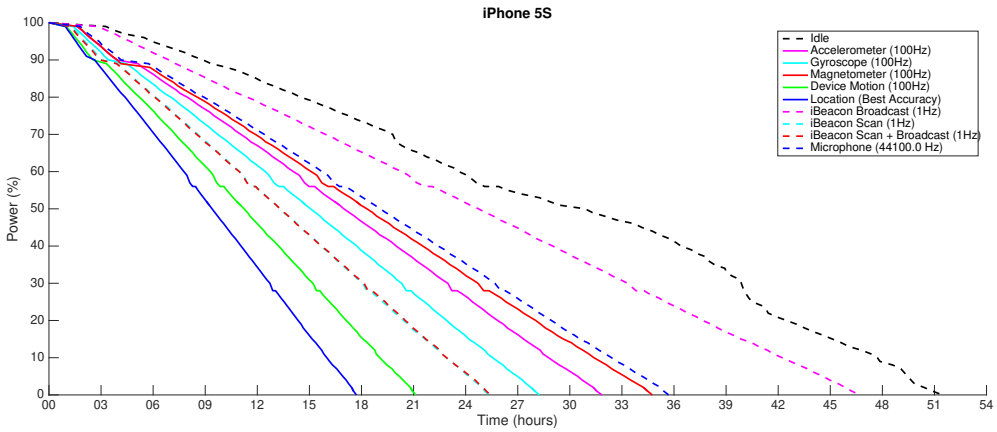


Figure 3.3: Battery consumption of SensingKit running on an iPhone 5S.

Table 3.2: Device Specification

Model	iPhone 5S
Storage	32 GB
Operating System	iOS 9.0.2
Processor	1.3 Ghz Dual-core
Memory	1GB LPDDR3
Battery	1560 mAh
Bluetooth	4.0
Purchased	September 2013

The complete source-code of the two apps is available in the SensingKit GitHub repository at https://github.com/SensingKit. Both iOS and Android versions of this app are available for free in the Apple App Store and Google Play Store.

3.4 Evaluation of Battery Consumption

The battery life performance was measured while using SensingKit in an iPhone 5S device running iOS 9.0.2 (Table 3.2). The device was fully erased and restored into the manufacturer's default settings. No other third-party applications were installed or running in the background. The device was set to "Flight Mode", having WiFi, Bluetooth and Cellular connectivity disabled. Finally, the Background App Refresh setting was set to "Off" and the Low Power Mode to "On", in an attempt to minimise the impact that the operating system has on the device's battery life.

Figure 3.3 and Table 3.3 show the energy consumption of SensingKit running on the mobile device described above. More specifically, they present the consumption of the library while using the accelerometer, gyroscope, magnetometer, device motion (fused motion and orientation data), location (GPS), iBeacon™ and microphone sensors. In the case of iBeacon™ sensor, the two modes (*i.e.* Broadcasting and Scanning) are evaluated

Table 3.3: Battery consumption of SensingKit in iOS

Sensor	Sample Rate	Hours Lasted
Idle	-	51.27
Accelerometer	100 Hz	31.51
Gyroscope	100 Hz	28.15
Magnetometer	100 Hz	34.45
Device Motion	100 Hz	21.07
Location	Best Accuracy	17.42
iBeacon Broadcast	1 Hz	46.43
iBeacon Scan	1 Hz	25.21
iBeacon Scan & Broadcast	1 Hz	25.26
Microphone	44100.0 Hz	35.41

separately, as well as a combination of both of them together. In addition, the "idle" mode of the library is visualised that represents the power consumption of the library while only senses the battery status.

Figure 3.3 shows that all measurements follow a similar pattern in regard to the battery level drop. More specifically, all sensors had a slower drop between levels 100% and 98%, a faster battery drop between 98% and 90% and a more linear pattern between 90% to 0%. Two bumps in the measurement curves can also be observed at the levels of 56% and 28%. Since these effects are present in all sensor measurements at the same battery levels, they can be explained as small battery defects due to the daily phone use.

The location (GPS) sensor in "Best Accuracy" mode is the most power expensive sensor of all, as the device only lasted for 17.42 hours compared to the "idle" mode that lasted for 51.27 hours. GPS sensor is well known for its extensive power consumption, not only because it receives signal from multiple satellites simultaneously in order to estimate the devices distance from them, but also because of the expensive trigonometric operations (trilateration) that is performing in order to estimate the device's position on the surface of the earth.

From all motion and orientation sensors, magnetometer is the one that performed best, with the device lasting for 34.45 hours in 100 Hz sampling rate. Even though newer iPhone models combine accelerometer and gyroscope on the same unit (*e.g.* Bosch Sensortech BMI160 for iPhone X) and could result in similar battery performance between the two sensors, iPhone 5S include a separate unit for accelerometer (Bosch Sensortech BMA220) and gyroscope (STMicroelectronics). Thus, accelerometer sensor configured at 100 Hz sampling rate lasted for 31.51 hours, whereas gyroscope, known as a relative expensive sensor, lasted for 28.15 hours, using the same sampling rate. Device motion sensor was the most expensive of all motion sensors, lasting for 21.07 hours. The reason is that this sensor is using a combination of accelerometer, gyroscope and magnetometer in order to provide calibrated and more accurate data using sensor fusion techniques

performed entirely on hardware.

Recording audio using the microphone sensor lasted for 35.41 hours, despite its high sampling rate of 44100.0 Hz.

Evaluating the iBeacon™ sensor in the three different modes explained above showed interesting results. While the sensor was set in the "broadcast" mode, the device lasted 46.43 hours, highly comparable to the "idle" mode (51 hours). More interestingly, there were only 5 minutes difference between the "scan" and "scan and broadcast" modes, as the devices lasted 25.21 and 25.26 hours respectively. That proves that broadcasting an iBeacon™ signal has almost no effect on the device's battery, while scanning for other iBeacon™ devices is quite expensive. The reason is that iOS not only scans for the presence of other devices but is also "ranging" in 1 Hz sampling rate in order to estimate the other beacon's proximity based on the RSSI explained above.

Note that Figure 3.3 only represents the battery consumption on the specific mobile device listed in Table 3.2 and should only be viewed as a comparison between the available sensors rather than an indicator of each sensor's power consumption.

3.5 Pilot Study: Analysing Pedestrian Behaviour

It is well known that crowds have several common behavioural patterns. For example, pedestrians inside a crowd tend to move together as a unit; with group members walking at the same speed, following the same trajectories and quickly reform after they become separated [64]. An interesting phenomenon in small groups is the synchronization of stepping that often occurs in people walking side by side. This mutual engagement can either occur intentionally (*e.g.* a public procession that occurs by imitation) or unintentionally when that happens subconsciously, a behaviour known in psychology as "mirroring" [11]. Previous studies suggest that the reason behind an unintentional sync in walking is that they share a common feeling of unity and close relationship [42].

In most cases, a group walk also includes engagement with a conversation. People tend to look at each other while walking, interact with gestures and at the same time have a turn-taking conversation. According to Moussaid *et al.* [59], group members usually have a V-shaped walking formation that facilitates social interactions between members. According to Battersby *et al.* [5], gaze and head orientation becomes problematic in conversations between more than two participants. The reason is that the gaze can only focus on a single person at a time, especially when the participants are walking side by side. These non-verbal social signals were expected to influence the walking patterns and synchronization of these walking groups.

Modern technology such as mobile sensing can be used to explore these kinds of group behaviours. Accelerometer sensors such as the ones embedded in modern smartphones

can be used to analyse the walking patterns of people engaged in a conversation.

Due to importance of movement coordination in interpersonal communication, there have been a vast number of studies in this space by different methods such as observational studies or use of video-based gait analysis. Analysis of walk synchronization has been of interest in psychology [81, 11] and robotics [56], as well as for behavioural analysis of individuals [60]. In the latter study, paired individuals were observed to have synchronous walking rhythm and gait while having a phone conversation.

The work presented in this section not only focuses on the physical act of walking, but on specific gait analysis and walk synchronization. High-quality 3-axis accelerometer data was used at a minimum sampling rate of 90Hz in order to accurately capture the gait synchronization between two individuals, in addition to studying the effect of the third person on the synchronization. Understanding the phenomenon of gait synchronization between pedestrians that belong to the same group is relevant to the aim of this thesis as it can be used as a predictor in systems aiming to detect pedestrian flocks using mobile sensor data. Moreover, it is important for crowd-sensing and potentially security applications in scenarios which could be explored in the continuation of this work.

The findings also empirically support those of Richardson *et al.* [81] and Shockley *et al.* [86], who have used posture sway to understand the interpersonal coordination in the context of a cooperative verbal task, showing that *conversation* is responsible for such coordination. The way in which a conversation potentially causes locomotion coordination amongst two individuals remains debated amongst scholars.

3.5.1 Experiment

This study analyses the walking behaviour of pedestrians existing in a group of two or three people. It focuses on the synchronization behaviour that occurs when a group of people is engaged in their walking activity.

This section starts with a detailed description of the experiment including the materials used, the participants, the experimental design and the procedure followed.

Materials

CrowdSense App for Android was installed in three Samsung Galaxy S2 Android smartphones. All devices were updated to the latest Android version officially supported by the manufacturer (Jelly Bean v4.1.2). CrowdSense was configured to only record accelerometer sensor data at the fastest sampling rate possible. A short audio sample of 10 seconds was also recorded and used to synchronize the sensor data between the three devices. Table 3.4 shows the three smartphones used in this experiment, with the average sampling rate for capturing accelerometer data.

Table 3.4: Device Specification

Participant	Type	Op. System	Sampling Mean (SD)
P1	Samsung Galaxy S2	Jelly Bean 4.1.2	91.93Hz (2.15)
P2	Samsung Galaxy S2	Jelly Bean 4.1.2	90.90Hz (2.37)
P3	Samsung Galaxy S2	Jelly Bean 4.1.2	91.64Hz (2.23)

Table 3.5: Demographic Information

Participant	Gender	Height	Weight
P1	Male	1.80m	75kg
P2	Male	1.72m	76kg
P3	Male	1.86m	89kg

Participants

The experiment involved the recruitment of three participants (P1, P2 and P3). The participants were students from Queen Mary University of London and had no previous knowledge of the current study. Table 3.5 shows the demographic information of the recruited participants.

Experimental Design

As stated above, this experiment focused on the syncing behaviour of people walking in groups of two (Scenario 1) and three people (Scenario 2). For that reason, the participants were assigned to the groups listed in Table 3.6. The idea behind this separation is that in Scenario 2, Scenario 1 should be repeated with another participant (P3) injected between the participants of the initial group.

Procedure

The experiment took place in the Mile End Park, an open and usually not crowded park in East London. All three participants were informed that this experiment will aim to understand the walking behaviour of people and received a short demonstration of CrowdSense application. Finally, each participant did a short walk, with the mobile device placed on the left pocket.

In order to facilitate their conversation during the walk, each group was asked to decide on a preferred conversational topic, chosen from a list of 25 subjects. This also helped

Table 3.6: Groups Configuration

Scenario	Group	Participants
S1	G1	P1, P2
S2	G2	P1, P3, P2

the group to concentrate on the conversation and not on their walking activity. The two groups walked for 12 minutes in the park. The group was observed from a distance of approximately five meters while notes were kept using a voice recorder.

All data collection and analysis was made with informed consent and approved by the Queen Mary University of London research ethics committee (Appendix A).

3.5.2 Results and Discussion

Before starting the data analysis, all data were synchronized using the audio sample described in 3.5.1. Using this technique, an accuracy of ±50ms was achieved. Table 3.4 shows the mean and standard deviation (SD) of the sampling rate of the three devices when using the accelerometer sensor. Since a smartphone is not a real-time system, the requested sampling rate (fastest in our case) is only a suggestion to the system, with the actual rate being rather unstable. Thus, data was interpolated to 100Hz using a linear interpolation method. Finally, the magnitude (resultant acceleration) of the 3-axis accelerometer sensor was computed using the formula $\sqrt{x^2 + y^2 + z^2}$.

Figure 3.4a shows a five second period of walking activity, performed by P1. In this periodic signal, the high peaks represent the acceleration produced by the left leg while doing one step (x = 16, 137, 255 and 372) whereas the lower acceleration peaks are from the stride while the other leg is doing another step. The duration of every step (produced by the same foot) can be calculated from the distance between two high-peaks (1.21 sec). This repeated pattern is also visible in the autocorrelation analysis of the complete walk, as shown in Figure 3.4b. The figure indicates that the walking rhythm is a periodic pattern with features that reflect on the walking activity of the individual.

The higher correlation between two participants implies that their steps are also synchronized. Bivariate correlations of the acceleration signals between the participants in Scenario 1 show a positive correlation (Pearson Correlation = 0.23, $p < 0.01$). Similar but smaller correlations exist when comparing the signals of Group B: 0.02 ($p < 0.01$) for P1 and P2, 0.11 ($p < 0.01$) for P1 and P3, and 0.03 ($p < 0.01$) for M2 and M3. The p values in this context represents the probability that this result would exist if the correlation coefficient was in fact zero (null hypothesis). The effect can be visually identified in the following heat map plot (Figure 3.5) that presents the combinations of all bivariate correlations between the recorded data. In the same plot, it is obvious that data from two different walks (*e.g.* P1 of Scenario 1 with P1 of Scenario 2) report very low correlations.

Similar correlation values can be identified in Figure 3.6, where a cross-correlation was performed with a maximum lag of 1 second. The maximum correlation between P1 and P2 in Scenario 1 (Figure 3.6a) is 0.24 at lag = 0.01s, higher than all correlations between participants in Scenario 2 (Figure 3.6b, c and d). Smaller positive correlations appear in Scenario 2, with 0.05 at lag = -0.6s (P1 and P2), 0.08 at lag = -0.14s (P2 and P3), and

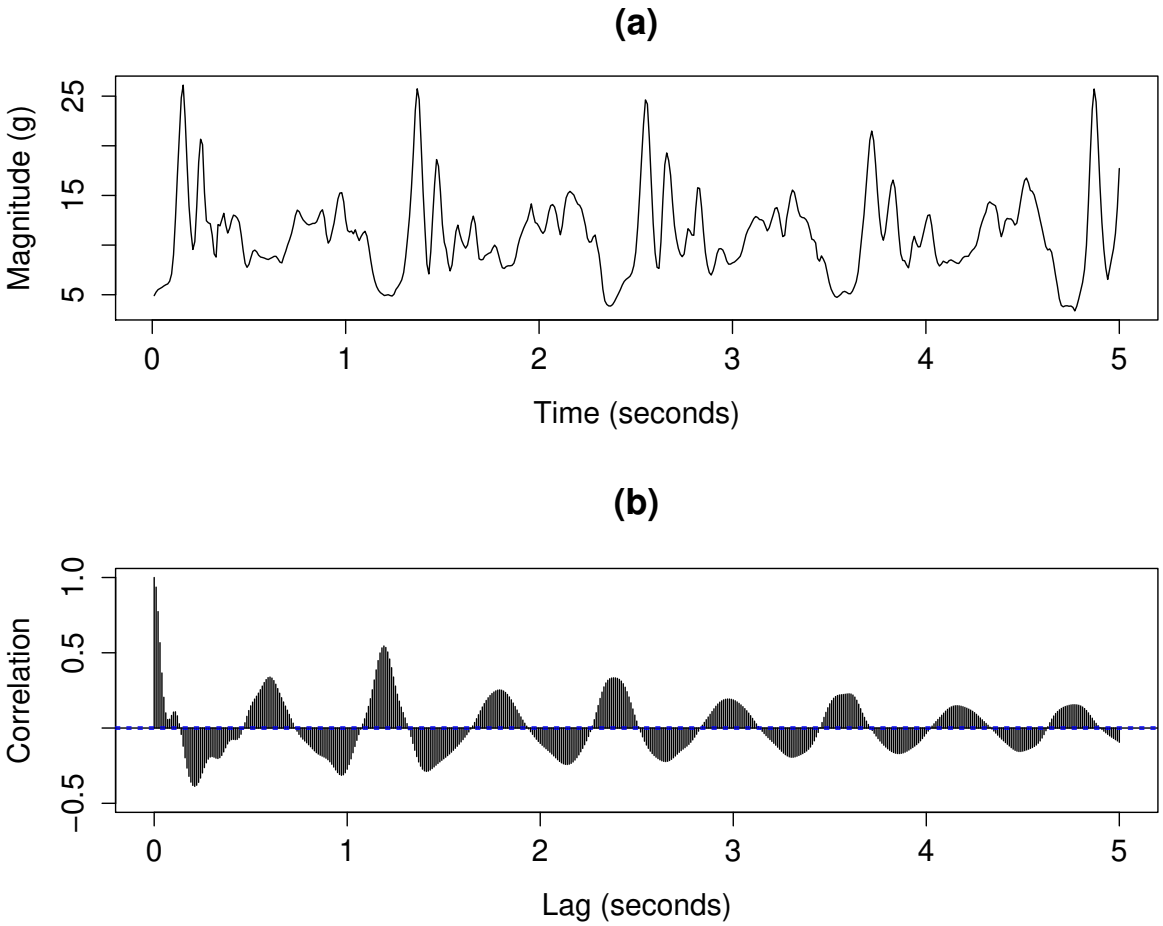


Figure 3.4: Acceleration Magnitude of 5 sec (a) and Autocorrelation (b) of Participant 1 in Scenario 1. The dotted blue lines are a 95% confidence interval, indicating that if the assumption that a correlation at this lag exists holds, 95% of the time the correlation will lie within this interval.

0.13 at lag = -0.03 (P1 and P3). This indicates that the two participants in Scenario 1 have the highest correlation and with the minimum lag in their stepping (lag = 0.01s). Since both analyses P1 × P2 and P2 × P3 show much smaller correlation and high lag in the sync, it can be assumed that P2 is the least synchronized person of the group.

The same synchronization was observed visually by the researcher in Group A during the experiment, but only in cases were two of the people were discussing together while the other was not paying attention to the conversation. A likely explanation of this correlation is the engagement of the conversation that affects the participants to synchronize their steps unconsciously. According to Lakens and Stel "*The tendency to synchronize movement rhythms has been theorized to play an important role in the formation of a social unit*" [42]. When three people are walking together, the conversation is taking place in turns between two people, changing their gaze so that each person looks at each other. This influences their walking, making their steps to correlate again in each turn

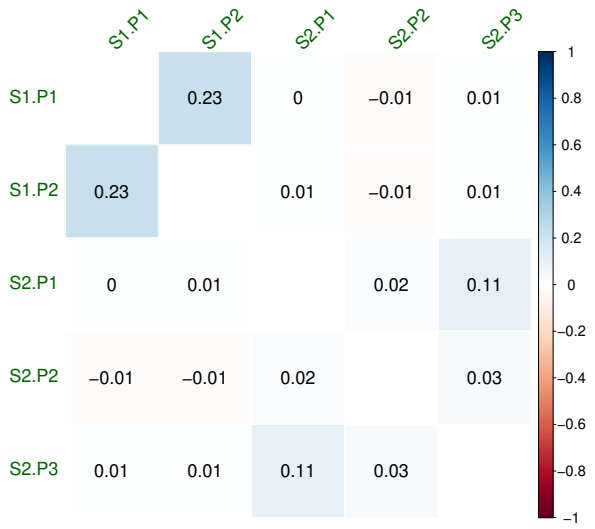


Figure 3.5: Pearson Correlation matrix of all 5 datasets, visualized as a heat map plot.

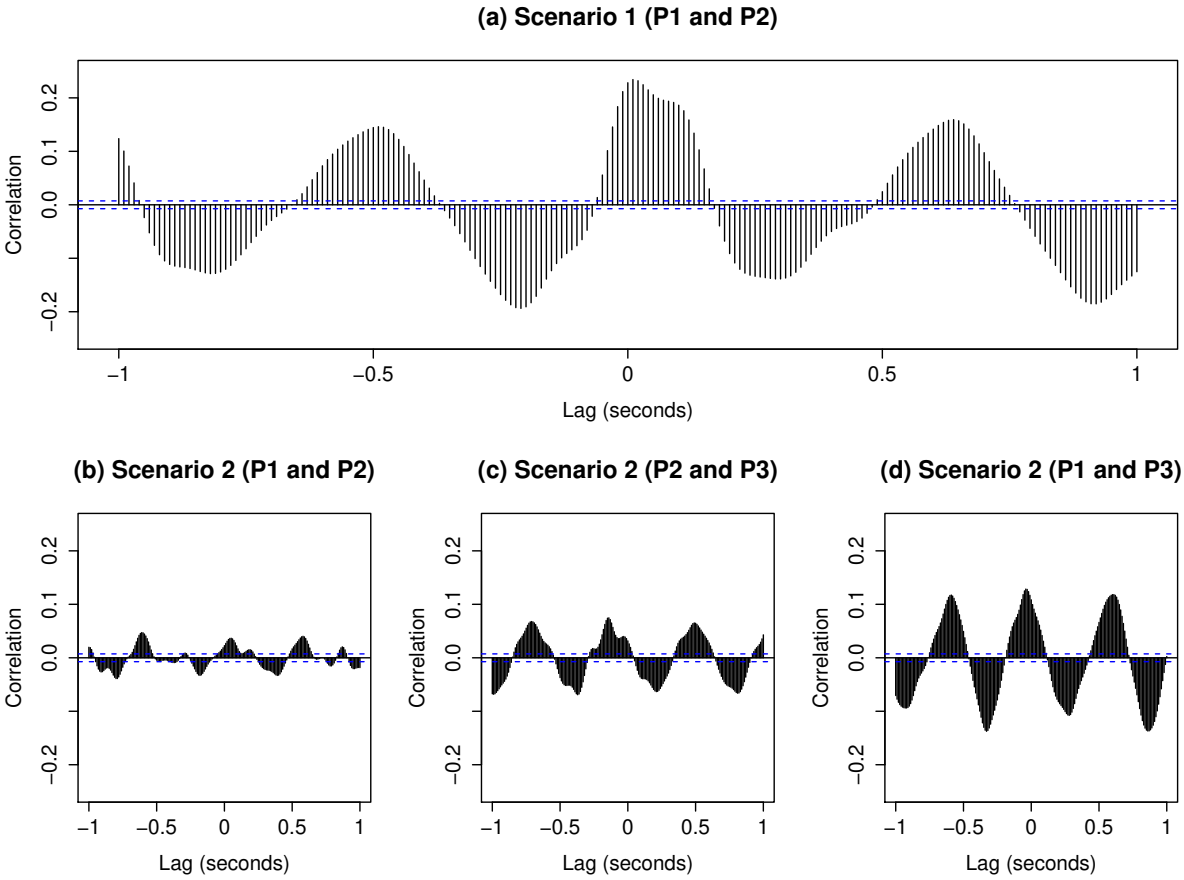


Figure 3.6: Cross-correlation analysis of Scenario 1 (a) - 2 people walking, and Scenario 2 (b, c, d) - 3 people walking in a park for 12 minutes. The dotted blue lines are a 95% confidence interval, indicating that if the assumption that the pair is in sync holds, 95% of the time the correlation will lie within this interval.

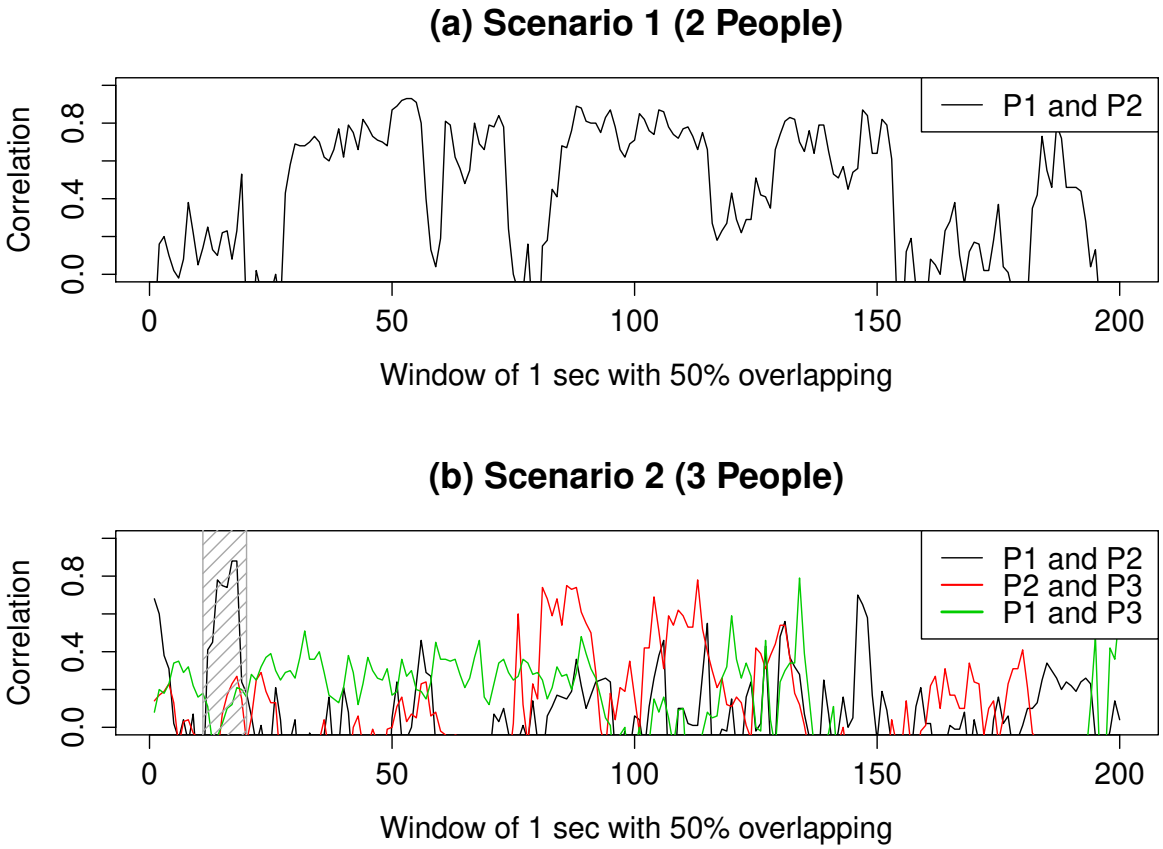


Figure 3.7: Pearson Correlation applied to windows of 1 sec (with 50% overlap) for Scenario 1 (a) and Scenario 2 (b).

and break the synchronization between the participants of the past turn.

This effect is also clear in Figure 3.7, where the correlation for every second is plotted for both Scenario 1 and Scenario 2. In Figure 3.7a, the two people are not synchronized at windows 1 to 27, but become in sync after window 28. In Figure 3.7b it is obvious that there is a weaker synchronization between the people in the group. There are moments that P1 and P2 are in sync (window 12 - 19) and only a few moments that all three people look synchronized.

3.5.3 Conclusion

Interpersonal communication through voice, gestures, and gait is an essential part of the human evolution. Through analyses of subconscious actions such as gait synchronization and gestures, it is possible to infer rich information about individuals' relationships. Availability of highly sensitive smartphones enables us to capture these interactions with high accuracy.

In this work the accelerometer data from individuals walking in different formations

was analysed, while having conversations in groups of two or three. High frequency data from the gait motions were processed and cross-correlated. The results empirically confirm previously published observatory studies: the non-verbal social signals such as the gaze, head orientation and gestures between individuals play a significant factor in gait synchronization between two individuals walking together. A third person being involved in this formation can distort the synchronization, unless they are not actively engaged in the conversation between a pair.

One of the limitations of this study is the restricted number of participants. Even though the walks that the participants made per scenario was relative long (approximately 12 minutes), repetition of this study with more participants is suggested in a continuation of this work. In addition, more fine-grained evaluation with video recordings is required to understand why and how the participants are synchronising with each-other.

These results and methodology can have a number of benefits for different disciplines. They can be beneficial for scientists studying human interaction, or for organizations interested in crowd sensing, or inferring individuals' relationships. For example, this can be a useful predictor in systems aiming to detect pedestrian flocks using mobile sensor data.

3.6 Summary

In this chapter, a continuous sensing system was presented. A first prototype of this system was presented in ACM MobiCom'14 Poster Session and is available in open-source under the GNU Lesser General Public License v3.0. Compared to other mobile sensing frameworks, SensingKit works in both iOS and Android platforms using the same API. It supports modern mobile sensing technologies such as the use of Bluetooth Smart (BLE) for proximity estimation and indoor localisation, or the use of motion co-processors for sensing motion, orientation or motion activity. Moreover, it uses the device's CPU time base register rather than the system's clock as a sensor data timestamp to avoid timing issues when the system clock changes unintentionally. Finally, it utilises all sensor fusion technologies that are available into the two operating systems, providing calibrated and accurate sensor data.

The performance of each sensor in regard to its battery consumption was evaluated for some of the most commonly used sensors in research studies. That includes the evaluation of accelerometer, gyroscope, magnetometer, device motion, location, iBeacon (BLE) and microphone sensors using SensingKit-iOS version running on an iPhone 5S device.

For the purpose of this research, a mobile application that utilises SensingKit framework was designed, titled *CrowdSense*. The application is capable of capturing data from all sensors supported by SensingKit and save them into the device's memory in Comma-

Separated-Values (CSV) format. It is available for free in Apple App Store and Google Play Store since August 2015. More information about SensingKit as well as the complete source-code is available at https://www.sensingkit.org.

A pilot study was conducted that evaluates SensingKit framework as a tool capable of conducted research studies. More specifically, the study investigated the subconscious phenomenon of gait synchronisation between individuals while they walk in a natural environment in groups of two and three. This work is a first step towards detecting whether two individuals belong in the same walking group, also known as pedestrian flock. This can be particular useful in detecting relationships between individuals or detecting the group formation and numbers for crowd-sensing applications.

The next chapter will present a study that focuses on detecting stationary interactions. More specifically, it will evaluate current mobile sensors available in smartphones for estimating proximity between mobile devices. This will lead into the development of the DetecTN algorithm, an approach for detecting stationary interactions happening in social events.

Chapter 4

Towards Detecting Social Interactions

4.1 Overview

Over the years, there have been many attempts for detecting social interactions automatically, primarily from video. Most of the initial works are either based on manual annotated videos [26, 15] or use computationally expensive computer vision techniques [87, 6] that rely on external CCTV camera surveillance. Other approaches include custom made wearable hardware [14, 57] that include advanced sensors (*i.e.* infrared light) and report increased accuracy, but at the same time can be expensive and not easy to scale in larger environments. With the rapid rise in the variety of available smartphones and their wide range of embedded sensors, researchers have the opportunity to explore social interactions in an automated way that depends entirely on the use of mobile sensing technology [52, 65], without the need of additional wearable equipment or computer vision systems that can also have implications with the user's privacy. Moreover, it is easier and cost efficient to deploy in unknown or new spaces as it relies on people's own hardware.

While most early systems that use mobile sensing report reasonable results, they have only been focused on and evaluated in detecting one-to-one social interactions, a situation that only covers a subset of the formations that occur in natural environments. Furthermore, they rely on pre-trained models that only work with specific mobile devices. This chapter presents an initial study aiming to detect stationary social interactions of various sizes using mobile sensing technology.

The first experiments evaluate whether a system that automatically detects stationary interactions of various sizes could be feasible with current technology that is available in a modern smartphone. More specifically, an evaluation of the Bluetooth Smart (BLE) sensor is presented in Section 4.2, as a way of estimating the proximity between two people. Three methods are evaluated, all based on the Received Signal Strength Indicator (RSSI)

of the Bluetooth Smart sensor and how the signal propagates through the environment; the Path Loss Model (PLM) [51], a pre-trained model from Radius Networks [76], and finally what Apple reports as proximity in their implementation of Bluetooth Smart known as iBeacon™ [2].

Resulting from the findings of this experiment, Section 4.3 proposes an initial group interaction study that evaluates a novel algorithm, named DetecTN, for detecting stationary interactions inside crowds. It uses information extracted from the Bluetooth Smart sensor, as an indication of the user proximity, as well as the device's accelerometer sensor, as an indication of the user's motion activity (*i.e.* Stationary vs. Moving). Unlike previous work in mobile crowd sensing, the DetecTN algorithm is capable of detecting dynamic groups of variety of sizes and is not device dependent. The findings report a performance of 80.9% precision and 92.4% recall while detecting interactions second-by-second.

Finally, the implications of this preliminary work are discussed in Section 4.4, alongside the reasoning behind the development of a successor algorithm presented in Chapter 5.

4.2 Validation Experiments

One of the most important predictors when detecting social interactions is obviously the proximity between people. There have been several ways of estimating the distance between devices using wireless sensors such as *Time of Arrival*, *Time Difference of Arrival*, *Angle of Arrival* and using the *Received Signal Strength Indicator (RSSI)*. At this moment, the only method that is available in today's smartphones is using the RSSI of either the Bluetooth or the WiFi sensor.

In the past, researchers have used the RSSI of Bluetooth [46, 25, 65], WiFi [51] or even a combination of them [3] by measuring the RSSI of every wireless sensor available in range and comparing it with a *Measured Power* constant that indicates the signal strength (in dBm) at a known distance (usually 1m). In 2010, the Bluetooth Special Interest Group released Bluetooth v4.0 with a Low Energy feature (BLE) that was branded as Bluetooth Smart [2]. Bluetooth Smart is low cost for consumers, has low latency in communications (6 ms) and is power efficient. Moreover, it supports a low energy advertising mode where the device periodically broadcasts specially formatted advertising packets to all devices in range with a customizable sample rate of approximately 3Hz. This packet includes 31 bytes of information that developers can use to advertise, for example, a unique ID for each user, but also the measured power constant that was mentioned above. The advantage of using this technology for proximity estimation is that each manufacturer can configure the device to use its own pre-calibrated measured power constant, making the proximity estimation more accurate. In addition, devices do not need to maintain a connection with each-other in order to measure the RSSI, having a minimum impact on the

device's battery life. In its latest version, marketed as Bluetooth 5 [7], the sensor provides additional benefits including longer range (x4) and longer capacity in the advertising packet (x8).

Apple developed a closed-source protocol based on Bluetooth Smart, branded as iBeacon™ and supported it as of iOS 7 in June 2013 in all mobile devices with Bluetooth 4.0 or greater (iPhone 4 or newer) [2]. The specification of iBeacon's advertising packet includes: a 16 byte *Universally Unique Identifier (UUID)* used to separate beacon applications, two 2-byte unsigned integer identifiers named *Major*, that separates beacon groups (*e.g.* located on the same venue or floor), and *Minor*, that separates individual beacons within the group, and a 1 byte *Measured Power* value used as an RSSI reference at 1-meter distance. The remaining 9 available bytes are used as a static prefix and cannot be customised by the developer. An iOS application can register to monitor for beacons of specific UUIDs and estimate its proximity whenever a beacon exists within range. The app can also advertise iBeacon™ packets, however, only while the device is unlocked (*i.e.* in-use while the screen is on) and the app remains in the foreground. Furthermore, it is not possible to customize the Broadcasting Power of the Bluetooth sensor, using the maximum power by default. Note that even though iBeacon™ is an Apple product for iOS devices, it is possible to scan or broadcast as an iBeacon™ from Android platform using third-party libraries such as the Android Beacon Library from Radius Networks[1].

In this section, iBeacon™ protocol is evaluated as a way of proximity estimation between smartphone devices. An experiment was conducted using two iPhone devices. By using SensingKit iBeacon™ Proximity sensor in *Scan & Broadcast* configuration, proximity data was collected (RSSI and Accuracy[2]) for five minutes in 12 distances from 0.00m to 3.00m, every 0.25m. It is important to mention that each device was both broadcasting and receiving an iBeacon™ signal, resulting into a total of 300 measurements for each device per distance. The experiment took place in the Performance Lab of Queen Mary University of London, an empty room 10.6×8.16 meters long that is mainly used for a variety of performance research and recording applications. Figure 4.1 shows the RSSI patterns over distance, as measured by the two iPhone devices.

The figure demonstrates that the pattern of the RSSI signal is device hardware dependent (iPhone 5S vs. iPhone 6S) and proves the requirement of a measured power constant as a reference point for a pre-known distance as explained above. It also shows the instability and fluctuation of the RSSI signal, mainly because of environmental factors (*e.g.* reflections, obstacles, background noise) as reported by other similar works [51, 46]. The same experiment was repeated at the Mobile Antenna EMC Screened Anechoic Chamber of Queen Mary University of London, a $6 \times 5 \times 3$ meters chamber designed

[1] https://github.com/AltBeacon/android-beacon-library
[2] Apple's proximity estimation.

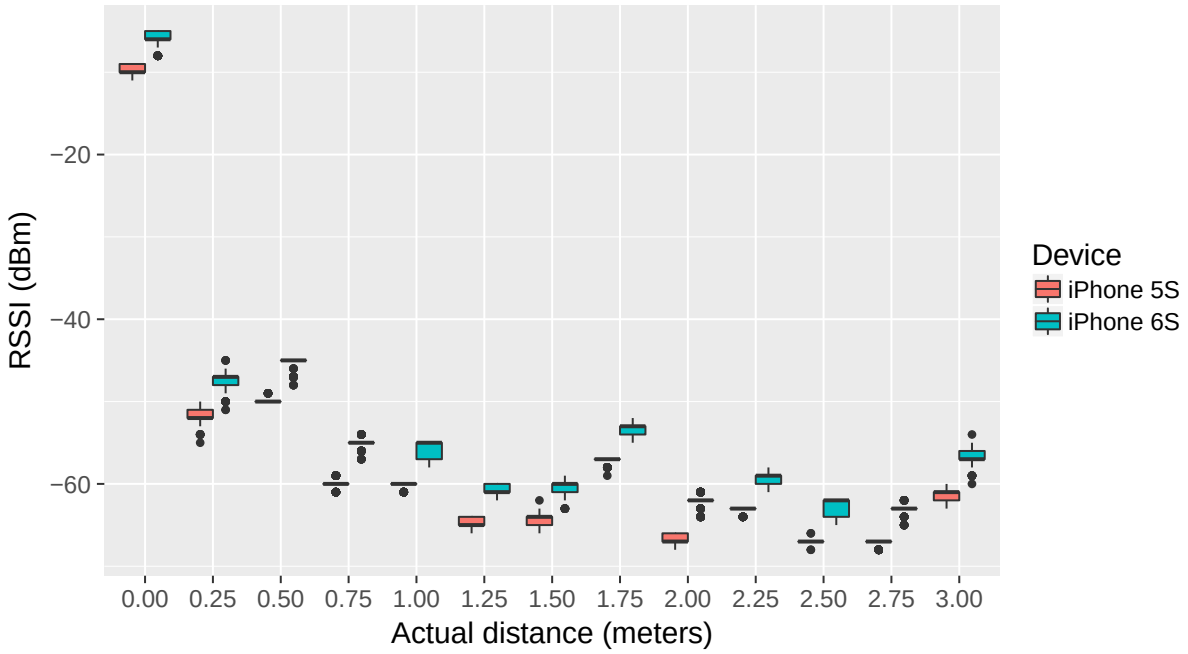


Figure 4.1: Box plot of Bluetooth Smart RSSI versus distance using two iPhone devices.

to completely absorb reflections of electromagnetic waves. Two metallic tripods were used to hold the two devices. Because of their reflecting material they both contributed to the multi-path components. Although the pattern of the received RSSI in both environments is similar, the actual captured value in the chamber is lower by approximately 5-7 dBm which highlights the reduction in reflections and multi-path components. In addition, the similarity in the patterns could be attributed to the receiver's own automatic gain control.

Different methods have been evaluated for estimating distance using the Bluetooth Smart RSSI and the Measured Power constant (-57 for iPhone 5S and -56 for iPhone 6S) as recorded by the iBeacon™ Proximity sensor. The Path Loss Model (PLM) [51] was tested, based on the theory of signal loss as it propagates through space:

$$P(d) = P(d_0) - 10 \times n \times log_{10}(d) - X, \tag{4.1}$$

where $P(d_0)$ is the measured power (in dBm) at 1-meter distance, n the path loss exponent, d the distance in which the the RSSI is estimated and X a component that describes the path loss by possible obstacles between the transmitter and the receiver. The values $n = 1.5$ and $X = 1$ were chosen as the environment was indoors and no obstacles between the two devices were expected when a group of people is interacting with each-other. The same model can easily be transformed in order to estimate the distance (d) from the RSSI ($P(d)$), as shown in the following formula:

$$d = 10^{\frac{P(d_0)-P(d)-X}{10\times n}} \tag{4.2}$$

A second model was tested, provided by Radius Networks [76], a company that produces beacon technology solutions compatible with both iBeacon™ and Eddystone™:

$$r = \frac{P(d)}{P(d_0)} \tag{4.3}$$

$$d = \begin{cases} r^{10}, & \text{if } r < 1.0 \\ 0.89976 \times r^{7.7095} + 0.111, & \text{otherwise} \end{cases} \tag{4.4}$$

According to Radius Networks, *"The three constants in the formula (0.89976, 7.7095 and 0.111) are a best fit curve based on a number of measured signal strengths at various known distances from a Nexus 4"* [76].

Finally, the two methods have been compared with iBeacon's distance estimation (reported as *Accuracy* by Apple). As iBeacon™ technology is closed-source, the details of this method are unknown. One of the limitations when using Apple's estimation is that a delay of 20 seconds is observed every time the distance changes, indicating that Apple's method normalises the RSSI signal through a rolling mean function before estimating the distance. Even though that reports a more stable measurement, it is less responsive and not suitable for this work that aims to evaluate the interactions in windows shorter than 20 seconds.

Figure 4.2 shows the accuracy of the distance estimated with the three methods described above. Since two RSSI measurements exist (from Phone A to Phone B and from Phone B to Phone A), the model was applied twice and the average of the two distances was used. The results indicate a low accuracy in all three models, especially at the distance greater than 1.25 meters. Apple and PLM models reported similar results, with an average estimated error of 0.65 ($SD = 0.44$) and 0.64 ($SD = 0.48$) meters accordingly. Radius model showed slightly better results with an average error of 0.53 ($SD = 0.45$) meters overall. Interestingly, the estimated proximity at 2.25m is the most accurate of all measured proximities, achieving a low error of approximately 0.1 meters for PLM, 0.2 meters for Apple's and 0.4 for Radius's model. It would have been expected that this accuracy should exist at 1 meter, the distance that was used as a reference in the Measured Power constant. These outliers can be attributed to the reflections and multi-path components of the environment, as mentioned earlier in this section.

Results from this experiment suggest that all three approaches on proximity estimation report similar accuracy. It is hard to estimate proximity with good accuracy, especially with distances above 1 meter. In the next section, proximity based on Radius model will be evaluated as a predictor for social interactions.

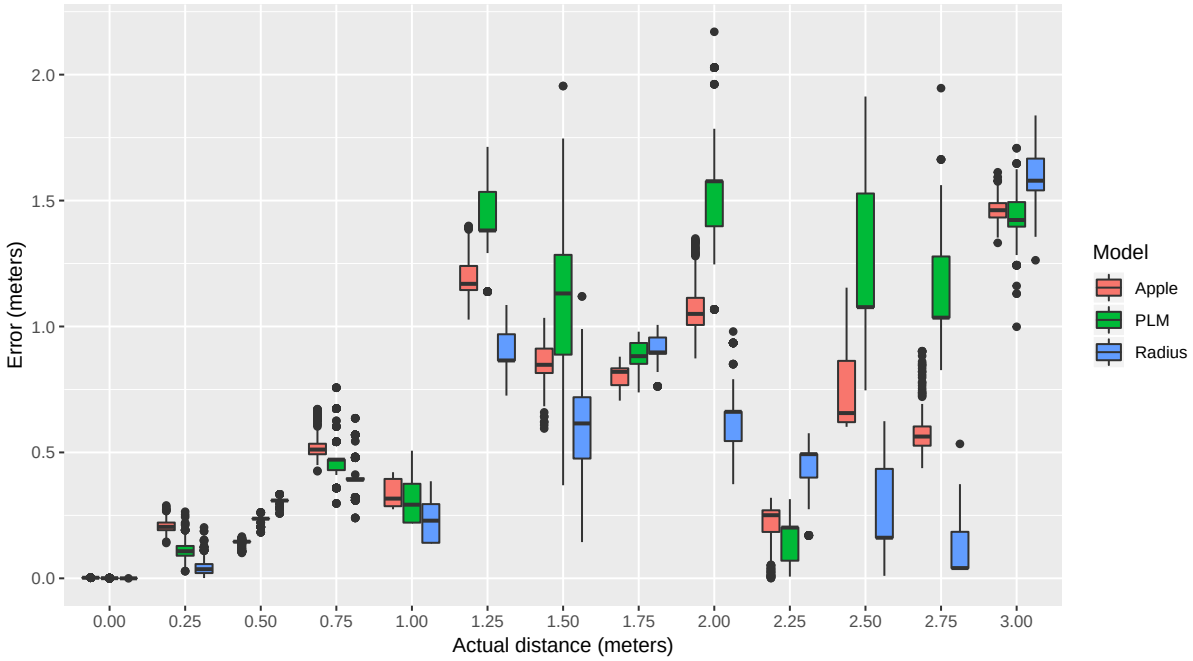


Figure 4.2: Evaluation of Apple's iBeacon™, PLM and Radius distance estimation models.

4.3 Preliminary Study in Detecting F-formations

This study aims to automatically detect social interactions taking place in social events, depending entirely in sensors available in a modern smartphone. The definition of a social interaction that is used throughout this section is inspired from the theory of conversation clusters or Kendon's *F-formations* [38]:

> *An F-formation system arises when two or more people cooperate together to maintain a space between them to which they all have direct and exclusive access.*

In contrast to previous related works, this approach has been evaluated in scenarios where interactions of more than two people are being involved. This section starts with a detailed description of the experiment, including the participants (4.3.1), the sensor data collection (4.3.2) and a description of the procedure that was followed (4.3.3). It continues with an explanation of the ground truth used in this study (4.3.4), the data pre-processing procedure that was followed (4.3.5), and a description of the DetecTN algorithm (4.3.6). The section concludes by presenting the results of this study (4.3.7).

4.3.1 Participants

The experiment involved the recruitment of six participants, all research students from Queen Mary University of London that had no previous experience with the current study.

Table 4.1: Participant demographic information.

	Gender	Height	Weight	Device	iOS
P1	Male	1.80m	74kg	iPhone 6S	9.1
P2	Male	1.88m	63kg	iPhone 6	8.3
P3	Male	1.85m	72kg	iPhone 6	9.1
P4	Female	1.72m	97kg	iPhone 6	9.1
P5	Female	1.64m	59kg	iPhone 5S	9.1
P6	Male	1.90m	95kg	iPhone 5S	9.0.2

Table 4.1 shows the demographic information of the recruited participants as well as the iPhone device and iOS version they used in the experiment.

4.3.2 Sensor Data Collection

As a sensor data collection tool, the open-source CrowdSense App for iOS v1.2.2 was used, presented in Section 3.3. CrowdSense is an iOS application based on SensingKit [28] framework. It provides access to all sensors supported by SensingKit, when these are available in the device. The application collects sensor data and saves them into the device's memory in CSV format. It can be run in all iOS devices with iOS 8 or greater and is available for free in Apple's App Store.

As shown in Table 4.1, the experiment used one iPhone 6S, three iPhone 6 and two iPhone 5S devices, installed with CrowdSense and configured to collect Motion Activity, Device Motion (that includes Linear Acceleration) in the maximum sampling rate of 100Hz and iBeacon™ Proximity data in *Scan & Broadcast* mode. In order to make sure that this experiment can easily be replicated in large scale with unknown devices, the default Measured Power value of iBeacon™ Broadcast was used, as defined by the manufacturer, avoiding a manual re-calibration. A unique ID per device was set into the *Minor* identifier of the sensor, in order to uniquely identify each participant. A short audio sample of 10 seconds was also recorded and used to synchronize the sensor data between the six devices.

To overcome the restriction of Bluetooth Smart sensor not broadcasting a beacon signal while the device is locked, participants had to place the device in their pockets while the screen is on. By using the device's proximity sensor, the app was automatically switching the screen off and avoid the battery consumption produced by the screen.

4.3.3 Procedure

The experiment took place at the Performance Lab of Queen Mary University of London. The Performance Lab is a large space (10.6×8.16) suitable for live performance experiments (Figure 4.3). A DMX lighting rig is also installed in the ceiling of the space, useful for installing Lights, Cameras or other related equipment. Two full-HD video cameras

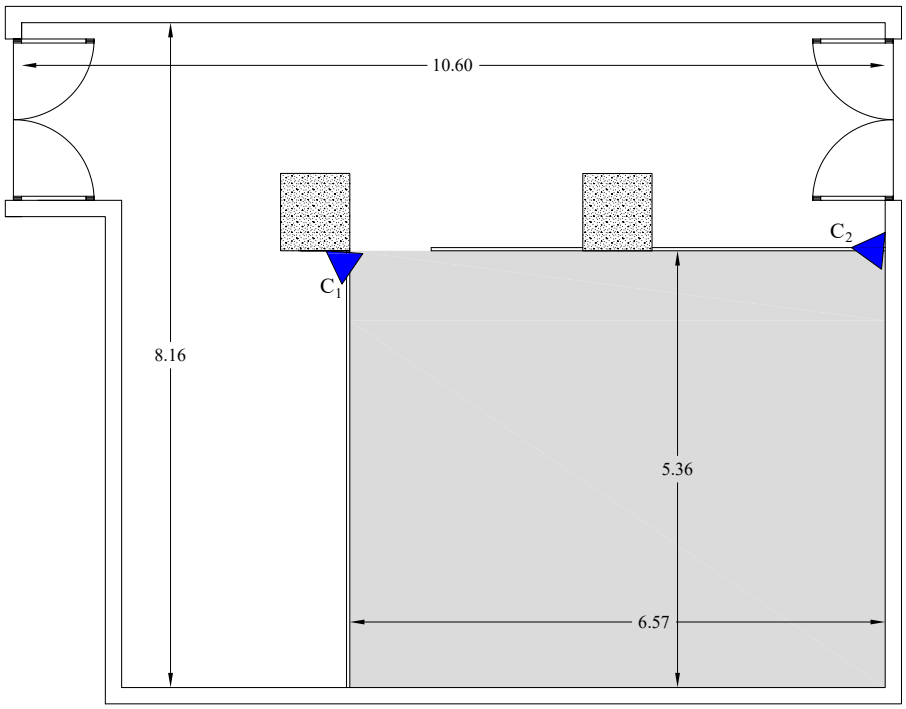


Figure 4.3: Floor plan of the Performance Lab. The area that the study took place is highlighted in grey. The two cameras (C_i) are highlighted in blue. The area that the study took place is highlighted in grey.

were installed in the lighting rig, used to record the experiment from two different angles. This video recording was used as ground truth for evaluating the accuracy of the detection of the interactions happening inside the room.

All six participants were informed that the experiment explores the ways people interact in social events and received a short demonstration of CrowdSense application. They signed the required consent forms and completed a short questionnaire with demographic questions (*i.e.* gender, weight, height, iPhone model and iOS version). During the experiment, one of the researchers gave instructions based on a predefined script about the social groups that they should create (*e.g. "P1 should now leave group A and join group B."*). In order to facilitate their social interaction and make it as real as possible, participants were asked to introduce themselves and briefly describe their research area to each other. In total, they interacted for approximately 5 minutes and formed nine group combinations. Table 4.2 shows a description of the tested scenarios and different group combinations that they formed during the experiment. These orchestrated scenarios were chosen in order to simulate stationary interactions that are happening during social events. More specifically, the following four events were tested:

Table 4.2: Scenes with group formations tested during the experiment.

Name	Groups	Description
S01	3	A group of three people walk and form a stationary F-formation.
S02	3, 3	Another group of three people walk and form another stationary F-formation.
S03	2, 4	One participant leaves group A and joins group B.
S04	3, 3	Another participant leaves group B and joins group A.
S05	5, 1	Two participants leave group A and joins group B. The remaining person walks on the opposite direction.
S06	6	All participants join group B.
S07	2, 4	Two people from group B leave and form a separate formation (group A).
S08	4, 1, 1	Group A is now split and one of the participants walks close to group B twice, but do not join their formation.
S09	5, 1	The same participant now joins group B.
S10		Group splits and participants leave the room.

- A group of people is forming an interactive group (F-formation).
- One or more participants leave the group.
- One of more participants join another group.
- A group of people is dissolving an interactive group.

All data collection and analysis was made with informed consent and approved by the Queen Mary University of London research ethics committee (Appendix A).

4.3.4 Ground Truth

The performance of the algorithm was evaluated by using the video recording as ground truth. An external researcher was asked to annotate the social interactions using ELAN multimedia annotator software [97] as shown in Figure 4.4. The exact instructions, based on Kendon's F-formation [38] explained before, were:

> *An interaction begins at the moment two or more people are stationary and cooperate together to maintain a space between them to which they all have direct and exclusive access.*

All annotations were extracted from ELAN into a CSV file that includes the participant's ID, a unique ID per group formation, as well as the exact begin and end time that the participant was part of this group.

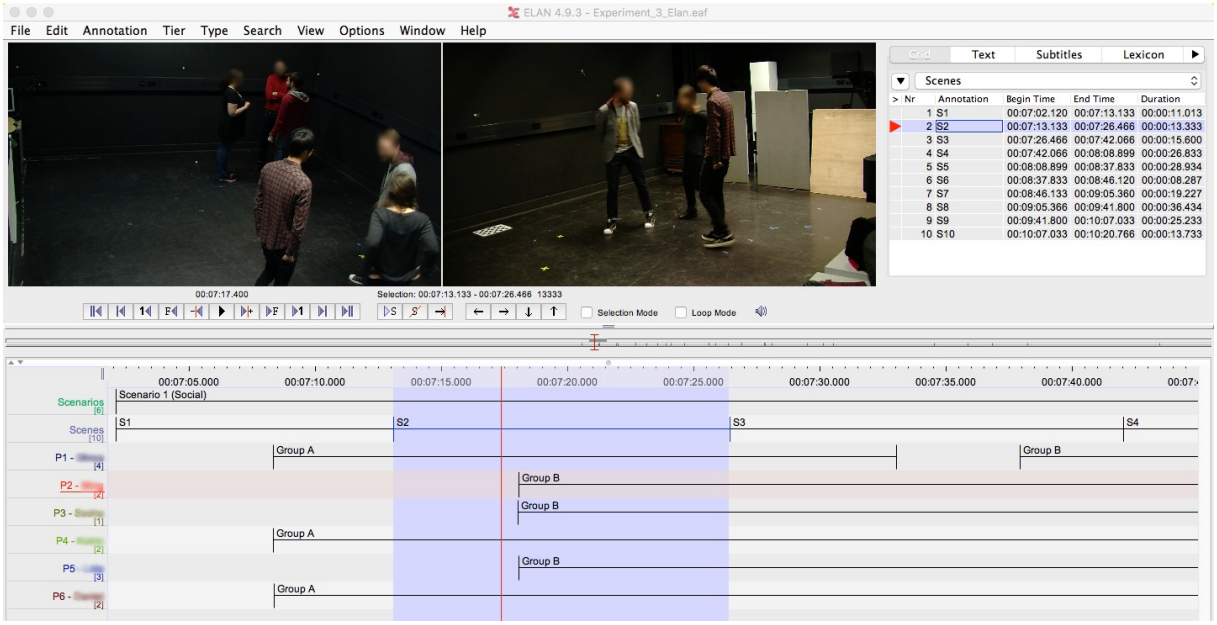


Figure 4.4: Manual annotations using ELAN multimedia annotator software.

4.3.5 Data Pre-Processing

Before starting the data analysis, all data was synchronized using the audio sample described previously, achieving an accuracy of ±50ms. All iBeacon™ Proximity data was filtered by removing values reported as *Unknown* with −1 as RSSI. This usually occurs at the beginning of the beacon ranging process due to insufficient measurements to determine the state of the other device [2] or for a few seconds after the device gets out of the beacon's broadcasting range. Based on the results of the previous experiment, Radius proximity model was applied in all RSSI measurements for estimating the proximity between devices.

Even though data from Motion Activity sensor report the user activity, classified as *Stationary*, *Walking*, *Running*, *Driving* and *Cycling*, there is a delay of 3-5 seconds before reporting the activity. For the purpose of this work, a responsive motion activity recognition is a key requirement when identifying stationary interactions. Thus, the use of Linear Acceleration data was chosen as a way to classify the user's activity between *Stationary* and *In-Motion*. This sensor only includes the acceleration of the device, excluding the gravity from the raw accelerometer data. The magnitude of the 3-axis sensor was calculated using the formula $\sqrt{x^2 + y^2 + z^2}$ in order to compute the total acceleration in one vector. This process was required since each user had the device in a different physical alignment and individual axis reading would not have provided useful information. Data was also averaged per 1 second windows and the threshold of 0.15g was empirically chosen to classify a participant between *Stationary* and *In-Motion*.

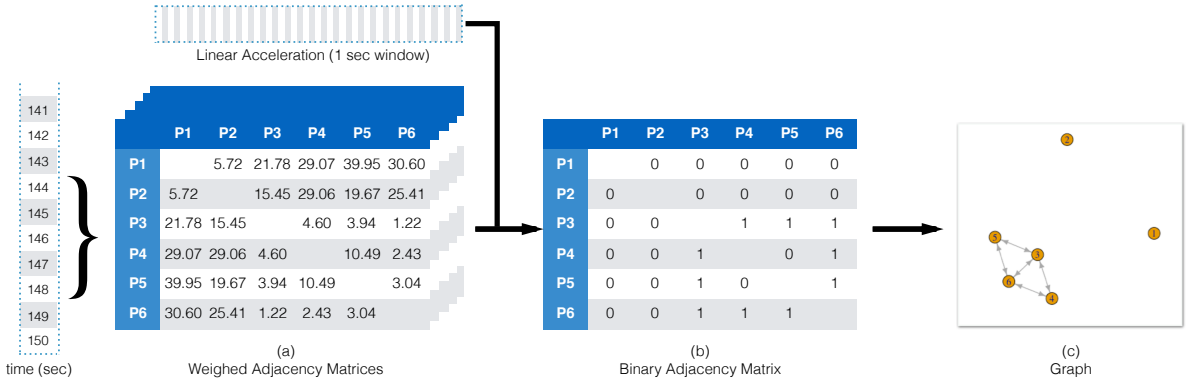


Figure 4.5: Overview of the DetecTN algorithm used for social interaction detection.

4.3.6 The DetecTN Algorithm

Inspired from the work of [26], the concept of this approach for detecting Social Interactions is based on the Graph Theory. Each moment (in seconds) is represented as an undirected weighed graph $G = (V, E, w)$, with a set of vertices V and weighed edges $E(w)$. Each vertex corresponds to a participant, and each weighted edge correspond to the distance as estimated by the iBeacon™ Proximity sensors.

An array of weighed adjacency matrices was used to represent the estimated distance of each participant for each second (Figure 4.5a). Since this work only focuses on stationary interactions, when a participant is classified as *stationary*, he/she is equally classified as non-interacting. The weighed adjacency matrices are converted into binary adjacency matrices (Figure 4.5b) using Algorithm 1. The algorithm is based on three variables: A range (*from* – *to*) of estimated distance that two participants should have, and x that represents the time (in seconds) that this distance should be maintained. When these conditions are met, an edge between the two nodes is created and represented by 1 in the matrix. An interaction is identified when a connected component exists in the graph, as shown in Figure 4.5c (Participants 3, 4, 5 and 6).

Algorithm 1: Vertice creation of binary adjacency matrix

```
if activity is not stationary then
    w ← 0
else
    if from ≤ distance ≤ to and time ≥ x then
        w ← 1
    else
        w ← 0
    end if
end if
```

4.3.7 Results

To report the performance of a classification model, a confusion matrix is very frequently used, that is a table that reports the:

- *True Positives (TP)*: Interactions that the algorithm correctly identified as interactions.
- *False Positives (FP)*: Non-interaction that the algorithm incorrectly identified as interactions.
- *True Negatives (TN)*: Non-interactions that the algorithm correctly identified as non-interactions.
- *False Negatives (FN)*: Interactions that the algorithm incorrectly identified as non-interactions.

To report the performance of this algorithm, Precision and Recall were used, two measurements that are heavily used in binary classification problems. In the context of this work:

- *Precision* (also called positive predictive value) is: from all detected interactions, how many of them did this algorithm detect correctly?
  $Precision = TP/(TP + FP)$
- *Recall* (also called sensitivity) is: from all interactions taking place, how many of them did the model detect?
  $Recall = TP/(TP + FN)$

Note that it is trivial to achieve a 100% recall, just by always making a positive prediction (*i.e.* interacting). Thus, a performance measure that represents both the precision and recall was required. The F_1 score was chosen as it is a harmonic balance between precision and recall into one measure:

$$F_1 = 2 \times \frac{precision * recall}{precision + recall} \tag{4.5}$$

Other similar measurements could be used if needed. For example, F_2 score could be used that weights recall higher than precision, in case it is required to have a model that classifies an interaction easier (*i.e.* probability is relatively low) but ending up reporting more false positives. An alternative is $F_{0.5}$ which puts more emphasis on precision rather than recall.

The DetecTN algorithm was tested with different values. More specifically, variable *to* was manipulated in the range between 2 and 10 meters, and variable x in the range

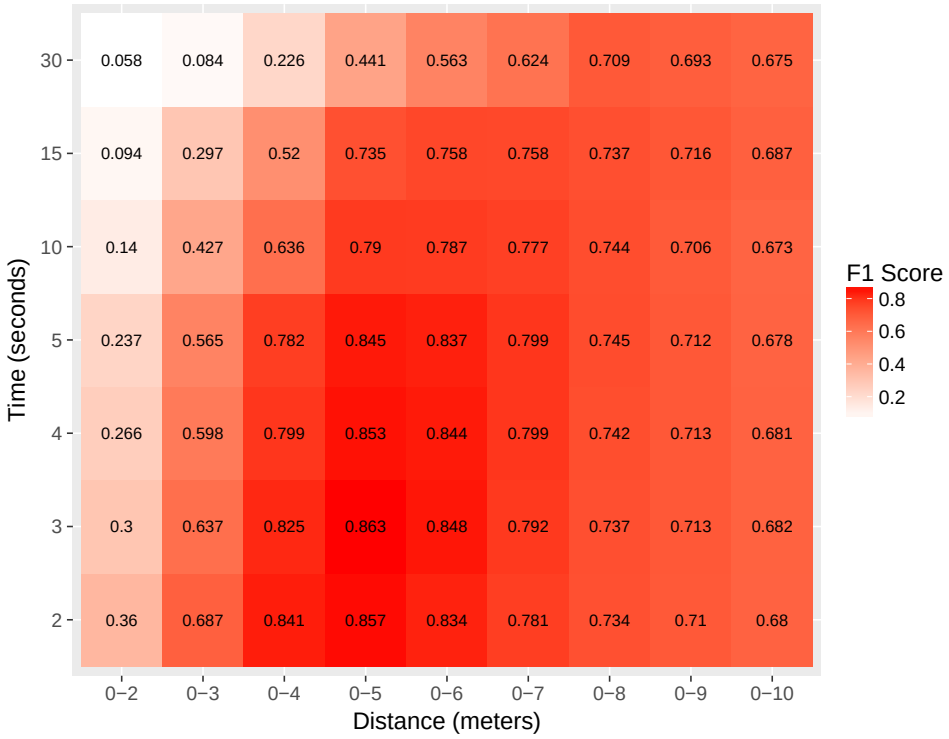


Figure 4.6: Performance of the social interaction detection algorithm at 63 different spatial-temporal scales. Warmer colours in the plot represent higher F_1 score performance.

Table 4.3: Confusion matrix reporting the performance of the DetecTN algorithm with parameters: $from = 0$, $to = 5$ and $x = 3$.

	Predicted Class		
	Positive	Negative	Total
Actual Positive	**1052 (TP)**	87 (FN)	1139
Actual Negative	248 (FP)	**1598 (TN)**	1846
Total	1300	1685	2985

between 2 to 60 seconds. Increasing variable *from* from 0 meters always resulted in a negative effect in the performance of the algorithm, thus was always set to 0. The aim of performing a parameter manipulation was to discover the best configuration that maximises the performance of the algorithm (*i.e.* maximise F_1 score).

Results from the parameter manipulation reported that the algorithm performs best with an estimated distance between 0 and 5 meters for more than 3 seconds (Figure 4.6). The maximum performance that was achieved is 80.9% precision and 92.4% recall. Table 4.3 summarizes the results of this experiment using confusion matrices.

4.4 Summary

In this chapter, a system was introduced that automatically detects social interactions inside crowds depending entirely on the embedded sensors of smartphone devices. This work is capable of detecting not only one-to-one, but also interactions of variety of group sizes. By utilizing the Bluetooth Smart and motion sensors of each device, it is able to estimate the distance between people as well as to classify the user's activity between stationary and in-motion. In addition, by using a novel algorithm it is able to predict interactions inside a small group of people, reaching a performance of 80.9% precision and 92.4% recall.

As mentioned earlier, the aim of the work in this section was to evaluate whether a system that automatically detects stationary interactions of various sizes could be feasible with current technology available in a modern smartphone. Even though the results reported a high accuracy in terms of precision and recall when detecting the interactions, these need to be further evaluated in a realistic environment that social interactions take place naturally. In contrast, the collected dataset was small (3:18 minutes long) and included only a few artificially created interactions that the researcher orchestrated. It is expected that in a real environment where interactions can take place in shorter distances (*i.e.* people standing very close to each-other due to limited space), the current approach will report increased false positives (FP), resulting into a decreased precision. This assumption was verified in a live demonstration of the algorithm at ACM MobiSys 2017 (Appendix B), with interactive groups being very close in distance (less than one meter) and in orientations that allowed direct contact between the two devices, identified as interacting together.

One of the main reasons behind this outcome is the inaccuracy of the proximity estimation. As mentioned in Section 4.2, the only option available in today's smartphone devices for detecting proximity is the use of RSSI signal from a wireless sensor (*i.e.* Bluetooth or WiFi). If other technologies ever become available, the accuracy of this method will greatly benefit. One solution that could be worth evaluating is the manipulation of the signal strength of the Bluetooth Smart sensor, known as *Broadcast Power*. By choosing a lower value in this parameter, the sensor would only be capable of detecting interactions in shorter distances. Unfortunately, currently the iOS platform does not allow the manipulation of such parameter.

One limitation of the evaluation approach is that the reported performance (based on the parameter tuning reported in Figure 4.6) is only valid to the specific environment with the current participants. Due to the limited duration of the experiment, it has not been tested in an unseen evaluation set, resulting into a parameter overfitting. A proper evaluation with an unseen (*i.e.* never trained with) dataset should be included in future evaluation of such system.

Another important restriction of this approach comes from the iOS system that does not allow the device to broadcast as an iBeacon™ in the background. It was possible to overcome this limitation by asking the participants to place the phone while the screen is on. This can be feasible in small experiments by instructing the people to do so, but it is not possible in larger scale events. Similar limitations exist in the Android OS as previously mentioned in Section 2.2.5.

In the next chapter, a successor of this work will be presented that overcomes these limitations by using an external beacon hardware that people can place in their pockets. This device can broadcast a beacon signal, simulating a device beacon broadcasting, while each participant's smartphone can be used as a beacon scanner. In addition, it will allow the manipulation of the *Broadcast Power* parameter, reducing the broadcasting range in shorter distances. Finally, a supervised machine learning approach will be evaluated that uses features extracted from a wider range of mobile sensors, aiming to achieve good accuracy in a more realistic social event where participants interact physically with each-other.

Chapter 5

Detecting Social Interactions

5.1 Overview

Chapter 4 presented the DetecTN algorithm, an approach for detecting stationary interactions using mobile sensing. More specifically, it used the Bluetooth Smart sensor to estimate the proximity between devices, and the accelerometer sensor to detect the motion state of each user. By fusing the data from the two sensors, it was capable of detecting interactions with a performance of 80.9% precision and 92.4% recall on a small dataset with six participants. This approach has some limitations which are summarized below:

- Proximity estimation based on the Bluetooth Smart RSSI of mobile devices is noisy, making it hard to differentiate between specific proximities. For example, measuring the RSSI in 0.75m and 1.75m distances reported very similar signal patterns, making it hard to cluster between zones that a social interaction could be feasible, and consequently, making hard for DetecTN to detect social interactions accurately. One solution that was suggested in Chapter 4 is the manipulation of the sensor's *Broadcast Power* configuration. This could reduce the sensor's broadcasting range and allow more fine-grained classification of distances. However, the iOS platform does not allow configuration of this setting, restricting it by default to the maximum power.

- iOS platform does not allow the device to be used as a beacon broadcaster (*i.e.* broadcast an iBeacon™ signal) while the screen of the device is off. This limitation makes this approach problematic for collecting data as people naturally keep their device locked while interacting. This behaviour was also observed in the preliminary study (Section 4).

Beside the technical limitations, this preliminary study was short (5 minutes long) and all interactions that took place were artificially made based on the researcher's guidance

during the experiment. Moreover, the complete dataset was used as both a training and an evaluation set, resulting into parameter overfitting. Thus, this may not reflect the true nature of social interactions.

The aim of the work presented in this chapter is to develop and evaluate an enhanced version of the DetecTN algorithm presented in Chapter 4, that is based on supervised machine learning. In contrast to other related works [52, 65], the evaluation of such approach uses a dataset that took place in a natural, non-artificial social setting. To overcome the limitations mentioned earlier, wearable beacons were used in order to broadcast a beacon signal while CrowdSense App was running as a background process in each user's smartphone. This also allowed to customise the broadcasting power of each beacon, achieving better accuracy in estimating the social space of each participant. Moreover, additional mobile sensors were used (*i.e.* accelerometer and gyroscope) in order to capture previously unexplored social signals that can possible contribute into the social interaction detection. Finally, the use of modern BLE sensor is used to estimate the proximity between participants that provides higher sampling rate (1 Hz) and improved performance that is also device independent.

The rest of this chapter has the following structure: Firstly, two validation experiments are conducted (Section 5.2) that evaluate the wearable beacons as a way to estimate whether two participants are close enough for a social interaction to be feasible. The aim is to investigate if the beacon's RSSI on a custom *Broadcasting Power* configuration can be a good predictor for a supervised machine learning classifier. Following the validation experiments, a case study is presented in Section 5.3 with 24 participants interacting in a social event for 45 minutes, while collecting a rich data set of mobile sensor data. Section 5.4 presents all steps that were followed to analyse the collected data and build a binary machine learning classifier. The results from the evaluation of the case study, presented in Section 5.5, report a performance increase of 26.7% over a proximity based simplistic approach. Section 5.6 presents a discussion of the results as well as the implications of the study.

5.2 Validation Experiments

To overcome the limitations summarised in Section 5.1, external coin-shaped beacons were used (RadBeacon Dot from Radius Networks[1]), referred to as *coin beacons* in the rest of this thesis and used to simulate a smartphone's iBeacon™ broadcasting functionality. These beacons are low cost (starting from $10 each), small sized ($35mm \times 35mm \times 15mm$) and can be easily placed in each participant's pocket. Furthermore, the broadcasting power can be configured into various settings (*i.e.* -18, -15, -12, -9, -6, -3, 0 and +3 dBm),

[1] https://www.radiusnetworks.com

allowing to fine-tune the broadcasting range of each device.

Beacon manufacturers generally configure the broadcasting setting into the highest possible when distributing indoor localisation solutions in order to achieve increased range, but also better accuracy in proximity estimation. However, for the purpose of this work, the requirement is to set the beacons in such a manner as to achieve the highest possible accuracy in the binary detection of a pair of participants within the zone that a social interaction is feasible. Therefore, different broadcasting power values were tested, explained in detail in the rest of this section.

All data collection and analysis was made with informed consent and approved by the Queen Mary University of London research ethics committee (Appendix A).

5.2.1 Evaluating the Effect of Proximity in the RSSI

In order to evaluate what is the most optimal broadcasting power setting for detecting if a pair is within a social enabled zone, a short experiment was conducted. Two participants were recruited: P1, male with height 1.79m and weight 73kg, and P2, male with height 1.83m and weight 87kg. P1 had the role of the broadcaster and was equipped with eight coin-shaped beacons of the same type, each configured in one of the available broadcasting power setting mentioned earlier. P2 had the role of the receiver and had an iPhone SE device placed in one of his pockets. CrowdSense App was used once again to collect iBeacon™ Proximity data from all eight beacons, for 30 seconds in 15 distances from 0.25 to 4.00m, every 0.25m.

Previous similar evaluations either used tripods [33] to install the required equipment (*i.e.* the beacons and the phone), or water bottles [65] simulating the body's water effecting the beacon's RSSI. However, these works did not capture the effect of human posture or blockage by body parts. Moreover, the signal is not only affected by the body's water, but also from the electric properties of human tissues (muscle, fat and skin) [78]. In this experiment, actual participants were used as a more realistic environmental setting.

Figure 5.1 indicates how the RSSI from each beacon changes while the distance increases. Only two plots are presented in this section (*i.e.* with broadcasting power +3 and -18 dBm), with the remaining six being available in Appendix C. It is clear from the figure that each beacon, due to its configuration, has a different RSSI range and a unique pattern. For example, in the case of the beacon configured in the highest +3 dBm power, it is hard to differentiate between distances 0.75 and 1.25, or 1.00 and 1.50. Moreover, most of the signals highly fluctuate, especially in relative long distances (greater than 1.75m). It was decided to choose the minimum *Broadcast Power* possible (*i.e.* -18 dBm) as it is easy to separate the RSSI of less than, and greater than 1m distance. Moreover, the signal looks relative smooth compared to the others, possibly resulting into a less classification error within the zone of consideration. This decision was also suggested

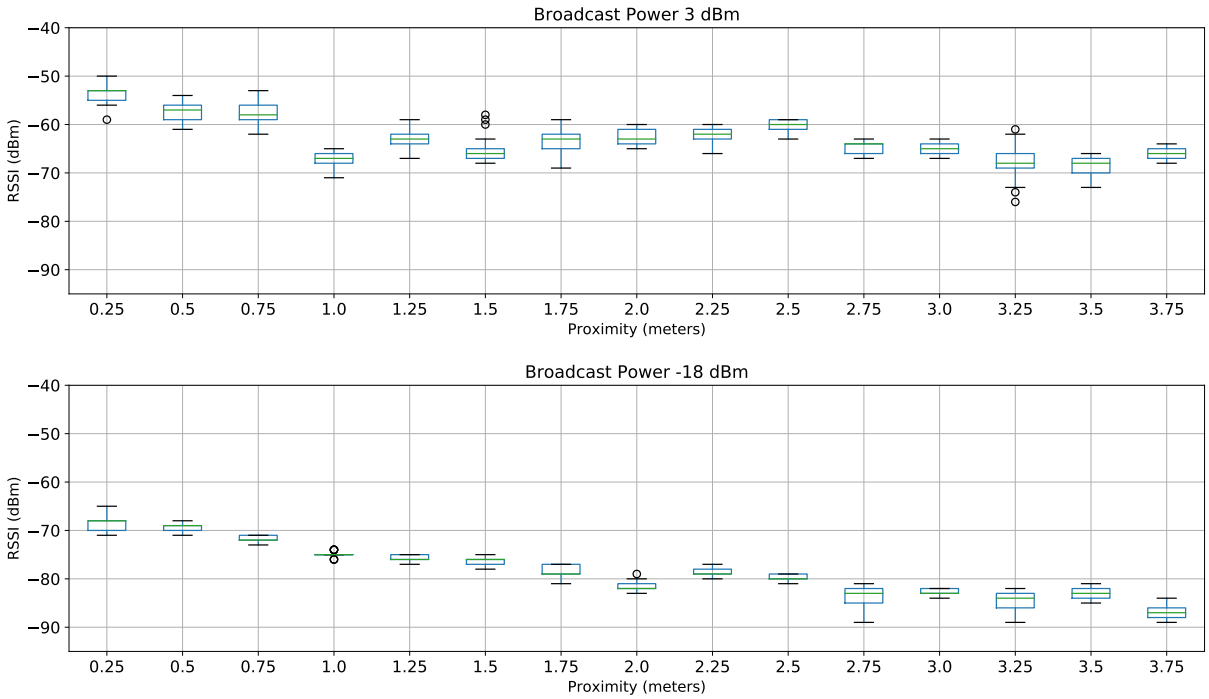


Figure 5.1: Measuring the effect of proximity on iBeacon™ signal. Each distance (x axis) was maintained for 30 seconds while P1 (broadcaster) had eight coin-shaped beacons attached to the front of his body. P2 (receiver) was collecting the beacon's RSSI signal using an iPhone SE device placed on his left pocket.

in [52] where they used the device's WiFi sensor in lowest power for the same purpose of detecting social interactions.

5.2.2 Evaluating the Effect of Body Orientation in the RSSI

As mentioned before, BLE uses the relatively low 2.4 GHz radio frequencies, resulting into RSSI measurements that are highly affected by the human body. Thus, it is obvious that the relative body orientation between two people will also affect the signal significantly. This can be seen as a desirable effect as a machine learning classifier could possibly memorise the different patterns caused by the orientation difference and only report a positive prediction (*i.e.* an interaction is happening) when the orientation and distance between people is optimal.

A second short experiment was conducted to report how the RSSI is affected from the relative body orientation and whether there are distinctive patterns that a machine learning classifier could benefit from. The same two participants were used (P1 and P2), and asked to stand facing each-other in 1m distance and engage into a conversation (Figure 5.2). P1 was the broadcaster having a coin beacon that was configured to -18 measured power placed in each of his pockets. P2 was the receiver having one iPhone SE phones in each of his pockets. CrowdSense iOS app was collecting data for 30 second on

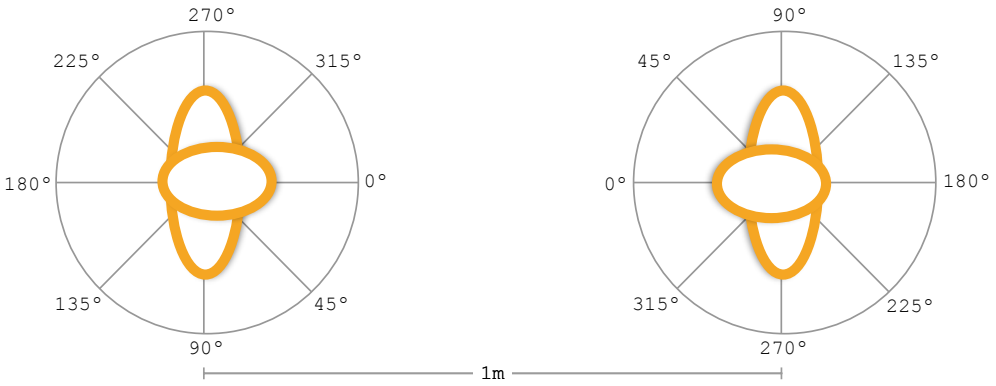


Figure 5.2: Configuration while measuring the effect of body orientation on iBeacon™ signal. P1 (left) is the broadcaster, with one beacon (AltBeacon Dot) placed in each of his two pockets. P2 (right) is the receiver, with one smartphone (iPhone SE) placed in each of his two pockets.

all 64 combinations of different orientations, with a resolution of 45°. For each 30 second window, data was collected for the following four cases:

(a) P1: beacon on the left pocket / P2: phone on the left pocket.

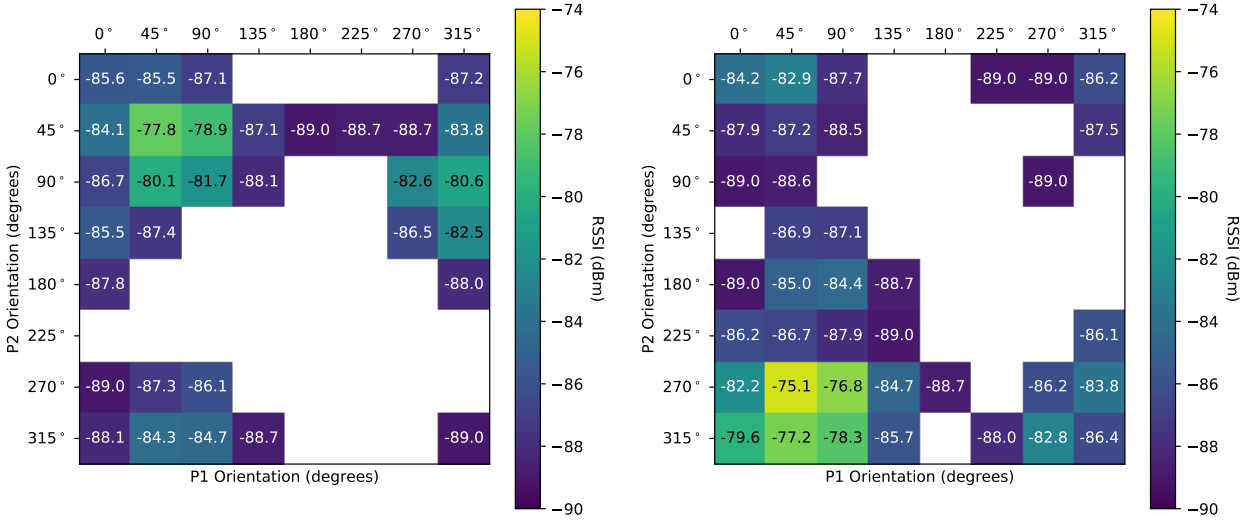
(b) P1: beacon on the left pocket / P2: phone on the right pocket.

(c) P1: beacon on the right pocket / P2: phone on the left pocket.

(d) P1: beacon on the right pocket / P2: phone on the right pocket.

Figure 5.3 visualises the mean of the RSSI for 20 seconds (excluding the first and last 5 seconds required for the orientation change) in a form of four heat-maps (one for each of the four cases mentioned earlier). As expected, the RSSI varies based on the relative orientation of the two participants. Interestingly, there are orientations that the device could not receive any signal from the beacon due to the low broadcasting power setting that was used, as the body was hiding the signal from the broadcaster. These cases, highlighted as blanks in the four figures, are desirable in our case as it is hard to maintain a social interaction in such orientations.

Figure 5.4 shows five examples of different orientations with P1 being the broadcaster, with one coin beacon placed in his left pocket and P2 being the receiver, with the smartphone device placed in his left pocket. Using the heat-map from Figure 5.3a as a reference, it is obvious that the orientations shown in the first three examples (a, b and c) report an RSSI of -85.6, -84.3 and -77.8 respectively. Note that in these three orientations, a social interaction is feasible due to small relative angle the two participants have. In contrast, in the last two examples (d and e), the RSSI is unknown due to the participant's body hiding the signal.



(a) P1: left pocket / P2: left pocket

(b) P1: left pocket / P2: right pocket

(c) P1: right pocket / P2: left pocket

(d) P1: right pocket / P2: right pocket

Figure 5.3: Measuring the effect of body orientation in iBeacon™ signal. Heat map visualisation of the mean RSSI of 20 second window over different relative orientations of the two participants. Blank entries indicates no data due to the low Broadcasting Power configuration used.

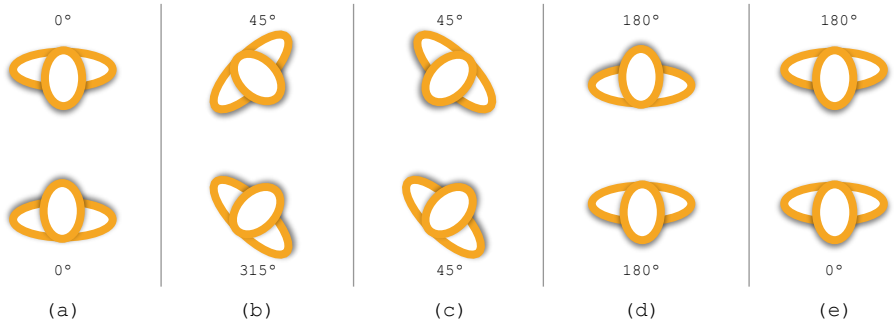


Figure 5.4: Examples of different body orientations with P1 (up) being the broadcaster and P2 (down) being the receiver.

One important observation is that the results are not symmetric. For instance, someone would expect that in a scenario where P1 is 90° and P2 135°, the measured RSSI would be similar to the symmetric scenario where P1 is 135° and P2 90° (Figure 5.3a). The asymmetry in the results can be attributed to the biometric differences (*e.g.* weight and height) between the two participants, as well as to the configuration, using P1 as the beacon broadcaster and P2 as the beacon receiver.

Another important discovery is that the RSSI is very different based on the configuration of in which pocket each device was placed (left or right). This suggests that knowing this configuration (*i.e.* in which pocket the user placed his/her phone) would result in better accuracy. Note that in Figure 5.3, only the mean RSSI is visualised. Since the signal also fluctuates (similar to Figure 5.2), a feature representing the standard deviation (SD) could also be a useful predictor.

5.3 Data Set

In order to identify and evaluate the sensors needed for detecting stationary interactions in a natural setting, data was collected from participants during a social networking event. This section includes a description of the recruited participants (Section 5.3.1), the procedure that was followed while conducting the case study (Section 5.3.2), as well as the sensor data that was collected (Section 5.3.3).

5.3.1 Participants

37 potential participants were recruited via email and flyers; 24 of those took part in the actual study of which 9 were male and 15 females, with average weight 63.75kg ($\pm$18.02), and average height 167.21cm ($\pm$9.11). Participants were selected based on their mobile phone model (iPhone 4 or higher) and the version of their operating system (iOS 7 or higher), based on the availability of the iBeacon™ sensor. Two devices experienced errors

during the study (*i.e.* Bluetooth Smart sensor reported an internal error and did not collect data) and were excluded from the data analysis, resulting into 22 valid participants.

5.3.2 Procedure

Several days before the study, participants were instructed to install CrowdSense App, based on SensingKit for iOS v0.5 [32], presented in Chapter 3. The difference between this version and the one available in the App Store is that it automated the sensor test, selection and configuration process, allowing the participants to easily register their details through an electronic form, and finally submit the data into a cloud-based server.

On the day of the study, participants were invited to the Performance Lab of Queen Mary University of London, a floor plan of which is shown at Figure 5.5. The Performance lab is 10.60×8.16 meters, with 3.90 meters height; it is suitable for such type of experiments not only due to its isolation from outside noise and environmental factors, but also due to it being a natural space often used for social events and performances. It provides a DMX lighting rig installed in 3.27 meters height which was used to fix two HD cameras (C_i) recording video (but not audio) in $25fps$, covering an area of 6.57×5.36 meters (highlighted in grey in Figure 5.5). These videos were annotated to provide the ground truth for social interactions (see Section 5.4.3). This area was restricted using plastic dividers of 1.94 meters height to make sure that all interactions would be recorded by the cameras. Additionally, five Estimote Location Beacons[2] (B_i) were installed into the lighting rig, configured into the device default $300ms$ *Advertising Interval* and $-12dBm$ *Broadcasting Power*.

Before the study began, participants were asked to read the information sheet and sign the required consent form. Participants were equipped with a Radius Networks RadBeacon Dot[3] each (coin shaped Bluetooth 4 based low energy beacons), to place in one of their pockets. All coin beacons have been pre-configured to $10ms$ advertising interval (highest) and $-18dBm$ broadcasting power (lowest), based on the results reported in Section 5.2. Half of the participants were hereby instructed to place the beacons in the left pocket and the other half in the right pocket, in a counterbalanced manner to account for random factors when people place their phone in their pocket. The phone was always placed in the pocket of the other side to avoid signal interference between the two devices.

During the setup process of the study, participants were guided through CrowdSense App configuration. This process included a facial photograph used to enable the later ground truth video annotations and completion of the demographic collection form for the gender, weight and height of the participant. As a last setup step, participants were

[2]https://estimote.com
[3]https://radiusnetworks.com

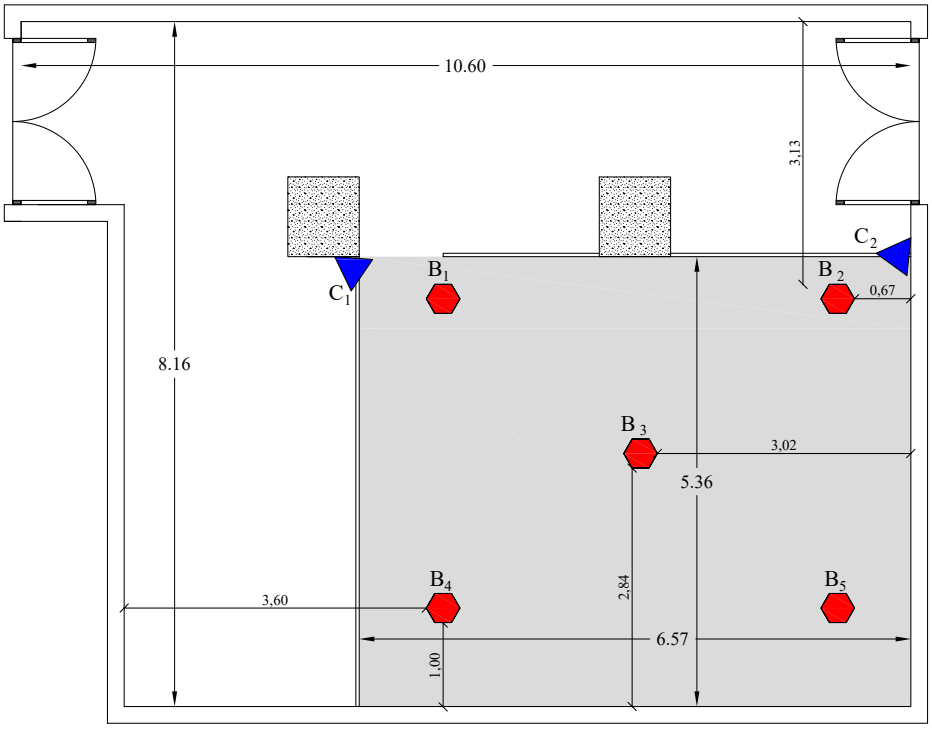


Figure 5.5: Floor plan of the Performance Lab. The two cameras (C_i) are highlighted in blue, while the five beacons (B_i) are highlighted in red and installed in a lighting rig, in 3.27m height. The area that the study took place is highlighted in grey.

asked to synchronously perform a wave-movement in front of the cameras. The recorded sensor data of each participant was later synced with the $25fps$ video feed, achieving a sync accuracy of $\pm$40ms.

Participants were then instructed to socially network for a total of 45 minutes. Snacks and beverages were served before and after the end of the experiment. The discussion topic was intentionally left open, trying to simulate a realistic networking scenario with minimum influences from our side. After the networking session, participants returned the beacons, submitted the collected data to our cloud-based server and were reimbursed with £20 for their time.

In total, 99 one-to-one interactions were observed with a mean duration of 252.7s ($\pm$164s) and 25 group interactions (*i.e.* interactions that include more than two participants) with a mean duration of 110s ($\pm$135s).

All data collection and analysis was made with informed consent and approved by the Queen Mary University of London research ethics committee (Appendix A).

5.3.3 Sensor Data Set

The dataset that was collected during the case study includes for each participant, the following sensor data:

- **iBeacon™ Proximity:** This sensor provides the RSSI from the mobile device with all beacons available in range. That includes all 24 coin beacons that the participants carried in their pocket but also the high performance beacons installed in the ceiling of the room.
- **Linear Acceleration:** It measures the device acceleration changes in three-dimensional space. This sensor excludes the 1g acceleration produced by the gravity.
- **Gravity:** It represents the orientation of the device relative to the ground, by measuring the 1g acceleration produced by the gravity.
- **Rotation Rate:** It measures the device's rate of rotation around each of the three spatial axes.

The sampling rate was set to the maximum supported (100Hz) for all motion and orientation sensors. iBeacon™ Proximity sensor has a fixed (non-customizable) sample rate of 1Hz.

The dataset of this study will be available in CrawdDad repository[4] after the completion of this thesis.

5.4 Data Analysis

This section includes the details of the data analysis procedure that was followed in order to develop the supervised machine learning model. It starts with a presentation of the data pre-processing procedure that was followed (Section 5.4.1), a description of the Target Variable (Section 5.4.2) as well as the Ground Truth that was used (Section 5.4.3). The section continues with a presentation of a simple probabilistic approach for detecting interactions, named Normalised Inverted Proximity (Section 5.4.4). It continues with a detailed presentation of the selected features (Section 5.4.5) and concludes with the evaluation procedure that was followed (Section 5.4.6) as well as an explanation of the model choice and model tuning process (Section 5.4.7).

[4]https://crawdad.org

5.4.1 Data Pre-processing

During the pre-processing process, the data and video feed were synchronised based on the distinguished movement the participants performed (*i.e.* synchronously perform a wave-movement in front of the cameras) as mentioned in Section 5.3.2.

For all iBeacon™ Proximity data, all data reporting *Unknown* values (where RSSI is -1) were excluded. This usually occurs at the beginning of iBeacon™ ranging process due to insufficient measurements to determine the state of the other device [2], or for a few seconds after the device gets out of the beacon's broadcasting range. All measurements from each user's beacon (*i.e.* from his phone to his owed beacon) were also excluded. The Path Loss Model (PLM) was also applied in order to estimate the proximity (d) between each device and all beacons in range using the RSSI ($P(d)$), as shown in the following formula:

$$d = 10^{\frac{P(d_0)-P(d)-X}{10\times n}}, \tag{5.1}$$

where $P(d_0)$ is the Measured Power (in dBm) at 1-meter distance, n the path loss exponent, d the distance in which the the RSSI is estimated and X a component that describes the path loss by possible obstacles between the transmitter and the receiver. The value $n = 1.5$ was set as a default constant for indoor environments [52]. The value $X = 0$ was also chosen as it was required to measure a direct contact where no obstacles (*e.g.* other participants) between the two devices exist. In the situation that another participant exists in between, PLM would report a longer distance due to the decreased RSSI, and consequently, the accuracy of the distance estimation will decrease. This is a desired effect as, in the case of the coin beacons, it is only wanted to cluster whether the two users are within a range that a social interaction can be achieved. According to Hall [22], personal social interactions are achievable between 0.5 and 1.5 meters distance. Moreover, since all five long range beacons were installed in the ceiling of the room, there was expected a clear path between the phone and most of the ceiling beacons.

Since the mobile devices are not real-time systems, setting a sample rate is only a suggestion to the operating system and actual rate will vary second to second. Thus, the signal for the *Device Motion* sensor was resampled and interpolated to 100Hz. Finally, the magnitude was computed from the three axes of all motion data (*i.e.* User Acceleration, Gravity and Rotation Rate) available in the dataset, a process required since each user had their device in a different physical alignment and individual axis reading would not have provided useful information.

The iBeacon™ sensor was the only sensor that reported missing values. Since most machine learning algorithms (*e.g.* Logistic Regression, Random Forest) do not accept features with unknown values, a data imputation process was required. Thus, missing values

for external beacon data were imputed using linear interpolation [55] on the estimated distance from the device to the ceiling beacons. Since users are changing their state less frequent, it should be possible to estimate any possible missing value reliably using this approach. For the interpersonal distances inferred from the coin beacon, missing values were replaced with the maximum available distance, since in these cases, due to the low *Broadcast Power* that was used, the reason for missing data was that the device was out of range from the broadcaster.

5.4.2 Target Variable

The dataset has a total of 645,895 labels for each combination of the 22 valid participants interacting for a total of 45 minutes in the case study. The target variable is binary, with the following two classes: 1 when a pair of participants is interacting, and 0 when it is not. That resulted into 38,332 labels in class 1 (6.31%), and 607,563 labels in class 0 (93.69%).

The dataset is naturally imbalanced since it includes one label for all combinations of the participants interacting with each-other per second. The level of this imbalance obviously depends on the number of people interacting, but also on the type of interaction (*e.g.* one-to-one, groups of three etc.). For instance, a small event of 6 people will include $C(6, 2) = 15$ pairs, and thus 15 labels per second. If only three participants (A, B and C) interact together in a group of three, the dataset will include three labels of 1, one for each combination of them (AB, AC and BC), while the remaining 12 will belong to the class 0.

5.4.3 Ground Truth

The case study, recorded from two different angles for convenience, was annotated by two independent annotators using ELAN multimedia annotator software [97]. The annotations were cross-validated afterwards and finally verified by a third person. The instructions that the annotators followed were based on Kendon's F-formation system [38]:

> *An interaction begins at the moment two or more stationary people cooperate together to maintain a space between them to which they all have direct and exclusive access.*

The aim of the study was to detect stationary interaction only. Therefore, the annotators logged the begin and end of each stationary interaction for each participant separately, using a unique ID per interaction.

5.4.4 Normalised Inverted Proximity

The *Normalized Inverted Proximity* (NIP) is suggested by this work as a simplistic approach for detecting social interactions using proximity-based information. More specifically, the distance of two participants is used (computed using the Path Loss Model discussed in Section 5.4.1) with all unknown values (*i.e.* when the pair is out of beacon range) being replaced with the max of all distance estimations. The data are then normalized to the range of 0 and 1 and the outcome is inverted. This baseline was chosen as a simplistic approach when using proximity-based sensors, and can be computed using the formula below:

$$\hat{y} = 1 - \frac{x - x_{min}}{x_{max} - x_{min}}, \tag{5.2}$$

where $\hat{y}$ is an estimate as to whether the pair is interacting, x is the estimated proximity between the pairs, x_{min} and x_{max} are the minimum and maximum estimated proximity between all pairs in the dataset. Because $\hat{y}$ is in the range [0, 1] it can be compared to probability estimates.

The reason that the DetecTN algorithm is not used in this case is that it directly produces binary predictions and not probabilities. Thus, it would not be possible to compare the two approaches using the probabilistic based plots used in the rest of this chapter (*i.e.* Precision-Recall (PR) and Receiver Operating Characteristic (ROC) plot). Probabilistic predictions have the advantage that the designer can choose a cut-off threshold that maximises precision or recall, depending on the use case. For example, it might be desired to choose a high precision over recall so that the model only makes a positive prediction when the probability of an interaction is very high, resulting in an accurate result with the disadvantage of losing some interactions that took place. The Normalised Inverted Proximity is also based on the estimated proximity between two people and is expected to report similar results.

5.4.5 Feature Selection

A series of common features were computed for all $C(22, 2) = 231$ combinations of the participant pairs. Features reflecting the current moment were initially computed, in a static window of 1 second, following with features reflecting past information. A set of features that are commonly included in mobile sensing problems were used, such as features extracted from motion and orientation sensors. The intuition behind the inclusion of those features is to capture the social signals that exist in conversations such as *mirroring*, when a person is subconsciously imitating the gestures or movements of another [12].

Based on the results from the validation experiments reported in Section 5.2, additional features were explored that provide more precise information for detecting the social

Table 5.1: List of the features used in the data analysis.

Table of Features	
Interpersonal Space Features	$f_{prox\_rssi\_mean}$ $f_{prox\_rssi\_diff}$
Device Position Features	$f_{device\_position\_LL}$ $f_{device\_position\_LR}$ $f_{device\_position\_RL}$ $f_{device\_position\_RR}$
Indoor Positioning Features	$f_{external\_beacon\_1\_diff}$ $f_{external\_beacon\_2\_diff}$ $f_{external\_beacon\_3\_diff}$ $f_{external\_beacon\_4\_diff}$ $f_{external\_beacon\_5\_diff}$
Motion and Orientation Features	$f_{time\_since\_moving\_diff}$ $f_{device\_linear\_acc\_ccf\_lag}$ $f_{device\_linear\_acc\_ccf\_max}$ $f_{device\_gravity\_ccf\_lag}$ $f_{device\_gravity\_ccf\_max}$ $f_{device\_rotation\_rate\_ccf\_lag}$ $f_{device\_rotation\_rate\_ccf\_max}$
Example of Past Information Features	$f_{external\_beacon\_1\_diff\_past\_min}$ $f_{external\_beacon\_1\_diff\_past\_max}$ $f_{external\_beacon\_1\_diff\_past\_mean}$ $f_{external\_beacon\_1\_diff\_past\_std}$ $f_{time\_since\_moving\_diff\_past\_min}$ $f_{time\_since\_moving\_diff\_past\_max}$ $f_{time\_since\_moving\_diff\_past\_mean}$ $f_{time\_since\_moving\_diff\_past\_std}$

interactions based on the participant's position (*i.e.* interpersonal space, device position and indoor positioning features). Table 5.1 lists the extracted features used in the data analysis of this work. The rest of this section reports on all 74 produced features and the selection strategy that was followed.

Interpersonal Space Features

iBeacon™ Proximity sensor data of a pair includes two measurements: Let $rssi_{ij}$ be the RSSI between the two participants as measured from the device of user i and $rssi_{ji}$ be the RSSI from the same distance as measured from the device of user j. The mean of the two measurement was computed as an indication of how close the two participants are in space:

$$f_{prox\_rssi\_mean} = (rssi_{ij} + rssi_{ji})/2 \tag{5.3}$$

In addition, a feature that represents the absolute difference between the two measurements was computed:

$$f_{prox\_rssi\_diff} = |rssi_{ij} - rssi_{ji}| \tag{5.4}$$

It is expected that people closer together would be more likely to also engage in an interaction. Note that in this case, the raw RSSI was used as the same hardware was used for broadcasting a beacon signal across all participants, and thus, a *Measured Power* constant is not required. In the case of multiple devices being used, then a feature that estimates the interpersonal distance based on a calibrated *Measured Power* constant would be required, using the PLM equation mentioned in Section 4.2.

Device Position Features

As mentioned in Section 5.2, information about the device position is important as it highly influences the RSSI signal between the two devices. For that reason, four features have been developed that include the information of the device position (left vs. right per participant) using one-hot encoding:

- $f_{device\_position\_LL}$: Both P1 and P2 placed the device on the left pocket.
- $f_{device\_position\_LR}$: P1 placed the device on the left pocket, P2 on the right.
- $f_{device\_position\_RL}$: P1 placed the device on the right pocket, P2 on the left.
- $f_{device\_position\_RR}$: Both P1 and P2 placed the device on the right pocket.

Indoor Positioning Features

The absolute difference of each participant from the five ceiling beacons was computed using the following formula:

$$f_{external\_beacon\_k\_diff} = |D_{ik} - D_{jk}|, \tag{5.5}$$

where D_{ik} is the distance reported from user i's, and D_{jk} is the distance reported from user j's mobile device to the fixed installation k. It is expected that users close together would result in similar distances from the ceiling beacons and the feature will be close to zero. Note that estimated distance using PLM was used in this case instead of the raw RSSI as the high-performance beacons installed in the room ceiling were used.

Motion and Orientation Features

By using the measurements of the linear acceleration sensor, a feature that indicates the time since the participant has moved (in seconds) was added. A threshold of $0.15g$ was

empirically chosen, indicating whether a user is moving or not, and computed the absolute difference between the pair. It is expected that if two users are moving, they will stop at the same moment and engage into a conversation, and thus, the value of that feature will be close to zero. When both users had the status 'in motion', the feature was set to NaN (Unknown).

For all motion sensor data (*i.e.* Linear Acceleration, Gravity, Rotation Rate), a cross correlation function was applied on an overlapping window of 10 seconds and extracted the maximum correlation, as well as the distance (in seconds) from the max correlation, as an indication of how similar a pair is behaving on those windows. The 10 seconds constant was chosen as indicated by [52], but further investigation in the range of 2 to 60 also verified it as the most optimal constant. An alternative to the cross correlation function was also tested based on the Dynamic Time Warping (DTW) [85] method. However, due to its high computational complexity as well as its low predictive power in this context, it was excluded from the final feature list.

Past Information Features

In order to take advantage of past information available in the data set, the *min*, *max*, *mean* and *std* was computed on all time-series features (*i.e.* excluding the one-hot encoded device positioning features), in an overlapping window of 10 seconds. Note that due to the length of those features, only some representative examples are listed in Table 5.1.

5.4.6 Evaluation Procedure

For evaluating the performance of the model, a standard 10-fold cross-validation schema was used. The dataset was initially split over time, however, due to the time-series nature of our study, a significant overfitting was reported. More particularly, since participants were changing their interactive state at any given moment, the model was memorizing the features per split and inferring them back with very high performance, due to information leakage. Thus, it has been decided to split the data per participant combination (*i.e.* 23 samples out of 231, due to the 10-fold schema) rather than over time.

In the context of this work, *Precision* is: from the detected interactions, how many of them did the model detect correctly, whereas *Recall* is: from all interactions taking place, how many of them did the model detect. Depending on the use case, applications can emphasize on one of these measures over the other. If it is more important not miss a detected interaction, an emphasis can be given on recall. If it is more important to avoid bothering users with false detections, an emphasis can be given on precision.

The evaluation metrics that will be used in the rest of this report is *Precision-Recall (PR)* curve and *Receiver Operating Characteristic (ROC)* curve. Even though ROC curves

are heavily used when reporting performance in classification problems, due to the nature of our dataset being unbalanced, PR plots are also included as suggested by [84] and [17] for imbalanced datasets. In the rest of this section, both curves will be reported, together with the Average Precision (AP) for the PR curve and the Area Under Curve (AUC) for the ROC curve to represent the performance numerically.

5.4.7 Model Choice

Three different machine learning models were evaluated that represent different classes of Machine Learning: *Logistic Regression* [63], *Random Forest* [8] and *XGBoost* [13]. A further evaluation of Support Vector Machines (SVM) [18] was also explored, however due to its scaling limitation as well as its complexity when making probabilistic predictions it has been excluded from the analysis. All three models were used as part of the Python library scikit-learn [66] v0.19. In the case of XGBoost model, its scikit-learn python wrapper was used for XGBoost v0.7.2.

An initial parameter tuning was performed on a 20% subset of the dataset (*i.e.* 46 samples out of 231) using all three models. This subset was only used for the model tuning task and was never used in the training/validation procedure. The aim was to discover the model with the configuration that maximises the Average Precision (AP) performance. More specifically, a grid search algorithm over all possible combinations of the most influential parameters was followed, based on the following strategy:

- For Logistic Regression and Random Forest, the *class_weight* parameter was set to *balanced*. For the XGBoost, the equivalent parameter *scale_pos_weight* was set to $sum(negative\_cases)/sum(positive\_cases)$, as suggested by XGBoost documentation[5]. This was required due to the imbalanced nature of the dataset.

- For the Logistic Regression model, the inverse of regularization strength (*i.e.* C parameter), was manipulated with the values [1, 10, 100, 1000].

- For the decision trees -based models (*i.e.* Random Forest and XGBoost), the total number of trees (*i.e. n_estimators* parameter) was set to 50. Moreover, the maximum depth of the tree (*i.e. max_depth*) was tested with values [4, 6, 8, 10].

- For the Random Forest model, the number of features to consider when looking for the best split (*i.e. max_features* parameter) was tested with values ['sqrt', 'log2', 'None']. The value 'None' includes all features from the data set. For the XGBoost model, the equivalent parameter (*i.e. colsample_bytree*) was tested with values [0.2, 0.4, 0.6, 0.8, 1].

[5]https://github.com/dmlc/xgboost/blob/master/doc/parameter.md

For the rest available parameters, the library default values were used.

Tables D.1, D.2 and D.3 listed in Appendix D, report the results of all parameter manipulation of the three machine learning classification algorithms. XGBoost reported the best performance in the 20% subset with an AP 83.0% and ROC AUC 95.2%. The model was configured with `n_estimators=100`, `max_depth=10` and `colsample_bytree=0.2`.

A final and more extended model tuning was conducted to the model with the best performance (*i.e.* XGBoost), exploring two additional parameters including the *n_estimators* and *max_depth* mentioned above:

- The sub-sample ratio of the training instance (*i.e. subsample* parameter) was tested with values [0.5, 0.75, 1].
- The model's learning rate (*i.e. learning_rate* parameter) was tested with values [0.01, 0.05, 0.1].

This resulted into a total of 180 configuration, with results shown in Table D.4. The configuration with the best performance of AP 83.8% and ROC AUC 95.2% had the parameters `n_estimators=100`, `max_depth=4`, `colsample_bytree=0.2`, `subsample=0.5` and `learning_rate=0.05`. This configuration is used in the rest of this section for training and validation the model with the remaining 80% of the data set.

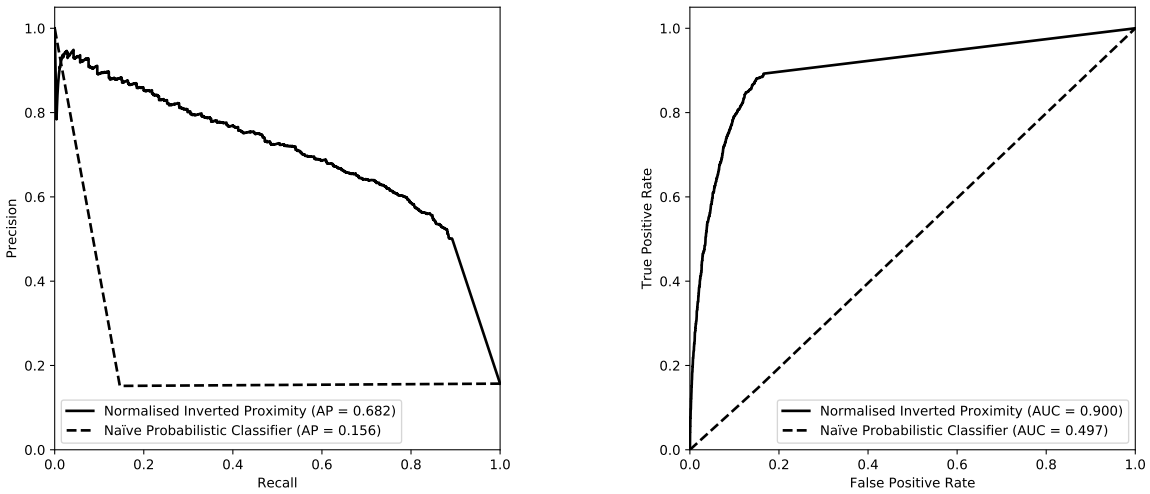
5.5 Results

This section presents the results from the analysis that was reported in Section 5.4. It starts with a report of the two baselines that were used (Section 5.5.1), and continues with the report of the general performance of the model (Section 5.5.2). It continues with an improvement of the predictive performance using a normalisation approach (Section 5.5.3) and reports the results for every participant separately (Section 5.5.4). In addition, it presents an analysis of the most predictive sensors (Section 5.5.5), the most predictive features (Section 5.5.6), and reports the most optimal cut-off threshold using a subset of the dataset as ground truth (Section 5.5.7).

5.5.1 Baseline

For comparing the performance of the model, two different baselines were chosen: (1) a Naïve Probabilistic Classifier (NPC) that uses the training set distribution of the two classes as a weight when making a prediction, and the Normalised Inverted Proximity (NIP) that was reported in Section 5.4.4.

Figure 5.6 reports the results of the two baselines. The probabilistic classifier reported an AP of 15.6% and ROC AUC of 49.7%. In contrast, NIP performed very well, reporting



(a) PR Curve

(b) ROC Curve

Figure 5.6: Performance of the two baselines: a Naïve Probabilistic Classifier (Baseline) and the Normalized Inverted Proximity (NIP).

an AP of 68.2% and a ROC AUC of 90.0%. For the rest of this section, the two baselines will always be reported for easy comparison with the model's performance.

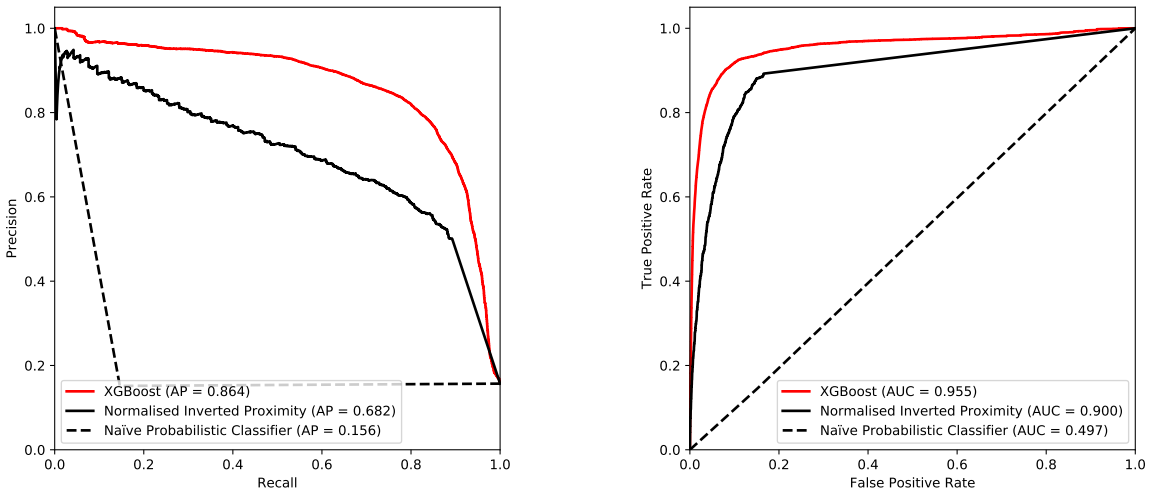
5.5.2 General Performance

Figure 5.7 shows the performance of the XGBoost classifier using a standard 10-fold validation on the remaining 80% samples of the dataset (*i.e.* excluding the 46 samples used for the model tuning). Results report a performance of 86.4% AP (*i.e.* a 26.7% increase from NIP and 453.8% increase from NPC baseline), and a similar 95.5% ROC AUC (*i.e.* a 6.1% increase from NIP and 92.2% increase from NPC baseline).

5.5.3 Normalise Predictions

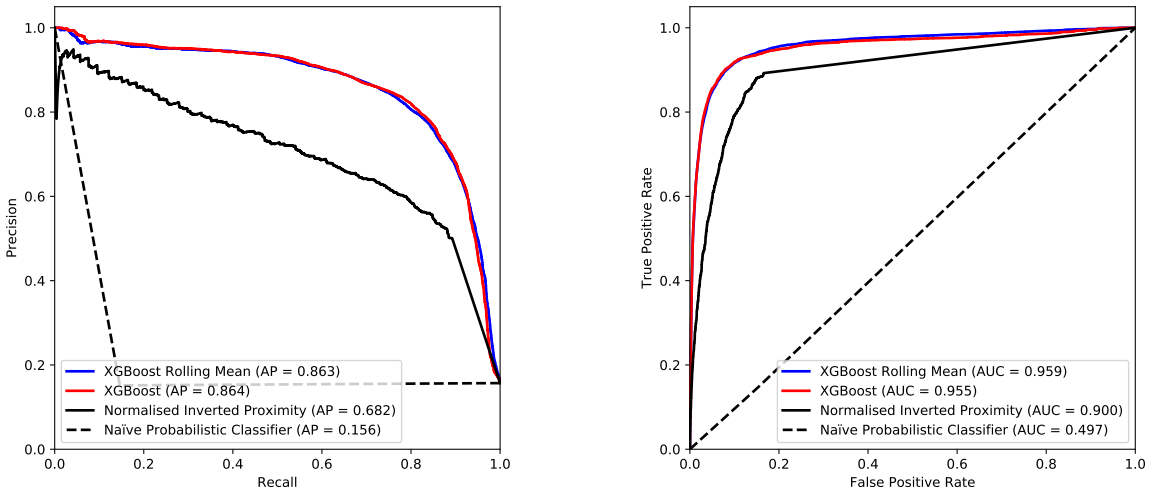
The approach of this work is based on real-time binary predictions of the status of each pair (not interacting vs. interacting). However, in reality interactions do not change in such high frequency. Statistics of the dataset support this claim, with interactions lasting for an average of 4.2 minutes. Thus, it is expected that by applying a normalisation on the model's prediction it would result into more realistic and stable predictions.

A rolling mean function was applied to smooth the prediction of the model. By experimenting with a variety of different rolling mean constants (*i.e.* range between 2 and 60), in the dataset used for model tuning, the constant of 11 seconds reported the best performance. Figure 5.8 reports the results when using the normalisation approach. Results were very similar to the ones produced directly from the XGBoost model, achieving



(a) PR Curve (b) ROC Curve

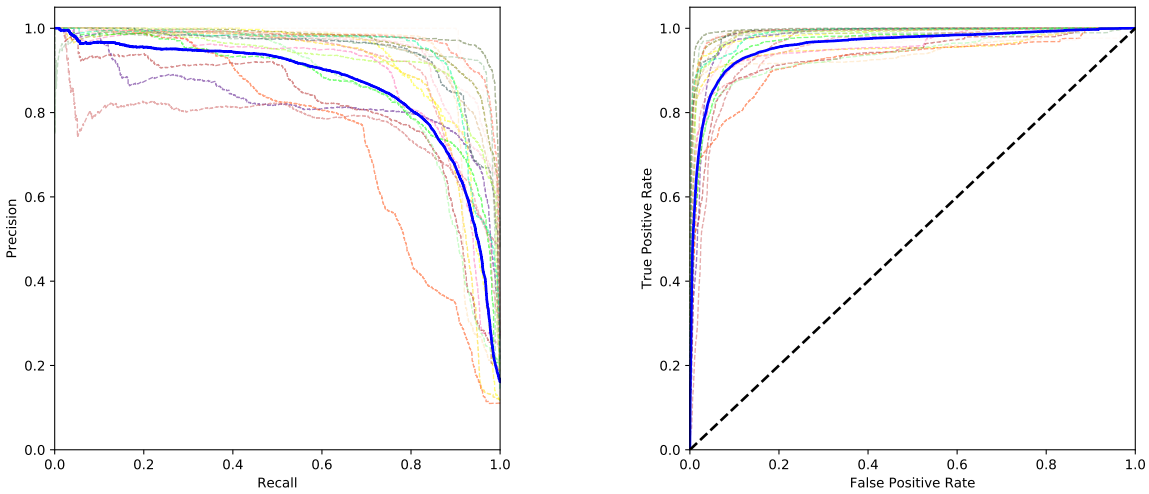
Figure 5.7: General performance of XGBoost classifier using a Precision-Recall (PR) Curve (left) and Receiver Operating Characteristic (ROC) Curve (right). The two figures also include the performance of the Naïve Probabilistic Classifier (NPC) and the Normalized Inverted Proximity (NIP) for easy comparison.



(a) PR Curve (b) ROC Curve

Figure 5.8: Performance of XGBoost classifier when the prediction probability is normalised using a 11 seconds of rolling mean window.

a total of 86.3% AP and 95.9% ROC AUC. However, due to the normalisation procedure, the results are expected to be more stable (*i.e.* less variations over time) and therefore, normalised results will be reported in the rest of this report.



(a) PR Curve

(b) ROC Curve

Figure 5.9: Performance of XGBoost classifier per participant. The coloured lines correspond to the performance of each participant, and the thick blue line corresponds to the overall average across all participants.

5.5.4 Performance Per Participant

Figure 5.9 provides a more fine-grained analysis of the results. It depicts the user-averaged PR and ROC curves for every participant, as well as the overall performance across all participants. It is clear that the model performance varies per participant, with some of them reporting almost perfect scores, while in some others perform lower than average (blue thick line).

5.5.5 Sensor Importance

As discussed in Section 3.4, accessing mobile sensor data has a significant effect on the battery life of the device, with some sensors such as the *Device Motion* being one of the most power expensive sensor of all others. This section provides a more fine-grained analysis of the results. It explores the sensor contribution of this approach, aiming to understand which sensors produce the features that are the worst predictors and how the model's performance will be affected when they are excluded.

For measuring the sensor contribution, a leave-one-sensor-out technique was used. The model was tuned (using the same approach discussed in Section 5.4.7 for XGBoost model only) and then validated with all features except the ones produced by the excluded sensor. The following sensors (or external information in the case of device position) were manipulated:

- Interpersonal Space Features (*i.e.* features related to the coin beacons).

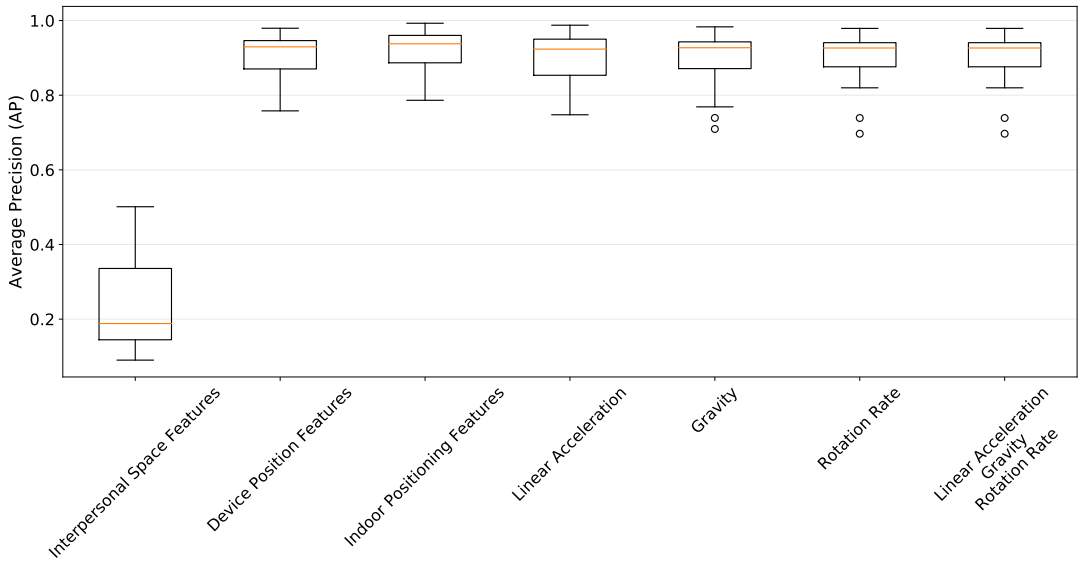


Figure 5.10: Sensor importance using leave-one-out as reported by the XGBoost classifier. The variability in the boxes corresponds to the considered participants.

- Device Position Features (*i.e.* the one-hot encoded information of the smartphone position).
- Indoor Positioning Features (*i.e.* features related to the external beacons).
- Motion and Orientation Features (*i.e.* features related to the Device Motion sensor).

In the case of *Motion and Orientation Features*, the three sensor-fused data (*i.e.* Linear Acceleration, Gravity and Rotation Rate) are explored together, but separately as well. Figure 5.10 shows the results from this analysis. It is obvious that by excluding the interpersonal space related features, the model reports random performance, similar to the NPC classifier (*i.e.* AP 18.3% and ROC AUC 55.2%). The remaining sensors have a less significant effect to the model performance, with the removal of *Device Position Features* reporting AP 86.0% and ROC AUC 95.9%, *Indoor Positioning Features* reporting AP 88.9% and ROC AUC 96.7%, and *Motion and Orientation Features* reporting AP 85.3% and ROC AUC 95.9%.

It is worth mentioning the slight improvement (approximately 3%) of the performance when the Indoor Positioning Features are excluded. These results suggest that due to the low accuracy of the proximity estimation between the device and the ceiling beacons, the use of those features can confuse the classifier.

5.5.6 Feature Importance

Table 5.2 reports the feature importance of the model. More specifically, it reports the top 15 features, together with their F score as reported by the XGBoost classifier. It is clear

Table 5.2: Feature importance as reported by the XGBoost classifier.

Feature	F Score
$f_{prox\_rssi\_mean\_past\_mean}$	85
$f_{prox\_rssi\_mean\_past\_max}$	63
$f_{external\_beacon\_3\_diff}$	52
$f_{device\_gravity\_ccf\_max\_past\_mean}$	51
$f_{external\_beacon\_5\_diff}$	42
$f_{external\_beacon\_2\_diff\_past\_min}$	40
$f_{external\_beacon\_2\_diff}$	38
$f_{device\_gravity\_ccf\_max\_past\_max}$	38
$f_{external\_beacon\_4\_diff\_past\_max}$	38
$f_{prox\_rssi\_mean}$	37
$f_{device\_rotation\_rate\_ccf\_max\_past\_max}$	37
$f_{time\_since\_moving\_diff\_past\_mean}$	36
$f_{device\_gravity\_ccf\_max}$	35
$f_{external\_beacon\_4\_diff\_past\_min}$	35
$f_{time\_since\_moving\_diff\_past\_mean}$	34

that the best predictors are the features extracted from the iBeacon™ sensor (Interpersonal Space Features), supporting the results from the previous section that a model that does not make use of that sensor will have a noticeable impact on the performance. Six features that belong to the Indoor Positioning category (use of external beacon infrastructure) are also reported as good predictors in this list. However, as discussed earlier, these features slightly confuse the classifier reporting a reduced performance in the validation set.

5.5.7 Probabilistic Threshold Choice

Until now, reporting the performance of the model was made through metrics or plots that depend on probabilities (*i.e.* Precision-Recall and ROC plots as well as numerical representations of these plots such as AP and ROC AUC). In a real-world implementation of this model, a binary prediction will be required instead of a probability. The designer of such system can choose a threshold (also called probability cut-off) of which probabilities greater or equal to this threshold would be classified as 1, and 0 in all other cases. Choosing such threshold would lead into reporting the performance of the model in terms of *precision* and *recall* as explained in Section 5.4.6. Depending on the use case, applications can emphasize on one of these measures over the other. For example, in a use case of a mobile app that users install in order to log their interactions at a social event, the designer could emphasise on recall if the requirement is not to lose people they've interacted with, even if that results into increased false positives.

For computing the most optimal threshold, the F1 score was used as a harmonic mean

Table 5.3: Confusion matrix reporting the performance of the XGBoost classifier with cut-off $p = 0.67$

	Predicted Class		
	Positive	Negative	Total
Actual Positive	**123918 (TP)**	5784 (FN)	129702
Actual Negative	3969 (FP)	**20170 (TN)**	24139
Total	127887	25954	153841

Table 5.4: Confusion matrix reporting the performance of the Normalised Inverted Precision (NIP) with cut-off $p = 0.48$.

	Predicted Class		
	Positive	Negative	Total
Actual Positive	**118589 (TP)**	11113 (FN)	129702
Actual Negative	6052 (FP)	**18087 (TN)**	24139
Total	124641	29200	153841

between precision and recall. Other similar measurements could be used if needed. For example, F_2 score could be used that weights recall higher than precision, or $F_{0.5}$ which puts more emphasis on precision rather than recall. F1 was maximised in the same set used for model tuning, following a similar approach to the one presented in Section 4.3.7. The value $p = 0.67$ for the XGBoost model and $p = 0.48$ for the NIP was found to be the most optimal that maximises the F1 score.

Figure 5.3 shows the confusion matrix of the XGBoost classifier using a cut-off $p = 0.67$. It reports a precision of 77.7% and recall of 83.6%. Figure 5.4 shows the confusion matrix of NIP using a cut-off $p = 0.48$. It reports a lower precision of 61.9% and lower recall of 74.9%.

5.6 Summary

The results of this work provide evidence that it is possible to detect interactive groups of various sizes relying in data collected from mobile devices, with a reasonable performance of 77.7% precision 83.9% recall. That means that 77.7% of the participants that the model discovered as interacting were correctly detected, and 83.9% of all interactions that actually took place during the event were detected by the model.

The dataset that have been analysed, even though extended compared to other similar studies [65, 52, 33], it only represents a subset of what is expected in similar social gatherings, such as conferences or other networking events. One technical challenge that arises is whether a real-time system that would continuously receive data from larger number of participants would be possible. Such system could be implemented as a cloud-based solution that either relies on a reliable internet connection or save the data temporarily

into the device and only submit when the event is completed. At this moment, such implementation is only possible using external wearable beacons that simulate the beacon broadcasting of each device, due to the restrictions of current mobile operating systems mentioned earlier.

One important question that arises is whether a system without a fixed infrastructure (*i.e.* that makes use of the fixed ceiling beacons) would be required for detecting the interactions within the crowd. Surprisingly, the indoor positioning features, not only did not improve the performance, but also had a negative impact of approximately 3%. The use of features that only depend on the interpersonal proximity or the motion and orientation sensors of each participants is an indication that it could work in new environments without requiring an additional model training. Note however that to analyse the data, features from all combinations of participants need to be extracted. This is possible with a reasonable number of participants but does not scale to larger crowds. The use of external beacons installed in the room, even though did not improve the performance of the final model, could be a possible solution to this limitation. This can be achieved by using the proximity of each external beacon as a pre-filter, cluster the crowd into smaller groups and only apply the model on each cluster. Such implementation could also benefit from parallel processing, assigning clusters to analyse each region in parallel.

Another challenge in the real-world application of the model is user adaptation. Requesting access to people's mobile phone can be hard as it highly depends on the people willingness to share their data. Concerns about the privacy of the data collection, or the battery life of their devices might discourage people from participating. However, according to Wirz *et al.* [95], people are open to share their data as long as they receive some benefits from it or if they realise that sharing such information is for their own good and safety. In the rest of this section, the battery consumption, privacy implications and limitations of this study are discussed separately.

5.6.1 Battery Consumption Implications

As presented in Chapter 3, collecting sensor data from mobile devices can have a noticeable impact on the device's battery. Some sensors, such as the *device motion* and *iBeacon Proximity* are consuming lots of processing power. Moreover, using the internet connection to periodically submit data packets to the cloud service would have an additional impact, something that is not considered in this section.

In the current study, the battery consumption of each device was also collected through the *Battery Status* sensor support of SensingKit framework [28]. For the total duration of the study, the battery drop was 0.08% per hour (SD: 0.06). That included a sensor data collection from eight sensors (*i.e.* accelerometer, gyroscope, magnetometer, device motion, heading, iBeacon™ proximity and battery status) stored into the device's memory in CSV

format. Note that in the current analysis, only features from the *device motion* sensor was used from the motion and orientation sensors, sampled in the highest sampling rate of 100Hz. According to the results reported in Section 5.5.5, equivalent results can be achieved by only using the iBeacon™ Proximity sensor.

5.6.2 Privacy Implications

Even though the method depends on using anonymous IDs when broadcasting iBeacon™ data and no other personal information is broadcast, there is always the danger that this anonymity can be compromised by tracking the openly available ID of a user. This is the reason that, according to Apple, an iOS device is only allowed to broadcast as an iBeacon™ while the device is unlocked and the app is actively running in the foreground [2].

A possible solution for protecting the user's privacy could be the use of encryption on the advertising packet, so that only authorized people or applications can make use of it. Google provides an official support of encryption in the latest Eddystone-EID frame type[6], released in April 2016. Even though not officially supported in iBeacon™ specification, third-party companies provide alternative solutions by rotating the beacon's attributes (*i.e.* Major and Minor) so that the broadcaster's ID is unpredictable[7].

5.6.3 Limitations

As mentioned earlier, an obvious limitation of the current method comes from the design of the iOS operating system, not allowing the phone to broadcast as a beacon while the device is locked. Although these limitations are software restrictions from the mobile operating systems and can change in future updates of the systems, it makes the use of an external wearable hardware essential for discovering the proximity between users, increasing the cost in a real-world implementation and application of such system.

Another limitation comes from the way that the system has been evaluated. As mentioned earlier, all important predictors do not depend on external infrastructure and the system should generalise into new environments. However, a proper evaluation with new data collected in an unknown environment is required in order to investigate this claim.

[6]https://developers.google.com/beacons/eddystone-eid
[7]https://developer.estimote.com/ibeacon/secure-uuid/

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Motivated by the lack of a modern, multi-platform continuous sensing framework, SensingKit was developed, a mobile framework that captures mobile sensor data from iOS and Android devices. SensingKit has been extensively used in all experiments of this research. Moreover, it has been used in projects from research institutes around the world, including mobile data privacy works [58, 50, 49], Quantified-Self [21], ResearchStack[1] from Cornell Tech's Small Data Lab, and Databox[2] project. Finally, it has been featured in other works in the mobile sensing research area, including [99, 16, 67, 27, 47, 39, 72, 71, 92, 69, 73, 70, 74, 41]. Compared to the existing mobile sensing frameworks that are limited to Android platform, this framework uses an almost identical API for accessing mobile sensor data from both iOS and Android environments. Moreover, it uses the latest technologies from the mobile operating systems such as sensor fusion techniques and modern sensors such as the BLE.

The framework was evaluated through an initial controlled experiment for understanding the characteristics of pedestrians that walk together in groups of two and three. More specifically, it investigated the pedestrian gait synchronisation phenomenon where the results showed that the presence of a third person highly effects their step synchronisation. This work has been featured in The Royal Institution Christmas Lectures 2017 "The Language of Life"[3]. The lecture, as presented by Prof. Sophie Scott and assisted by the writer of this thesis, showed how two children unconsciously synchronised their steps while walking. For more information about the study and its media coverage, please read the blog post at https://minoskt.github.io/blog/walking_in_sync.

A novel algorithm was developed for detecting stationary social interactions within

[1] http://researchstack.org
[2] http://databoxproject.uk
[3] http://www.rigb.org/christmas-lectures/watch/2017/the-language-of-life

crowds, named *DetecTN*. The DetecTN algorithm was presented as a first attempt for analysing mobile sensor data collected from a smartphone device with the aim to detect the social interactions that take place in a controlled environment. The algorithm was evaluated in a small case study and demonstrated in ACM MobiSys 2017 Demo session. Until now, it has been featured by two works in the area of children social behaviour analysis [61, 62]. Compared to other related works, this algorithm is capable of detecting dynamic groups of variety of sizes and is not device dependent.

A simplistic approach for making probabilistic predictions of stationary social interactions is also suggested in this thesis, named *Normalised Inverted Proximity* (NIP). In contrast to the DetecTN algorithm, it makes probabilistic predictions using the estimated proximity between two people and can be directly compared with other probabilistic solutions. NIP is suggested as a baseline in the current, but also in similar future works.

Finally, a supervised machine learning approach based on gradient-boosted trees was introduced as an extension to the DetecTN algorithm. The model uses information extracted from several mobile sensors such as Bluetooth Smart (BLE), accelerometer and gyroscope sensors. The validation was performed using a relatively large compared to other related works dataset, where 24 participants were asked to socialise with each other for 45 minutes in a natural environmental setting. By using a supervised machine learning method based on gradient-boosted trees, a performance increase of 26.7% was achieved over the NIP baseline. This work is the first to evaluate a social interaction detection method in a natural social event. Moreover, it uses some unique features from motion and orientation sensors capable of capturing the social signals that exist in conversations such as *mirroring*, when a person is subconsciously imitating the gestures or movements of another [12].

The research presented in this thesis has shown the potential of using mobile sensing techniques for measuring crowd dynamics in planned events. The implications of this research can be dedicated insights that could be monetised by event organisers willing to understand the social impact of their events, targeted advertisements with special offers targeted at groups of people (*i.e.* buy two and get one free promotions) or crowd management and crowd reconfiguration cases that are essential in critical situations such as in cases of emergency or evacuation management.

6.2 Future Work

This section provides some insight of the future work required to monitor and report events in a real-time manner.

One contribution of this work is the evaluation of the interaction detection approach in a realistic environment where different types of group configurations were naturally

created. Results from Chapter 5 reported a relative high performance in detecting the interactive state between two individuals, using the Bluetooth sensor's RSSI as a proxy for the interpersonal proximity. The use of additional sensors, such as accelerometer and gyroscope, only reported a minor performance increase. One of the main limitations of this approach is that it does not make use of features related to the relative orientation between users. The lack of such information can lead into increased false positive predictions when two non-interactive people are close in distance. In a future extension of this work, the use of magnetometer sensor will be explored as a way to measure the relative orientation between users. This extension can raise several technical challenges as detecting the user orientation from the phone requires that the device's orientation related to the user's gaze is also known.

The DetecTN algorithm presented in Chapter 4 suggested an approach for detecting the group formations inside a crowd. Inspired by the graph theory, the algorithm defines a group formation as the connected component within a graph. The main limitation of such approach is that it is not ideal for dense spaces, as any possible false positive prediction would result into larger connected components in the graph, and consequently, incorrectly identifying larger interactive groups. The use of community detection algorithms will be explored in the future as a way to cluster smaller groups inside the network. This can lead into detecting the group formations with better accuracy, improving the insight information that could be extracted from events.

Finally, a longer-term vision of this research would be a more robust group detection system that fuses mobile sensing and computer vision for analysing the behaviour of the crowd. Such system would make use of SensingKit framework embedded within already existing event apps, in order to collect in-the-wild mobile sensor data. Computer vision could be used to impute missing data sources (*i.e.* people that do not have the event app installed). A real-time version of the machine learning approach presented in Chapter 5 could be used, together with the future extensions mentioned above, in order to analyse both stationary and moving interactions and investigate how the crowd behaves in such events (*e.g.* how groups are formed or who are the main influencers in such groups).

Bibliography

[1] Nadav Aharony, Wei Pan, Cory Ip, Inas Khayal, and Alex Pentland. Social fmri: Investigating and shaping social mechanisms in the real world. *Pervasive and Mobile Computing*, 7(6):643 – 659, 2011. ISSN 1574-1192. The Ninth Annual {IEEE} International Conference on Pervasive Computing and Communications (PerCom 2011).

[2] Apple Inc. Getting Started with iBeacon v1.0. `https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf`, 2014. [Online; accessed 07-April-2018].

[3] Nilanjan Banerjee, Sharad Agarwal, Paramvir Bahl, Ranveer Chandra, Alec Wolman, and Mark Corner. *Pervasive Computing: 8th International Conference, Pervasive 2010, Helsinki, Finland, May 17-20, 2010. Proceedings*, chapter Virtual Compass: Relative Positioning to Sense Mobile Social Interactions, pages 1–21. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-12654-3.

[4] Nilanjan Banerjee, Sharad Agarwal, Paramvir Bahl, Ranveer Chandra, Alec Wolman, and Mark Corner. Virtual compass: Relative positioning to sense mobile social interactions. In Patrik Floréen, Antonio Krüger, and Mirjana Spasojevic, editors, *Pervasive Computing*, pages 1–21, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-12654-3.

[5] Stuart A. Battersby and Patrick GT Healey. Head and hand movements in the orchestration of dialogue. In *Proceedings of the 32nd Annual Conference of the Cognitive Science Society*. Cognitive Science Society, 2010.

[6] L. Bazzani, M. Cristani, and V. Murino. Decentralized particle filter for joint individual-group tracking. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1886–1893, June 2012.

[7] Bluetooth Special Interest Group (SIG). Bluetooth Core Specification v5.0. `https://www.bluetooth.com/specifications/bluetooth-core-specification`, 2016. [Online; accessed 07-April-2018].

[8] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001. ISSN 0885-6125.

[9] Francesco Calabrese, Francisco C Pereira, Giusy Di Lorenzo, Liang Liu, and Carlo Ratti. The geography of taste: Analyzing cell-phone mobility and social events. In *Pervasive Computing*, pages 22–37. Springer, 2010.

[10] Andrew T Campbell, Shane B Eisenman, Nicholas D Lane, Emiliano Miluzzo, and Ronald A Peterson. People-centric urban sensing. In *Proceedings of the 2nd annual international workshop on Wireless internet*, New York, NY, USA, August 2006. ACM.

[11] Tanya L. Chartrand and John A. Bargh. The chameleon effect: The perception-behavior link and social interaction. *Journal of Personality and Social Psychology*, 76(6):893–910, 1999.

[12] Tanya L Chartrand and John A Bargh. The chameleon effect: the perception–behavior link and social interaction. *Journal of personality and social psychology*, 76 (6):893, 1999.

[13] T. Chen and C. Guestrin. XGBoost: a scalable tree boosting system. In *Proc. of the ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 785–794, 2016.

[14] Tanzeem Choudhury and Alex Pentland. Sensing and modeling human networks using the sociometer. In *Proceedings of the 7th IEEE International Symposium on Wearable Computers*, ISWC '03, pages 216–, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-2034-0.

[15] Marco Cristani, Loris Bazzani, Giulia Paggetti, Andrea Fossati, Diego Tosato, Alessio Del Bue, Gloria Menegaz, and Vittorio Murino. Social interaction discovery by statistical analysis of f-formations. In *Proceedings of the British Machine Vision Conference*, pages 23.1–23.12. BMVA Press, 2011. ISBN 1-901725-43-X.

[16] Roshan Bharath Das, Aart van Halteren, and Henri Bal. Swan-fly: A flexible cloud-enabled framework for context-aware applications in smartphones. In *Sensors to Cloud Architectures Workshop (SCAW-2016), held in conjunction with HPCA-22*, 2016.

[17] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[18] Naiyang Deng, Yingjie Tian, and Chunhua Zhang. *Support Vector Machines: Optimization Based Theory, Algorithms, and Extensions*. Chapman & Hall/CRC, 1st edition, 2012. ISBN 143985792X, 9781439857922.

[19] Ajay Gandhi and Lotte Hoek. Introduction to crowds and conviviality: Ethnographies of the south asian city. *Ethnography*, 13(1):3–11, 2012.

[20] Enrique Garcia-Ceja, Venet Osmani, Alban Maxhuni, and Oscar Mayora. Detecting walking in synchrony through smartphone accelerometer and wi-fi traces. In Emile Aarts, Boris de Ruyter, Panos Markopoulos, Evert van Loenen, Reiner Wichert, Ben Schouten, Jacques Terken, Rob Van Kranenburg, Elke Den Ouden, and Gregory O'Hare, editors, *Ambient Intelligence*, volume 8850 of *Lecture Notes in Computer Science*, pages 33–46. Springer International Publishing, 2014. ISBN 978-3-319-14111-4.

[21] H. Haddadi, F. Ofli, Y. Mejova, I. Weber, and J. Srivastava. 360-degree quantified self. In *2015 International Conference on Healthcare Informatics*, pages 587–592, Oct 2015.

[22] Edward Twitchell Hall. The hidden dimension. 1966.

[23] Dirk Helbing and Péter Molnár. Social force model for pedestrian dynamics. *Phys. Rev. E*, 51:4282–4286, May 1995.

[24] S. A. Hoseinitabatabaei, A. Gluhak, and R. Tafazolli. udirect: A novel approach for pervasive observation of user direction with mobile phones. In *Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on*, pages 74–83, March 2011.

[25] W. Hu, G. Cao, S. V. Krishanamurthy, and P. Mohapatra. Mobility-assisted energy-aware user contact detection in mobile social networks. In *2013 IEEE 33rd International Conference on Distributed Computing Systems*, pages 155–164, July 2013.

[26] Hayley Hung and Ben Kröse. Detecting f-formations as dominant sets. In *Proceedings of the 13th International Conference on Multimodal Interfaces*, ICMI '11, pages 231–238, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0641-6.

[27] Muhammad Irfan, Lucio Marcenaro, and Laurissa Tokarchuk. Crowd analysis using visual and non-visual sensors, a survey. In *Signal and Information Processing (GlobalSIP), 2016 IEEE Global Conference on*, pages 1249–1254. IEEE, 2016.

[28] Kleomenis Katevas, Hamed Haddadi, and Laurissa Tokarchuk. Poster: Sensingkit: A multi-platform mobile sensing framework for large-scale experiments. In *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*,

MobiCom '14, pages 375–378, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2783-1.

[29] Kleomenis Katevas, Patrick GT Healey, and Matthew Tobias Harris. Robot stand-up: engineering a comic performance. In *Proceedings of the Workshop on Humanoid Robots and Creativity at the IEEE-RAS International Conference on Humanoid Robots Humanoids (Madrid)*, 2014.

[30] Kleomenis Katevas, Hamed Haddadi, Laurissa Tokarchuk, and Richard G. Clegg. Walking in sync: Two is company, three's a crowd. In *Proceedings of the 2Nd Workshop on Workshop on Physical Analytics*, WPA '15, pages 25–29, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3498-3.

[31] Kleomenis Katevas, Patrick GT Healey, and Matthew Tobias Harris. Robot comedy lab: experimenting with the social dynamics of live performance. *Frontiers in Psychology*, 6:1253, 2015.

[32] Kleomenis Katevas, Hamed Haddadi, and Laurissa Tokarchuk. Sensingkit: Evaluating the sensor power consumption in ios devices. In *Intelligent Environments (IE), 2016 12th International Conference on*, pages 222–225. IEEE, 2016.

[33] Kleomenis Katevas, Hamed Haddadi, Laurissa Tokarchuk, and Richard G. Clegg. Detecting group formations using iBeacon technology. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, UbiComp '16, pages 742–752, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4462-3.

[34] Kleomenis Katevas, Ilias Leontiadis, Martin Pielot, and Joan Serrà. Continual prediction of notification attendance with classical and deep network approaches. *arXiv preprint arXiv:1712.07120*, 2017.

[35] Kleomenis Katevas, Ilias Leontiadis, Martin Pielot, and Joan Serrà. Practical processing of mobile sensor data for continual deep learning predictions. In *Proceedings of the 1st International Workshop on Deep Learning for Mobile Systems and Applications*, pages 19–24. ACM, 2017.

[36] Kleomenis Katevas, Laurissa Tokarchuk, Hamed Haddadi, Richard G. Clegg, and Muhammad Irfan. Demo: Detecting group formations using ibeacon technology. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '17, pages 190–190, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4928-4.

[37] Kleomenis Katevas, Martin Pielot, and Ioannis Arapakis. Typical phone use habits: Intense use does not predict negative well-being. In *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services*. ACM, 2018.

[38] Adam Kendon. *Conducting interaction: Patterns of behavior in focused encounters*, volume 7. CUP Archive, 1990.

[39] Jong W Kim, Robert A Sottilare, Gregory Goodwin, and Xiangen Hu. Assessment c. *Design Recommendations for Intelligent Tutoring System-Volume 5: Assessment Methods*, 5:319, 2017.

[40] Mikkel Baun Kjærgaard, Martin Wirz, Daniel Roggen, and Gerhard Tröster. Detecting pedestrian flocks by fusion of multi-modal sensors in mobile phones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pages 240–249, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1224-0.

[41] Seungho Kuk, Junha Kim, Yongtae Park, and Hyogon Kim. Empirical determination of efficient sensing frequencies for magnetometer-based continuous human contact monitoring. *Sensors (Basel, Switzerland)*, 18(5), 2018.

[42] Daniël Lakens and Mariëlle Stel. If they move in sync, they must feel in sync: Movement synchrony leads to attributions of rapport and entitativity. *Social Cognition*, 29(1):1–14, 2011.

[43] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell. A survey of mobile phone sensing. *IEEE Communications Magazine*, 48(9):140–150, Sept 2010. ISSN 0163-6804.

[44] Nicholas D Lane, Shane B Eisenman, Mirco Musolesi, Emiliano Miluzzo, and Andrew T Campbell. *Urban sensing systems: opportunistic or participatory?* opportunistic or participatory? ACM, New York, New York, USA, February 2008.

[45] Jakob Eg Larsen, Piotr Sapiezynski, Arkadiusz Stopczynski, Morten Mørup, and Rasmus Theodorsen. Crowds, bluetooth, and rock'n'roll: understanding music festival participant behavior. In *PDM '13: Proceedings of the 1st ACM international workshop on Personal data meets distributed multimedia*, pages 11–18, New York, New York, USA, October 2013. ACM Request Permissions.

[46] S. Liu, Y. Jiang, and A. Striegel. Face-to-face proximity estimation using bluetooth on smartphones. *IEEE Transactions on Mobile Computing*, 13(4):811–823, April 2014. ISSN 1536-1233.

[47] Derrick Allen Lo. *Sensor Plot Kit: An iOS Framework for Real-time plotting of Wireless Sensors*. University of California, Irvine, 2015.

[48] Anmol Madan, Manuel Cebrian, David Lazer, and Alex Pentland. Social sensing for epidemiological behavior change. In *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*, UbiComp '10, pages 291–300, New York, NY, USA, September 2010. ACM.

[49] Mohammad Malekzadeh, Richard G Clegg, Andrea Cavallaro, and Hamed Haddadi. Privacy-preserving sensor data analysis for edge computing.

[50] Mohammad Malekzadeh, Richard G Clegg, Andrea Cavallaro, and Hamed Haddadi. Protecting sensory data against sensitive inferences. In *Proceedings of the 1st Workshop on Privacy by Design in Distributed Systems*, page 2. ACM, 2018.

[51] A. Matic, A. Papliatseyeu, V. Osmani, and O. Mayora-Ibarra. Tuning to your position: Fm radio based indoor localization with spontaneous recalibration. In *Pervasive Computing and Communications (PerCom), 2010 IEEE International Conference on*, pages 153–161, March 2010.

[52] A. Matic, V. Osmani, A. Maxhuni, and O. Mayora. Multi-modal mobile sensing of social interactions. In *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2012 6th International Conference on*, pages 105–114, May 2012.

[53] R. Mazzon, F. Poiesi, and A. Cavallaro. Detection and tracking of groups in crowd. In *Advanced Video and Signal Based Surveillance (AVSS), 2013 10th IEEE International Conference on*, pages 202–207, Aug 2013.

[54] R Mehran, A Oyama, and M Shah. Abnormal crowd behavior detection using social force model. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 935–942. IEEE, 2009.

[55] Erik Meijering. A chronology of interpolation: From ancient astronomy to modern signal and image processing. *Proceedings of the IEEE*, 90(3):319–342, 2002.

[56] Y. Miyake. Interpersonal synchronization of body motion and the walk-mate walking support robot. *Robotics, IEEE Transactions on*, 25(3):638–644, June 2009. ISSN 1552-3098.

[57] Alessandro Montanari, Zhao Tian, Elena Francu, Benjamin Lucas, Brian Jones, Xia Zhou, and Cecilia Mascolo. Measuring interaction proxemics with wearable light tags. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(1):25, 2018.

[58] Richard Mortier, Jianxin Zhao, Jon Crowcroft, Liang Wang, Qi Li, Hamed Haddadi, Yousef Amar, Andy Crabtree, James Colley, Tom Lodge, Tosh Brown, Derek McAuley, and Chris Greenhalgh. Personal data management with the databox: What's inside the box? In *Proceedings of the 2016 ACM Workshop on Cloud-Assisted Networking*, CAN '16, pages 49–54, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4673-3.

[59] Mehdi Moussaïd, Niriaska Perozo, Simon Garnier, Dirk Helbing, and Guy Theraulaz. The walking behaviour of pedestrian social groups and its impact on crowd dynamics. *PloS one*, 5(4):e10047, 2010.

[60] Roderick Murray-Smith, Andrew Ramsay, Simon Garrod, Melissa Jackson, and Bojan Musizza. Gait alignment in mobile phone conversations. In *Proceedings of the 9th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '07, pages 214–221, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-862-6.

[61] Shuta Nakamae, Shumpei Kataoka, Can Tang, Yue Pu, Simona Vasilache, Satoshi Saga, Buntarou Shizuki, and Shin Takahashi. Ble-based children's social behavior analysis system for crime prevention. In *International Conference on Social Computing and Social Media*, pages 429–439. Springer, 2017.

[62] Shuta Nakamae, Shumpei Kataoka, Can Tang, Simona Vasilache, Satoshi Saga, Buntarou Shizuki, and Shin Takahashi. Children's social behavior analysis system using ble and accelerometer. In *International Conference on Collaboration Technologies*, pages 153–167. Springer, 2017.

[63] John Ashworth Nelder and R Jacob Baker. *Generalized linear models*. Wiley Online Library, 1972.

[64] Great Britain Cabinet Office, Rose Challenger, Chris W Clegg, and Mark Robinson. Understanding crowd behaviours, April 2010.

[65] N. Palaghias, S. A. Hoseinitabatabaei, M. Nati, A. Gluhak, and K. Moessner. Accurate detection of real-world social interactions with smartphones. In *Communications (ICC), 2015 IEEE International Conference on*, pages 579–585, June 2015.

[66] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[67] Jia Peng, Yanmin Zhu, Wei Shu, and Min-You Wu. How multiple crowdsourcers compete for smartphone contributions? In *2015 IEEE Conference on Computer Communications Workshops*, pages 516–521. IEEE, April 2015.

[68] Martin Pielot, Bruno Cardoso, Kleomenis Katevas, Joan Serrà, Aleksandar Matic, and Nuria Oliver. Beyond interruptibility: Predicting opportune moments to engage mobile phone users. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(3):91, 2017.

[69] Ivan Miguel Pires, Nuno M Garcia, Nuno Pombo, and Francisco Flórez-Revuelta. A multiple source framework for the identification of activities of daily living based on mobile device data. *arXiv preprint arXiv:1711.00104*, 2017.

[70] Ivan Miguel Pires, Nuno M Garcia, Nuno Pombo, and Francisco Flórez-Revuelta. User environment detection with acoustic sensors embedded on mobile devices for the recognition of activities of daily living. *arXiv preprint arXiv:1711.00124*, 2017.

[71] Ivan Miguel Pires, Nuno M Garcia, Nuno Pombo, Francisco Flórez-Revuelta, and Susanna Spinsante. Data fusion on motion and magnetic sensors embedded on mobile devices for the identification of activities of daily living. *arXiv.org*, 2017.

[72] Ivan Miguel Pires, Nuno M Garcia, Nuno Pombo, Francisco Flórez-Revuelta, and Susanna Spinsante. Pattern recognition techniques for the identification of activities of daily living using mobile device accelerometer. *arXiv preprint arXiv:1711.00096*, 2017.

[73] Ivan Miguel Pires, Nuno M Garcia, Nuno Pombo, Francisco Flórez-Revuelta, and Susanna Spinsante. Approach for the development of a framework for the identification of activities of daily living using sensors in mobile devices. *Sensors*, 18(2):640, 2018.

[74] Ivan Miguel Pires, Nuno M Garcia, Nuno Pombo, Francisco Flórez-Revuelta, Susanna Spinsante, and Maria Canavarro Teixeira. Identification of activities of daily living through data fusion on motion and magnetic sensors embedded on mobile devices. *Pervasive and Mobile Computing*, 2018.

[75] Kiran K. Rachuri, Mirco Musolesi, Cecilia Mascolo, Peter J. Rentfrow, Chris Longworth, and Andrius Aucinas. Emotionsense: A mobile phones based adaptive platform for experimental social psychology research. In *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*, UbiComp '10, pages 281–290, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-843-8.

[76] Radius Networks. Fundamentals of Beacon Ranging. `http://developer.radiusnetworks.com/2014/12/04/fundamentals-of-beacon-ranging.html`, 2014. [Online; accessed 19-April-2016].

[77] Valentin Radu, Panagiota Katsikouli, Rik Sarkar, and Mahesh K. Marina. A semi-supervised learning approach for robust indoor-outdoor detection with smartphones. In *SenSys '14: Proceedings of the International Conference on Embedded Networked Sensor Systems*. ACM, 2014.

[78] Jacek Rapiński, Daniel Zinkiewicz, and Tomasz Stanislawek. Influence of human body on radio signal strength indicator readings in indoor positioning systems. *Technical Sciences/University of Warmia and Mazury in Olsztyn*, (19 (2)):117–127, 2016.

[79] Stuart Reeves, Scott Sherwood, and Barry Brown. Designing for crowds. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, NordiCHI '10, pages 393–402, New York, NY, USA, October 2010.

[80] Martin S. Remland, Tricia S. Jones, and Heidi Brinkman. Interpersonal distance, body orientation, and touch: Effects of culture, gender, and age. *The Journal of Social Psychology*, 135(3):281–297, 1995.

[81] Michael J Richardson, Kerry L Marsh, and RC Schmidt. Effects of visual and verbal interaction on unintentional interpersonal coordination. *Journal of Experimental Psychology: Human Perception and Performance*, 31(1):62, 2005.

[82] Daniel Roggen, Martin Wirz, Gerhard Tröster, and Dirk Helbing. Recognition of crowd behavior from mobile sensors with pattern analysis and graph clustering methods. *arXiv.org*, September 2011.

[83] David Sachs. Sensor Fusion on Android Devices: A Revolution in Motion Processing, August 2010.

[84] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3):e0118432, 2015.

[85] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[86] Kevin Shockley, Marie-Vee Santana, and Carol A Fowler. Mutual interpersonal postural constraints are involved in cooperative conversation. *Journal of Experimental Psychology: Human Perception and Performance*, 29(2):326, 2003.

[87] J. Sochman and D. C. Hogg. Who knows who - inverting the social force model for finding groups. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 830–837, Nov 2011.

[88] M B Srivastava, J A Burke, and M Hansen. Network system challenges in selective sharing and verification for personal, social, and urban-scale sensing applications. 2006.

[89] A Stopczynski, J E Larsen, S Lehmann, L Dynowski, and M Fuentes. Participatory bluetooth sensing: A method for acquiring spatio-temporal data about participant mobility and interactions at large scale events. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*, pages 242–247. IEEE, 2013.

[90] Clifford Stott and John Drury. Crowds, context and identity: Dynamic categorization processes in the'poll tax riot'. *Human relations*, 53(2):247–273, 2000.

[91] X. Su, H. Tong, and P. Ji. Activity recognition with smartphone sensors. *Tsinghua Science and Technology*, 19(3):235–249, June 2014.

[92] Xing Su. *Travel Mode Identification with Smartphone Sensors*. City University of New York, 2017.

[93] Dirk Trossen and Dana Pavel. Airs: A mobile sensing platform for lifestyle management research and applications. In *Mobile Wireless Middleware, Operating Systems, and Applications*, pages 1–15, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[94] M Wirz, T Franke, D Roggen, E Mitleton-Kelly, P Lukowicz, and G Troster. Inferring crowd conditions from pedestrians' location traces for real-time crowd monitoring during city-scale mass gatherings. In *2012 IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 367–372, 2012.

[95] M Wirz, T Franke, E Mitleton-Kelly, and D Roggen. Coenosense: A framework for real-time detection and visualization of collective behaviors in human crowds by tracking mobile devices. In *Proceedings of the European Conference on Complex Systems 2012*, pages 353–361. Springer International Publishing, 2013.

[96] Martin Wirz, Pablo Schläpfer, Mikkel Baun Kjærgaard, Daniel Roggen, Sebastian Feese, and Gerhard Tröster. Towards an online detection of pedestrian flocks in urban canyons by smoothed spatio-temporal clustering of gps trajectories. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Location-Based Social Networks*, pages 17–24, New York, NY, USA, November 2011. ACM.

[97] Peter Wittenburg, Hennie Brugman, Albert Russel, Alex Klassmann, and Han Sloetjes. Elan: a professional framework for multimodality research. In *Proceedings of LREC*, volume 2006, page 5th, 2006.

[98] Pang Wu, Jiang Zhu, and Joy Ying Zhang. Mobisens: A versatile mobile sensing platform for real-world applications. *Mobile Networks and Applications*, 18(1):60–80, 2012. ISSN 1572-8153.

[99] Haoyi Xiong, Yu Huang, Laura E Barnes, and Matthew S Gerber. Sensus: a cross-platform, general-purpose system for mobile crowdsensing in human-subject studies. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 415–426. ACM, 2016.

Appendix A

QMUL Ethics Committee Approval

The following approvals were received from Queen Mary Research Ethics Committee on the 18th February 2015, 25th August 2015 and 27th April 2018.

Queen Mary
University of London

Queen Mary, University of London
Room W117
Queen's Building
Queen Mary University of London
Mile End Road
London E1 4NS

Queen Mary Ethics of Research Committee
Hazel Covill
Research Ethics Administrator
Tel: +44 (0) 20 7882 7915
Email: h.covill@qmul.ac.uk

c/o Dr Laurissa Tokarchuk
CS 302, Peter Landin Building
Department of Engineering and Computer Science
Queen Mary University of London
Mile End
London

18th February 2015

To Whom It May Concern:

Re: QMERC1422a – Analysing walking behaviours using mobile sensing technology.

I can confirm that Mr Kleomenis Katevas has completed a Research Ethics Questionnaire with regard to the above research.

The result of which was the conclusion that his proposed work does not present any ethical concerns: is extremely low risk; and thus does not require the scrutiny of the full Research Ethics Committee.

Yours faithfully

Ms Hazel Covill – QMERC Administrator

Patron: Her Majesty the Queen
Incorporated by Royal Charter as Queen Mary and Westfield College, University of London

Queen Mary, University of London
Room W117
Queen's Building
Queen Mary University of London
Mile End Road
London E1 4NS

Queen Mary Ethics of Research Committee
Hazel Covill
Research Ethics Administrator
Tel: +44 (0) 20 7882 7915
Email: h.covill@qmul.ac.uk

c/o Dr Laurissa Tokarchuk
CS302, Peter Landin Building
Department of Computer Science
Queen Mary University of London
Mile End Road
London

25th August 2015

To Whom It May Concern:

Re: QMREC1543a – Identify and model different qualities of social interactions inside crowds.

I can confirm that Mr Kleomenis Katevas has completed a Research Ethics Questionnaire with regard to the above research.

The result of which was the conclusion that his proposed work does not present any ethical concerns; is extremely low risk; and thus does not require the scrutiny of the full Research Ethics Committee.

Yours faithfully

Ms Hazel Covill – QMERC Administrator

Patron: Her Majesty the Queen
Incorporated by Royal Charter as Queen Mary and Westfield College, University of London

Queen Mary, University of London
Room W117
Queen's Building
Queen Mary University of London
Mile End Road
London E1 4NS

Queen Mary Ethics of Research Committee
Hazel Covill
Research Ethics Administrator
Tel: +44 (0) 20 7882 7915
Email: h.covill@qmul.ac.uk

c/o Dr Laurissa Tokarchuk
CS 302, Peter Landin Building
School of Electronic Engineering
and Computer Science
Queen Mary University of London
Mile End
London

27th April 2018

To Whom It May Concern:

Re: QMREC1842 - Proximity Estimation using iBeacons.

I can confirm that Kleomenis Katevas has completed a Research Ethics Questionnaire with regard to the above research.

The result of which was the conclusion that his proposed work does not present any ethical concerns; is extremely low risk; and thus does not require the scrutiny of the full Research Ethics Committee.

Yours faithfully

Mr Jack Biddle – Research Approvals Advisor

Patron: Her Majesty the Queen
Incorporated by Royal Charter as Queen Mary and Westfield College, University of London

Appendix B

Featured Work

This appendix lists the research projects, articles and conferences that parts of this Ph.D. research has been featured in.

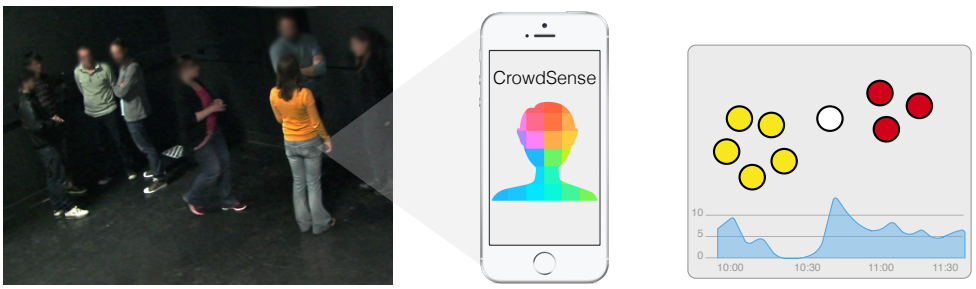
B.1 SensingKit framework

SensingKit framework, reported in Chapter 3, has been used in various research projects, including a Quantified Self application [21], as well as other M.Sc. and Ph.D. works at Queen Mary University of London. ResearchStack[1], a Software-Development-Kit (SDK) and User-Experience (UX) framework for Android-based research study apps, listed SensingKit as the official supported framework for accessing mobile sensor data. Databox project [58], a privacy-aware infrastructure for managing personal data natively support SensingKit through their SensingKit Driver[2]. Finally, SensingKit was recently featured in a new Wikipedia article titled "Mobile Phone Based Sensing Software"[3].

B.2 Walking in Sync work

The work presented in Section 3.5 has been featured in The Royal Institution Christmas Lectures 2017 – The Language of Life[4], in Part 2: Silent Messages. The lecture was presented by Prof. Sophie Scott at BBC Four on 27th December 2017.

[1] http://researchstack.org
[2] https://github.com/me-box/driver-sensingkit
[3] https://en.wikipedia.org/wiki/Mobile_phone_based_sensing_software
[4] http://www.rigb.org/christmas-lectures/watch/2017/the-language-of-life



a. Example of a social event b. CrowdSense App c. Back-end

Figure B.1: System Design of the demo presented in ACM MobiSys 2017.

B.3 Group Detection Algorithm

A real-time implementation of the group detection algorithm presented in Chapter 4 was demonstrated in ACM MobiSys 2017 that took place at the Niagara Falls, New York in June 2017 [36] (Figure B.1). The purpose of this demo was to present a technology that is capable of providing in-depth analytics about how the crowd behaves during social events.

Participants interested in this demo installed CrowdSense App for iOS into their iPhones, provided their name when requested by the app and started socialising with each-other. A back-end system analysed the streamed data using the same algorithm that was explained earlier in this section, detected the users' social interactions in real-time and visualised them on a screen as a graph through a web-interface.

Beside the real-time graph visualisation of the interactions, the web-interface provided the following analytics:

- A time-series figure that shows the number of social interactions taking place over time.
- A frequency diagram that shows the type of interactions (one-to-on, groups of three, four etc.) from the beginning of the social event.

Since the purpose of this demonstration was to present a real-time implementation of this system in a social event, no data collection was conducted, and thus, it is not possible to provide the analytics of this specific event.

Appendix C

Proximity Evaluation Plots

This appendix completes the results from the proximity evaluation of iBeacon™ signal, presented in Section 5.2.1. Each distance (x axis) was maintained for 30 seconds while P1 (broadcaster) had eight coin-shaped beacons attached to the front of his body. P2 (receiver) was collecting the beacon's RSSI signal using an iPhone SE device placed on his left pocket.

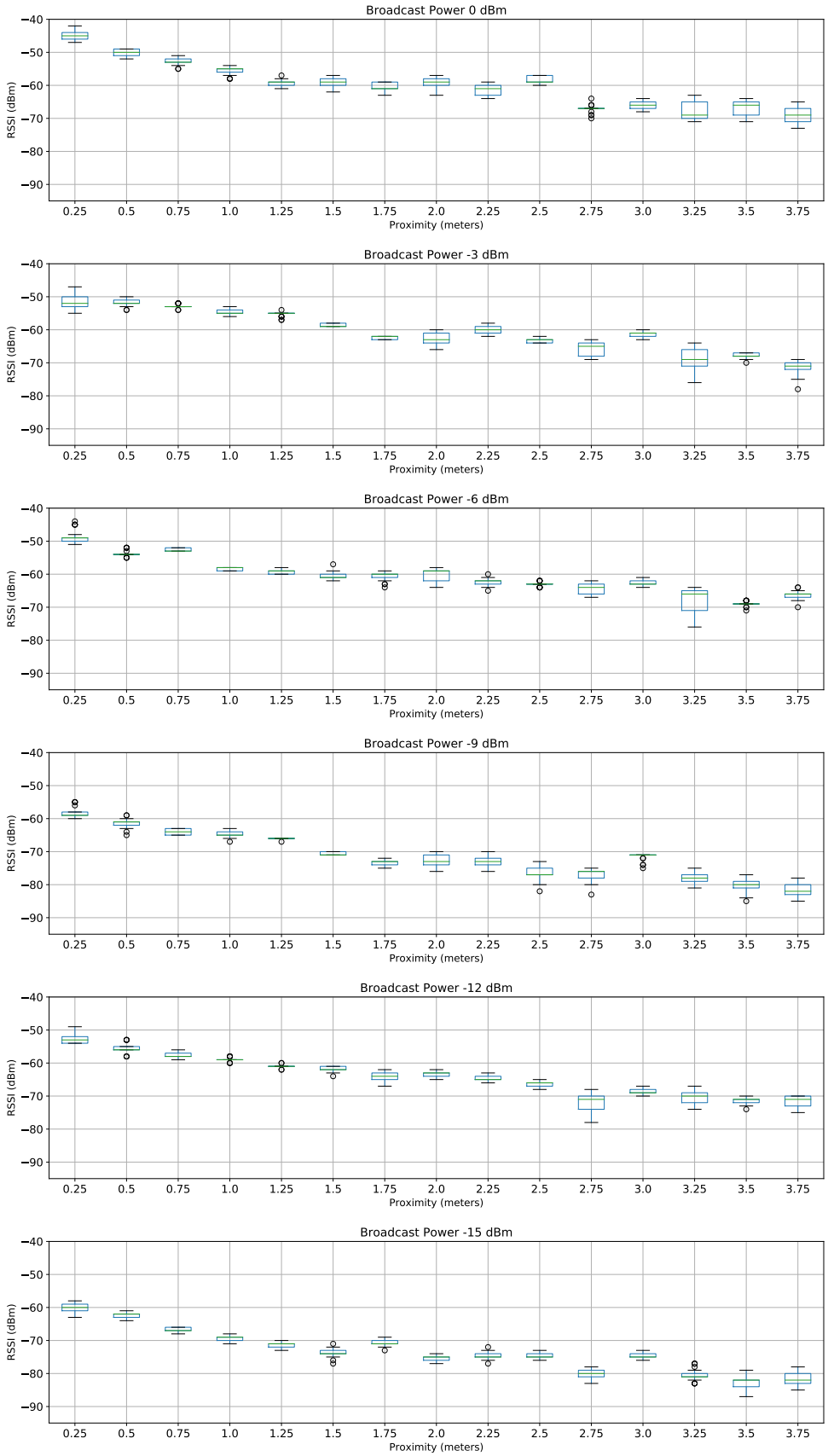


Figure C.1: Measuring the effect of proximity on iBeacon™ signal.

Appendix D

Model Tuning Performance

This appendix lists the results from the parameter tuning procedure required for choosing the most optimal model for the binary classification problem presented in Chapter 5.

Table D.1: Parameter Tuning for Logistic Regression model, using the Average Precision (AP) as a performance metric.

C			
1	*10*	*100*	*1000*
0.559	0.551	0.542	**0.566**

Table D.2: Parameter Tuning for Random Forest model, using the Average Precision (AP) as a performance metric.

		max_depth			
		4	*6*	*8*	*10*
max_features	*'sqrt'*	0.792	0.761	0.763	0.748
	'log2'	**0.826**	0.787	0.762	0.744
	'None'	0.628	0.612	0.613	0.602

Table D.3: Parameter Tuning for XGBoost model, using the Average Precision (AP) as a performance metric.

		max_depth			
		4	*6*	*8*	*10*
colsample_bytree	*0.2*	0.816	0.821	0.823	**0.830**
	0.4	0.810	0.782	0.802	0.803
	0.6	0.801	0.775	0.781	0.769
	0.8	0.817	0.767	0.773	0.771
	1	0.814	0.735	0.767	0.759

Table D.4: Detailed Parameter Tuning for XGBoost model, using the Average Precision (AP) as a performance metric.

				learning_rate											
				0.01				*0.05*				*0.1*			
				max_depth											
				4	*6*	*8*	*10*	*4*	*6*	*8*	*10*	*4*	*6*	*8*	*10*
subsample	*0.5*	colsample_bytree	*0.2*	0.825	0.815	0.812	0.801	**0.838**	0.823	0.819	0.813	0.835	0.807	0.807	0.816
			0.4	0.791	0.790	0.779	0.773	0.817	0.811	0.801	0.791	0.791	0.795	0.815	0.798
			0.6	0.787	0.759	0.742	0.736	0.813	0.791	0.754	0.759	0.795	0.781	0.761	0.783
			0.8	0.721	0.748	0.728	0.712	0.812	0.790	0.761	0.755	0.798	0.758	0.770	0.770
			1	0.733	0.740	0.717	0.713	0.814	0.789	0.753	0.734	0.793	0.784	0.777	0.755
	0.75		*0.2*	0.829	0.816	0.807	0.804	0.834	0.824	0.822	0.819	0.835	0.813	0.821	0.826
			0.4	0.791	0.787	0.769	0.763	0.803	0.806	0.796	0.783	0.795	0.809	0.814	0.812
			0.8	0.772	0.751	0.737	0.725	0.808	0.777	0.744	0.735	0.795	0.768	0.781	0.779
			0.6	0.728	0.707	0.694	0.681	0.811	0.778	0.738	0.739	0.809	0.754	0.763	0.764
			1	0.753	0.734	0.687	0.686	0.810	0.769	0.734	0.721	0.804	0.750	0.763	0.762
	1		*0.2*	0.814	0.818	0.820	0.815	0.831	0.830	0.830	0.838	0.816	0.821	0.823	0.830
			0.4	0.792	0.789	0.769	0.762	0.823	0.791	0.790	0.778	0.810	0.782	0.802	0.803
			0.6	0.783	0.746	0.735	0.715	0.822	0.777	0.747	0.725	0.801	0.775	0.781	0.769
			0.8	0.720	0.732	0.705	0.685	0.812	0.767	0.740	0.718	0.817	0.767	0.773	0.771
			1	0.707	0.634	0.574	0.572	0.809	0.764	0.728	0.720	0.814	0.735	0.767	0.759

Appendix E

SensingKit Developer Instructions

As described in Chapter 3, SensingKit library can be embedded in any iOS or Android application. This Appendix provides information on how to build and import the library to a mobile application. Furthermore, it includes basic information of how to use the library. A detailed description of the Application Programming Interface (API) of SensingKit is available at https://www.sensingkit.org.

E.1 SensingKit for Android

Building and Configuring the Library

Build the library using the command:

```
./gradlew build
```

Create an *app/libs* directory inside your project and copy the generated *SensingKitLib/build/outputs/aar/SensingKitLib.aar* file there. Edit your *app/build.gradle* file and add a *flatDir* entry as shown below:

```
repositories {
    mavenCentral()
    flatDir {
        dirs 'libs'
    }
}
```

In the same *app/build.gradle* file, add *SensingKitLib* as a dependency as shown below:

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
```

```
    implementation 'org.sensingkit:SensingKitLib-release@aar'
    implementation 'com.google.android.gms:play-services-location:16.0.0'
}
```

Using the Library

The following code listing shows how you can import and initialise SensingKit into your class.

```
import org.sensingkit.sensingkitlib.*;
import org.sensingkit.sensingkitlib.data.*;

public class MainActivity extends AppCompatActivity {

    SensingKitLibInterface mSensingKitLib;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        try {
            // Init SensingKit.
            // 'this' keyword refers to the Application Context.
            mSensingKitLib = SensingKitLib.sharedSensingKitLib(this);
        }
        catch (SKException ex) {
            // Handle Exception
        }
    }
}
```

The steps that you need to follow for sensing data using SensingKit are: Check for sensor availability, register the sensor, subscribe a sensor data listener and start sensing. The following code listing describes that process, using the *Battery Status* sensor as an example:

```
if (mSensingKitLib.isSensorAvailable(SKSensorType.BATTERY_STATUS)) {

    // Register sensor
    mSensingKitLib.registerSensor(SKSensorType.BATTERY_STATUS);

    // Subscribe a sensor data listener
```

```
    mSensingKitLib.subscribeSensorDataHandler(SKSensorType.BATTERY_STATUS,
                                              new SKSensorDataHandler() {
            @Override
            public void onDataReceived(final SKSensorType moduleType,
                                       final SKSensorData sensorData) {

                // Cast sensor data as SKBatteryStatusData
                SKBatteryStatusData batteryStatusData =
                    (SKBatteryStatusData)sensorData;

                // Print the battery level
                System.out.println("Battery Level: " +
                    batteryStatusData.getLevel());
            }
        });

    // Start sensing with Battery Status sensor
    mSensingKitLib.startContinuousSensingWithSensor(
                                       SKSensorType.BATTERY_STATUS);
}
```

E.2 SensingKit for iOS

Installing the Library

You can easily install SensingKit using CocoaPods, a popular dependency manager for Cocoa projects. For installing CocoaPods, use the following command:

```
$ gem install cocoapods
```

To integrate SensingKit into your Xcode project, specify it in your Podfile:

```
target <MyApp> do
  use_frameworks!

  pod 'SensingKit'
  # For the latest development version, please use:
  # pod 'SensingKit', :git =>
     'https://github.com/SensingKit/SensingKit-iOS.git', :branch => 'next'

end
```

Then, run the following command:

```
$ pod install
```

For more information about CocoaPods, visit https://cocoapods.org.

Using the Library

The following code listing shows how you can import and initialise SensingKit into an iOS app:

```
#import <SensingKit/SensingKit.h>

@property (nonatomic, strong) SensingKitLib *sensingKit;

- (void)viewDidLoad {
    [super viewDidLoad];

    // Get a singleton instance of SensingKit framework
    self.sensingKit = [SensingKitLib sharedSensingKitLib];
}
```

The steps that you need to follow for sensing data using SensingKit are: Check for sensor availability, register the sensor, subscribe a sensor data handler and start sensing. The following code listing describes that process, using the *Battery Status* sensor as an example:

```
if ([self.sensingKit isSensorAvailable:BatteryStatus]) {

    // Register sensor
    [self.sensingKit registerSensor:BatteryStatus];

    // Subscribe a sensor data handler
    [self.sensingKit subscribeToSensor:BatteryStatus
                      withHandler:^(SKSensorType sensorType,
                                    SKSensorData *sensorData,
                                    NSError *error) {

        if (!error) {
            // Cast sensor data as SKBatteryData
            SKBatteryData *batteryData = (SKBatteryData *)sensorData;

            // Print the battery level
            NSLog(@"Battery Level: %f", batteryData.level);
        }
    } error:NULL];
```

```
    // Start sensing with Battery Status sensor
    [self.sensingKit startContinuousSensingWithSensor:BatteryStatus
                                                error:NULL];
}
```

Appendix F

Source Code

The complete source-code of SensingKit framework and CrowdSense App for iOS and Android platforms is available under the GNU Lesser General Public License v3.0 at https://github.com/SensingKit. A detailed description of its application programming interface (API) can be found at https://www.sensingkit.org.