1506 UNIVERSITÀ DEGLI STUDI DI URBINO CARLO BO

Dipartimento di Scienze Pure e Applicate
Scuola di Scienze e Tecnologie dell'Informazione

Ph.D. Thesis

Mobile Crowd Sensing: Enabling Technologies and Applications

Tutor:
Chiar.mo. Prof. Alessandro Bogliolo

Candidate:
Dott. Saverio Delpriori

Dottorato in Scienze di Base e Applicazioni
curriculum
Scienze della Complessità
Ciclo XXX — A.A. 2016/2017
Settore Scientifico Disciplinare INF/01

Contents

List of Figures

Chapter 1

Introduction

Today we repeatedly join and leave dynamic and complex systems such as online communities, transportation networks, and cities. Most of the time we are not aware of behaving as linked nodes in massive networks, like traffic flows or economic systems. But fast-evolving systems like these need to be continuously monitored in order to extract information essential to decision making, proactive surveillance, and policy making.

The process of handling complex systems starts with the non-trivial task of collecting data about, analyzing, and understanding complex *contexts*. Until about a decade ago, the most suitable methods for examining these contexts took advantage of tools like Wireless Sensor Networks (WSNs) and emerging phenomena such as the Internet of Things (IoT). Although being innovative and on the cutting edge from a technological point of view, they were not flexible enough to provide building blocks of applications able to deal with contexts highly dynamic in both space and time dimensions. Moreover, even if such technologies would have been of great help in mining knowledge from complex networks, some shortcomings (e.g., high installation and maintenance cost, lack of scalability, steep learning curve, and limited node coverage) have hindered their full commercial exploitation and their pervasive diffusion in many real world applications.

With the recent widespread availability of smartphones, this scenario has radically changed. Having a full fledged computer with high connectivity and computing power but small enough to be carried around with no efforts by users, has enabled a new fast-growing sensing paradigm. Not only smartphones, but also smart vehicles and wearable devices — along side with further developments in cloud computing — offered an interesting opportunity for developing distributed applications able to quickly and precisely investigate even dynamic and quite complex phenomena.

Nowadays, standard smart device, and smartphones, in particular, sport a considerable number of embedded sensors. The capability to adequately *sense* the environment is paramount in collecting well-grounded data and has a pivotal role in any application which aims to understand a real-world event. Nonetheless, from the crowdsensing perspective, the main contribution of the smartphones diffusion doesn't lie in the introduction of new and more precise types of sensors. In general terms, having two sensing networks, one composed of motes[1] and the other one made of smartphones, we can easily identify some advantages of the second approach. Smart devices have much greater computing power which can be leveraged to filter or pre-process data collected before sending them to the application server. Connectivity has always been a major issue especially for ad hoc networks of motes with limited communication range, while even entry-level smartphones can use many different technologies to communicate. Having many connectivity interfaces such as Bluetooth, Wi-Fi, NFC, and broadband mobile network capabilities makes them capable not only to easily communicate with peers or with a central server but also enables them to act as a bridge between other connected devices with limited connection capabilities.

One of the main issues of traditional approaches was the inability to follow the dynamic changes (in terms of context and geographical position) due to limited ability to move toward or to follow the phenomenon. Most sensing applications aim to acquire knowledge about an event where the user is involved in some way or about the environment where the user spends some time. Smartphones are inherently portable and designed to act as truly personal devices, so users are accustomed to bring them wherever they go. As a result, there is no need to move the sensing device to the context we want to analyze since, most of the time, the device is already in the range of the event itself.

Having all these peculiar characteristics, smart devices constitute an amazing tool which can be leveraged to collect and analyze local data and, in general, to grasp a more clear understanding of the complex contexts we live in. This is one of the very reasons why a new fast-growing sensing paradigm has started gaining widespread adoption, leveraging the extensive presence of mobile personal devices in our lives and social participation of volunteer citizens. This new paradigm, called Mobile Crowd Sensing (MCS), has been widely adopted in distributed problem-solving applications, involving online or offline crowds.

To be concise, MCS is about relying on the crowd to perform sensing tasks through their sensors-enabled devices. Mobile Crowd Sensing (also referenced as Mobile Crowd Sensing and Computing [73]) was firstly introduced by Ganti et al. in 2011 and since then it has been the subject of a great deal of research [60, 72, 23, 95, 21]. Only in 2014, Guo et al. gave the first widely accepted formal definition:

> [...] a new sensing paradigm that empowers ordinary citizens to contribute data sensed or generated from their mobile devices and aggregates and fuses

[1] In a wireless sensor network a sensor node is commonly known as *mote*.

the data in the cloud for crowd intelligence extraction and human-centric service delivery. — Guo et al. [71]

1.1 Harnessing the Wisdom of Crowds

Over the past few years, the idea of crowd-powered problem solving has been a key research focus. In 2005, Surowiecki wrote a book titled "*The wisdom of crowds*", describing a general new phenomenon where "the aggregation of data or information from a group of people often results in better decisions than those made by a single individual from the group" [170] identified as *Crowd Wisdom*. Surowiecki stated that diversity of opinion, independence of thinking, decentralization, and judgment aggregation are four essential qualities which make a crowd *smart*.

Few years later Malone et al. tried to redefine the well known concept of *collective intelligence* [105] under the crowd-powered problem solving context [119].

Crowd Wisdom and Collective Intelligence share many aspects and, in particular, both focus on the advantage of group decision-making. In more recent years, those who tried to formalize the concept of MCS took inspiration from these two pillars extending the concept focusing more on crowd-powered data collection and processing.

The term *crowdsourcing* was introduced in 2006, in an article published in the *Wired magazine* by Howe [79]. A more formal definition was given by the same author in another study: "[...] crowdsourcing represents the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call. This can take the form of peer-production (when the job is performed collaboratively), but is also often undertaken by sole individuals. The crucial prerequisite is the use of the open call format and the large network of potential laborers." [78]. Here, key concepts are the initial open call and the presence of a large group of people (especially an online community) willing to participate by performing the requested service. The Wikipedia project is iconic: several thousands of contributors collaboratively create the most comprehensive encyclopedia of the world.

From its definition it is apparent how crowdsourcing follows a top-down approach, where a central institution releases an "open call" and somehow supervises the workers. On the other hand, as pointed out by Carreras et al., MCS initiatives usually follow a mixed bottom-up/top-down paradigm [23]. MCS applications are typically based on the direct involvement of users, trying to involve citizens in solving complex tasks or sensing complex contexts (through their smartphones) in order to solve issues such as public decision-making, urban planning, and quality assessment campaigns of public services. In some cases users can eventually submit reports about public issues, monitor

Table 1.1: Mobile Crowd Sensing literature summary partially taken from Guo et al. [72]

Concept	Definition and Relationship	By
Crowd Wisdom	The aggregation of data or information from a group of people often results in better decisions than those made by a single person from the group.	[170]
Collective Intelligence	Groups of individuals doing things collectively that seem intelligent.	[119]
Crowdsourcing	The practice of obtaining needed services or content by soliciting contributions from a large group of people, and especially from an online community.	[79]
Participatory Sensing	It tasks average citizens and companioned mobile devices to form participatory sensor networks for local knowledge gathering and sharing.	[18]
Mobile Crowd Sensing (and Computing)	An extension to the participatory sensing concept to have user participation in the whole computing lifecycle: (1) leveraging both offline mobile sensing and online social media data; (2) addressing the fusion of human and machine intelligence in both the sensing and computing process.	[72]

the air pollution, or even help in earthquake early detection, then the *central authority* can act to get a particular result based on collected data.

In 2006, Burke et al. proposed the closest concept to MCS: *participatory sensing* [18]. It tasks citizens to form participatory sensor networks through their personal mobile devices, harnessing their proximity for local data gathering and sharing.

When it was proposed, the definition of participatory sensing emphasized explicit user participation. In more recent years, with the widespread diffusion of smartphone sensing and mobile Internet techniques, the aim of crowd problem-solving systems has been broadened. We can see the definition of MCS — already seen in this Chapter — as an extension of the definition of participatory sensing. They differ in two main aspects: while participatory sensing only leverages data sensed from mobile devices by a physical community, MCS also exploits user-contributed data from online social network services (mainly open data from other projects); MCS usually harness both human and machine intelligence in both the sensing and computing process.

A very brief summary of key definitions and a reference to their first introduction is the shown in Table 1.1.

1.2 MCS applications

To date, a great deal of MCS applications has been developed and used in real-world scenarios. Presenting a comprehensive list of all the applications known so far is far beyond the scope of this work, but briefly describing some illustrative examples could contribute to gain a better understanding of the MCS peculiarities and challenges. For a more detailed analysis and a complete classification, please see the survey of Guo et al. [72] or the one from Zamora et al. [201].

As already seen, MCS applications can serve as sensing and processing instrument in many different fields. Due to mobile devices' inherent mobility, they can be utilized for sensing tasks aimed to gain better awareness and understanding of urban dynamics. Acquiring knowledge in such context is of prime importance in order to foster sustainable urban development and to improve citizens' life quality in terms of comfort, safety, convenience, security, and awareness. Many researchers have focused on studying urban mobility and behavioral patterns in urban areas, using MCS tools support to get their research question answered. For instance, Noulas et al. analyzed check-in histories of a large set of *location-based social network* (LBSN) users and found out that, for human movement prediction purposes, rank distance played a bigger role than physical distance [135]. Many other studies have investigated urban social structures and events starting from crowdsensed data. Crooks et al. studied the potential of Twitter as a distributed sensor system. They explored the spatial and temporal characteristics of Twitter feed activity responding to a strong earthquake, finding a way to identify impact areas where population has suffered major issues [40]. Large-scale data can be also collected by means of MCS platforms to analyze the actual social function of urban regions, a kind of data which is usually very difficult to obtain and that can be of primary importance concerning urban planning. For instance, Pan et al. started from the GPS log of taxi cabs to classify the social functions of urban land [139], while Karamshuk et al. tried to identify optimal placements for new retail stores [90].

Awareness of user location is the foundation of many modern and popular mobile applications, such as location search services, indoor positioning [199], location based-advertising [61], and so forth. But more useful and precise services can be enabled harnessing all the peculiar characteristics of personal mobile phones. As an example, Zheng et al. used crowdsourced user-location histories to build a map of points of interest which can be of help for people who are familiarizing with a new city [205]. Again, GeoLife [207, 206] is a MCS platform able to suggest new friendship looking at similarities in user-location logs, while CrowdSense@Place [29] is a framework able to exploit advanced sensing features of smartphones to opportunistically capture images and audio clips to better categorize places the user visits.

In many cases, the development of a particular MCS platform has been the answer to issues raised by pre-existing communities or grassroots initiatives. Citizens and policymakers have usually strong interests in matters like environmental monitoring, public safety, and healthcare, where the participatory and mobile essence of the MCS approach provides a novel way for collaboratively monitor the issue being considered. Besides, the moving nature of these topics draws the attention of online and offline communities. The potential of a community can be harnessed by MCS approaches to engage people and to make them participate in the data collection. It is not just a matter of the number of participants, rather someone who is moved by a topic not only will be more disposed to contribute but — as we are going to discuss in Section 1.3.5 — he or she will also be prone to provide better and more complete data [86]. As an example, Ruge et al. described how their application SoundOfTheCity [158] allowed users to link their feelings and experiences with the measured noise level, helping in providing information essential to have a more clear understanding of the context (is the high noise level in a party, at a festival or just in a crowded street?). This is an illustrative case of how qualitative data provided by users can enrich the quantitative data gathered through personal smart devices. In short, to fully harness their potentialities when analyzing such contexts, MCS applications should aim not only to collect as much data as possible but also to provide ways for users to enrich the collection with *thick data*[2] [183]. Other examples of MCS applications analyzing topics of common interest are: NoiseTube [118] which was a system able to exploit volunteers' smartphones to collect data about environmental noise in users' daily lives and to aggregate them to obtain a fine-grained noise map; U-Air [208] inferred air quality data by heterogeneous crowdsensed data comparing them against information from sensing stations and traffic information; the Great Backyard Bird Count project[3] (cited by Cuff et al. [41]) used volunteers to continuously report the count of watched birds in the US.

Mobile Crowd Sensing application can also be used for assisting in disaster relief [112, 111], such as earthquakes and floodings [36, 11], or in critical events like gas shortages in urban areas [132]. Healthcare is another field where MCS is helping a lot by collecting a wealth of data for applications more and more useful for an aging society like ours. Google researchers did pioneering work in 2006 using health-related search queries to estimate illnesses distribution in US [52], while Wesolowski et al. exploited the widespread diffusion of mobile phones to analyze malaria spreading in Kenya [189].

Also, many mobile social recommendation services, like friend, place, or itinerary recommendations, has been enabled by the body of data collected by MCS platforms.

[2]Thick Data can be defined as: "data related to qualitative aspects of human experience and behavior, particularly when used as context for the analysis of a large data set"

[3]Great Backyard Bird Count: http://www.birdsource.org/gbbc/ [last accessed 06 Jun. 2017]

Most of these systems shared the characteristic of combining mobile crowdsensed data and user-generated information from LBSN. In 2014 Yu et al. presented a system whose aim was to produce travel plan suggestions starting from data such as Points of Interest (POIs) features, temporal and spatial constraints and user data taken from social networks [200]. Yang et al. developed SEALs [197] a fine-grained preference-aware location search framework which leverages the crowdsourced traces from Foursquare, using check-ins and extracting users' sentiment about locations. More recently, Marakkalage et al. introduced a crowdsensing platform aimed to identify POIs among elderly population in Singapore [120]. Their system can passively collect the location history through GPS sensors embedded in users' smartphone, and determine popular places among the elderly.

Crowdsensed data in urban areas can be leveraged for public transportation system design, traffic forecasting, and real-time information systems, for monitoring the road network condition and so on. Location history is usually perceived as *sensitive* data by users who thus need to be motivated or reassured to take part in this kind of crowdsensing process employing some cooperation incentives. Most researchers in this field choose to collect data from user mobile phones, mainly using GPS, accelerometers, compass, gyroscope, and other related sensors already available on board of every average smartphone, while some others made use of other kinds of smart object like GPS tracking devices (which are usually already on board of vehicles). Many applications on traffic dynamics are based on leveraging data from public transportation, buses, and taxis [187, 20, 108]. Among many it is worth mentioning the work of Giannotti et al. where, using location histories of numerous GPS tracking devices embedded in vehicles (usually for safety reasons), researchers have used mobility patterns to analyze if the current official district division reflects actual traffic flows, travel customs and center of attraction for drivers [62]. An application only based on data sensed using personal user smartphone is SmartRoadSense [2]. The platform is a crowdsensing system used to monitor the surface status of the road network. The SmartRoadSense mobile app is able to detect and classify the road surface irregularities by means of accelerometers and send them to an in-cloud server. Aggregated data about road roughness are shown on an interactive online map and made available as open data [57].

1.2.1 MCS classification

Due to large heterogeneity of applications, many classifications have been proposed. MCS platforms can be classified based on the type of phenomena being monitored thus dividing them up in applications with a *personal* and a *community* sensing approach [60]. Applications collecting data about an individual — such as healthcare applications, fitness monitoring, and so on — belong to the first category whereas applications whose aim is to acquire data about large-scale phenomena are part of the second category. A community sensing application pertains to the monitoring of events or contexts not feasible by a single individual, because of the scale of the phenomenon is too broad (e.g., air pollution in an urban area) or because to get sufficiently precise data they have to be collected from many different sources with different tools (e.g., road conditions monitored by a single user, even if performed multiple times, may be significantly less accurate than when monitored by multiple users, using different vehicles).

Following another classification principle, MCS applications can also be divided into two other categories depending on how much users are aware of the sensing process. When users are asked to actively produce data by reporting an event, taking photos, or taking notes about a particular condition of a context, we can talk of *participatory* sensing approaches [18]. *Opportunistic* approaches, on the other hand, are those where the user involvement is minimal and the sensing process is more autonomous and transparent. To this category belong applications that continuously track users or systems where the volunteers can participate in the sensing campaign just by activating the process in their mobile client [102].

Many other classifications are possible. Applications can be divided into different categories based on their scale (e.g., urban, regional, national, worldwide), by categorizing their topic, by their infrastructure, or by the motivation incentives they employ to engage users.

1.2.2 MCS architecture

Regardless the category they fit into, all MCS applications share the same simple architecture in terms of main software components. As coordinated distributed software platforms, MCS systems are usually composed of a central in-cloud *application server* and several *mobile clients*. The central server is responsible for managing all the centralized phases of the whole sensing and processing procedure. It implicitly or explicitly assigns the sensing tasks to the users then it receives data collected by participants. The sensing task is performed in a decentralized manner through mobile clients of volunteers. The software client can collect data directly or with the help of the user. In some cases the main mobile client can draw data by connecting to secondary IoT clients, which would be otherwise unable to be directly connected to the application server. In Figure 1.1 the architecture of MCS is depicted in a synthetic but complete way. For a more comprehensive analysis of typical MCS architectures, see the work of Louta et al. [110].

A more detailed description of each possible phase of the crowdsensing process is shown in Figure 1.2. Many researchers tried to formulate a suitable reference process structure to illustrate the key functional blocks and to explain the key techniques of MCS systems [72, 193, 202, 110]. The following analysis is mainly based on their work. Few orig-

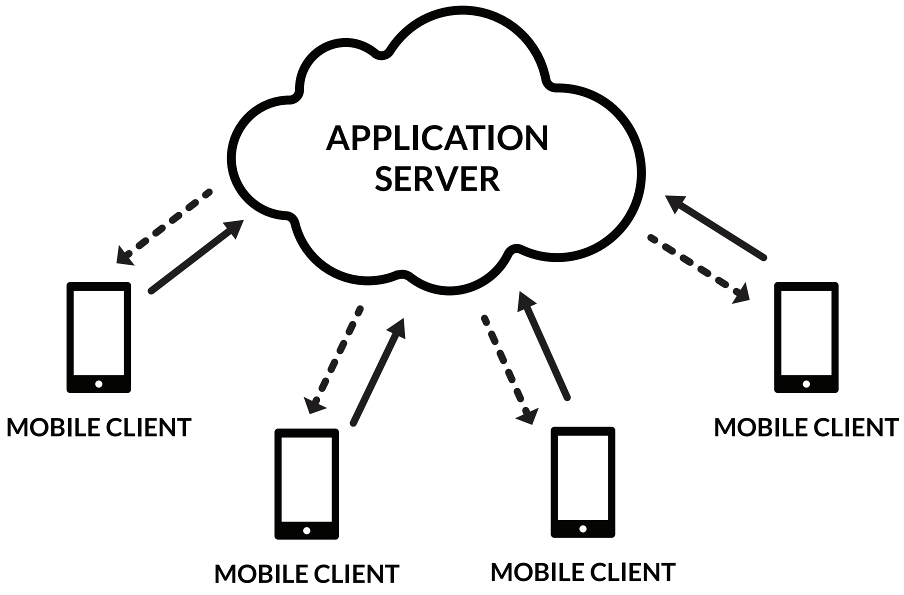


Figure 1.1: Software architecture of MCS applications.

inal contributions have been included in order to make the described process updated and coherent with some recent works discussed so far.

Task Creation In this phase, the *central entity* or the MCS organizer creates specific *tasks* and provides a detailed description of the required actions. The task creation can be even started by users, the same ones who will consume the data collected using the MCS application [146]. Depending on the platform used, the description could be either in natural language or domain specific language that software clients are able to understand and present to volunteers [151]. In some cases, the task creation is implicit in the platform structure. Volunteers who join the application are automatically tasked with a defined sensing operation [2].

Task Allocation This phase can occur in a centralized or in a decentralized way. The central entity can analyze the sensing task and assign it to specific participants (or nodes of the sensing network) possibly trying to respect given constraints: ensuring area coverage, minimizing the task completion estimated time, maintaining the number of volunteers involved under a given threshold, ensuring a minimal average trust value among the selected participants, and so on. Another approach is to notify all clients that a new task is available and let them choose whether to

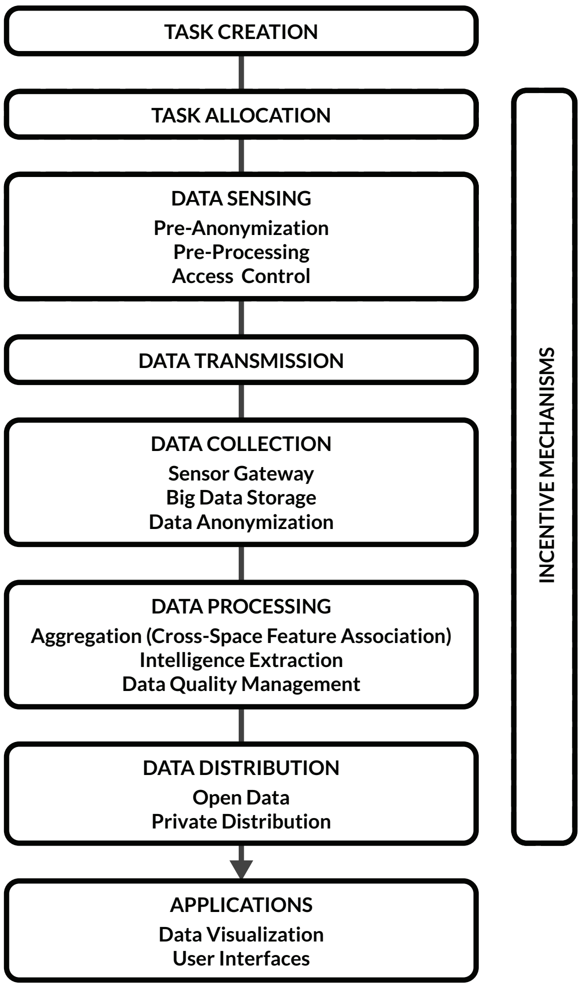


Figure 1.2: Proposed reference process structure for MCS applications.

take part in the sensing task or not [22, 153]. Depending on the motivation incentive systems utilized, some systems also allow approaches like auction based assignments [188, 203, 181].

Data Sensing Involves both information sensed from mobile devices and user-contributed data from mobile Internet applications [72]. MCS applications usually have to tackle security and privacy issues, thus providing users with automated or semi-automated mechanisms determining what kind of information they want to publish and whom to share them with is fundamental. Many MCS systems resort to *access control* mechanisms and *pre-anonymization* techniques. In order to reduce transmission costs and size, data are often *pre-processed* on board of the user device. Finding the appropriate tradeoff between the amount of processing to be done onboard of smartphones and in cloud — after the data have been transmitted — is a crucial parameter for a MCS application. To resolve this issue, Xiao et al. proposed an innovative approach using proxy Virtual Machines and a distributed cloud infrastructure as communication middleware between mobile clients and the central application server [193]. The data sensing is an inherently distributed phase.

Data Transmission Sensed data have to be transferred to the central server for further processing. In doing so, mobile clients need to take care of inevitable network interruptions and communication issues. MCS applications may adopt opportunistic transmission paradigm, storing data in mobile clients internal memory until a suitable network connection makes possible the upload to the server [37]. Other applications implement mechanisms to forward data from a client to its peers in order to exploit their connection while some MCS platforms provide nodes of the sensing network with cooperation tools to allow heterogeneous nodes to enhance the performance of data transmission [69].

Data Collection In this phase, data are received by mobile clients and stored into appropriate memory supports. Privacy-preserving techniques are applied to ensure security and to avoid that malicious users acquire collected data and can track them back to users. The *sensor gateway* module provides a standard approach — usually implementing common web services technologies [156] — to enable data collection from crowdsensed sources supplying a unified interface. MCS applications may collect a vast amount of heterogeneous data and *big data* storage systems are usually employed. Big data techniques simplify the collection of large-scale and complex data like noise level measured across a urban area. Sensing tools used by participants to evaluate the phenomenon at stance typically varies a lot, leading to significant differences in the accuracy of crowdsensed data. Therefore data are commonly transformed and unified before being stored and passed to the next phase in order to boost further processing. *Data anonymization* techniques like data coarsening, randomization, k-anonymization, spatial cloaking are applied to data to eliminate or to mask information that might compromise participant privacy by leading to their identification or by disclosing sensitive attributes [146].

Data Processing Aims to derive high-level intelligence from raw data received. Using logic-based inference rules and machine-learning techniques this step focuses on discovering frequent data patterns in order to extract crowd-intelligence starting from data sensed by mobile users and user-contributed data from other mobile Internet applications mixed together. The first step of the data processing is the *data aggregation* phase, in that, raw data from different users, time and space are combined on different dimensions and associated with reference known features (e.g., map-matching [45]). Then further data processing techniques are applied to extract the three kinds of crowd-intelligence (namely: user awareness, ambient awareness, and social awareness [70]). Most MCS applications also sport a *data quality management* module. When information passes through this phase, different statistic methods are applied to classify the quality of aggregated data.

Data Distribution Once data have been aggregated, and the crowd-intelligence has been extracted from them, this information is usually made publicly available to be re-used (often as *Open Data*) or only shared in a private way with authorities, communities, companies, etc.

Applications Finally data arrive at the stage where they are re-used, exploited or just shown. Many applications and services can be enabled by MCS platforms. The implementation of a usable *user interface* and of *data visualization* techniques (such as mapping, graphing, animation) are essential to fully exploit the crowd-intelligence extracted by the underlying MCS system starting from raw data. Both of them make data-based decision-making tasks and awareness sharing to users feasible.

Cooperation Incentives As shown in Figure 1.2, the cooperation incentives phase influences almost every other stage of the proposed framework. Users can be motivated to participate in a sensing task by using incentives in the task allocation phase. In the sensing phase, the idea of a possible future reward can motivate users to collect better or more data. Data transmission and collection encompass dealing with privacy-related issues. Strict privacy rules can seriously limit the kind of incentives the platform can make use of. The definition of a data quality index in the data processing stage can positively or negatively affect the type of rewarding token a user collects by providing some data. About the data distribution phase, a user could be rewarded with different quantity of rewards based on how much third-party applications or users are using the aggregated data he contributed to.

1.3 Challenges and open issues

As a relatively new paradigm, in the MCS field, many challenges are currently being addressed by researchers and developers. MCS application designers often borrow techniques and tools from related areas such as *big data* and *gamification* but many other problems still need an optimal and original solution. In this Section, many open issues are presented.

1.3.1 Data quality

Data quality is usually described as the degree of how suitable the information is for the objective we want to employ them. One of the drawbacks in the MCS process already seen is that there is no o little control over distributed sensing operations. Acquiring data on a particular phenomenon through MCS often means to collect a vast quantity of information coming from many different unsupervised sources. Each source produces data by means of automated or semi-automated processes whose measurements could be context-related, non-calibrated or just not precise enough. Particularly, the involvement of the human variable enhances the general lack of control over the way data are acquired.

Redundancy is the main strategy MCS platforms usually put in place against low-quality data collected by unknowing or malicious users. But it could not be enough, in particular for sensing tasks which require few participants or where redundancy is unsuitable (e.g., the phenomenon to be sensed is too transitory to be sensed many times or by many users). Therefore, volunteers and data selection are often needed to improve overall data quality. Filtering algorithms could be directly implemented in MCS clients aiming to filter out data before their transmission to the server. While server-side data filtering is commonly preferred because it can act as a barrier for data provided by malicious users who could not be effectively stopped with distributed approaches.

Data quality could also be a determining factor concerning how much a volunteer should be rewarded for the work he carried within a sensing task. Once data are processed they should trigger positive or negative feedback on the motivation incentive system implemented by the application, assuring greater rewards for users who provided higher quality data.

1.3.2 Scalability

Thanks to its distributed sensing approach, a MCS application could gather vast quantities of data in little time. Data coming from various sources also usually present heterogeneous structure and types. Volume, velocity, and variety are typical features of *big data* [103], indeed many MCS applications make use of big data analytics and big data storage techniques to manage such big, fast and complex body of information. Traditional approaches in storing and processing data revealed to be not suitable for handling demanding tasks such real-time sensing in wide areas. In order to achieve a comprehensive picture of the sensing object, MCS systems often try to integrate crowdsensed data and user-contributed knowledge bases. Once data have been received, pre-processed and stored in-cloud, machine learning and data visualization techniques help in mining and presenting the crowd-intelligence mined from raw data [22].

1.3.3 User interface

As discussed in Section 1.2.2, MCS platforms are commonly composed of a server-side stack and a client-side application. The latter usually consist in a mobile application running on at least two of the three major mobile platforms which — at the time of writing — are leading the market: Android, iOS and Windows Mobile (successor of Windows Phone). Most of the times, MCS applications aim to engage as many participants as possible so, lowering the barriers preventing users from joining the sensing task is a primary concern of platform designers. In terms of MCS mobile clients, this usually means to provide users with a native mobile application for their platform of choice. In order to speed-up the development and to contain implementation costs a multi-platform development approach (like Xamarin [75]) or a web-based approach are used [93].

Recently, the growth of mobile application market started to slow down, and mobile users became less prone to install new apps unless for utterly necessary tasks [35], preventing some MCS applications to reach many users.

Moreover, in 2013 Xiao et al. proposed 3 code design principles to "Lower the barriers to large-scale mobile crowdsensing":

> *i)* Separation of data collection and sharing from application-specific logic.
>
> *ii)* Removal of app installation on smartphones from the critical path of application deployment.
>
> *iii)* Decentralization of processing, and data aggregation near the source of data.

The principle II states the need for MCS application designers to consider alternative ways to cover the *last mile* between the platform and the volunteers. Nonetheless, a viable alternative to standard mobile applications is yet to come.

1.3.4 Privacy

MCS applications are strongly based on human participation. Many projects start as grassroots initiatives and, as seen in Section 1.2.2, the sensing phase is usually accomplished thanks to volunteers who are willing to help in the sensing campaign.

But human involvement inherently brings privacy concerns. Participants may not want to share their sensory data, as they may contain or reveal private or sensitive information about the user. Privacy concerns may discourage users from joining sensing a campaign. Disclosing of one's identity, exposing sensitive attributes or activities about participants are typical user's concerns, and they can lead to a participant withdrawal from a sensing task or to not making him joining the task in the first place. Therefore MCS applications have to commit to protecting users privacy.

The privacy issue in MCS has recently been studied extensively [145, 146]. Researchers defined detailed adversary models to understand how *malicious* and *semi-honest* entities within a MCS sensing paradigm could obtain private information about other users [145]. They have also studied most common attacks and found out what countermeasures have to be taken in order to prevent or stop such attacks. Most common techniques include data anonymization through use of *pseudonyms*, *connection anonymization*. Others act directly on the MCS data structure, trying to remove or conceal, spatio-temporal data or links between contiguous data, like: *spatio-temporal cloaking*, *spatial obfuscation*, and *data coarsening*. Some approaches try to acquire as little information as possible about users, applying *pull-based data collection* scheme [145] and *zero knowledge techniques* [51].

Many privacy-preserving techniques are based on the fundamental principle of making the user in control of how many and which information he wants to share. For instance, users could be willing to voluntarily disclose some personal privacy-concerning data, like sharing their current location in a social network. Projects like NervousNet [147] provide tools which let MCS participants to explicitly set sharing or not-sharing preferences about their information. This kind of approach tries to realize a privacy-preserving *by-design* approach.

Once data have been aggregated, and crowd-intelligence has been mined MCS applications commonly make results of their sensing process available privately or as Open Data. In both cases, the platform design needs to ensure that no one who has helped in any stage of the process can be back-tracked starting from shared data. Open data protection and group privacy are so fundamental to even raise ethical issues among researchers [54].

Is worth mentioning that applying strict privacy-preserving mechanisms in designing a MCS system, could hinder the implementation of many other useful features or even the sensing task aim itself.

1.3.5 Cooperation incentives

People involved in MCS help the sensing task process by providing their limited resources. To sense, collect and process crowdsensed data participants have to make either implicit efforts (e.g., energy, bandwidth, monetary costs) or explicit efforts (e.g., to collect or to give inputs, to assess process or other efforts, to act as bridge for other peers) [72]. As many MCS instances are grassroots initiatives, volunteers may be willing to help just because interested in the topic, but most of the time users need to be strongly motivated by providing incentives mechanisms [114]. Cooperation incentives can be broadly classified as *intrinsic* or *extrinsic* (financial) incentives [99]. Intrinsic incentives comprehend: *interest* in the topic (users are willing to participate because they think the topic is important or interesting), *social* or *ethical* motivation (such as altruism, or interest in public recognition), and personal *enjoyment* (in that, user enjoy the sensing process because it is an entertaining activity or a proper game). Intrinsic incentives are often insufficient to motivate volunteer and many MCS platform resort to extrinsic incentives which probably are the easiest way to motivate people regardless the type of activity they are asked to do. Extrinsic incentives include fiat money, virtual cash, or credits for online/offline services. The drawback of using financial incentives is that once money is involved users will be motivated to deceive the system to get more economic benefits. In this regard, financial incentive mechanisms based on game theory have been proposed [196, 84]. Unfortunately, these methods are quite complicated to implement and unsuitable for many types of MCS systems. There is still the need for a new rewarding platform able to encourage and attract people and which comply to the many different MCS application peculiarities.

1.4 Structure of this work

In the rest of this work, we address some of the MCS open issues presented so far. In Chapter 2 we tackle the data quality issue, proposing two original solutions about characterizing the quality of crowdsensed data [58, 45]. In Chapter 3 we discuss cooperation incentives, trying to address the coexistence between strict privacy-preserving mechanisms and functional rewarding platforms [33]. Chapter 4 contains an in-depth analysis of a new conversational interface paradigm, which we proposed to name *bot-plication* and we believe could be a suitable answer to some user interface hurdles of MCS applications [97]. In the last Chapter, we draw our conclusions on the MCS paradigm and outline possible future research directions.

Chapter 2

Data Quality

During the data collection phase, MCS data can be conceptually modeled as a stream or many single tokens of information flowing from clients towards the central server. On the server side, information will be received and stored discretely as numerous *data particles*, possibly linked one another. Along with application-specific values (e.g., noise level, location history, temperature, etc.), each particle typically includes a timestamp (or a time window), and geolocalization values (usually a point, a path, an area, or a set of these features). For instance, an urban noise level monitoring application would gather data particles containing: the recorded noise level (typically a numeric value), the timestamp of when the sensing has been performed, and the geolocalization of the place where the recording happened (usually expressed as a couple of latitude and longitude values). MCS applications generally process all these information together to have a complete understanding of the phenomenon being analyzed. Therefore an acceptable data quality of all the data contained in a particle is of prime importance. In general, applications analyzing transient events need precisely time-stamped information while for applications like SmartRoadSense [2], which gather data about an event happening in a moving context, having a good estimate of the sensing instrument's location is paramount.

In this chapter, data quality issues are discussed. In the first Section, we try to define an index based on simple statistical tools, suitable for MCS data and applicable on different parts of a data particle. In the second Section, we propose a map-matching algorithm capable of dealing with inaccurate GPS information and dense traces of data. This algorithm has been designed to be used by applications in which having a reliable association between data and features on a map is fundamental for the platform's purposes.

2.1 Bootstrap for Data Quality Estimation

As already described, the distributed architecture of MCS systems, the heterogeneity of sensing devices used, and the direct involvement of users influence the quality of data [72]. To fully exploit the wealth of data collected by MCS platforms, the number representing sensed quantity should be associated with quality indicators in order to help in discriminating between reliable and unreliable data. Having an estimate of such quality has a large impact on following processes carried on by other system components (e.g., data filtering, crowd-intelligence extraction, and rewarding mechanisms). MCS features that concur to define data quality can be broadly summarized as follows:

- The spatial and temporal resolution of the monitoring activity;
- Systematic and random errors in the measurement of the sensed quantity. Such errors are due to lack of precision and heterogeneous processing and communication capabilities in embedded sensors;
- Users with different trustworthiness levels may provide reliable or unreliable data.

To date, approaches to tackle the data quality issue have primarily focused on outlier identification and elimination [72], on collaborative resolutions of lack of consistency in data [127], or on reputation systems used to single out corrupted or misleading contributions [82, 83].

Crowdsensed data are obtained by a measurement activity. In MCS, it can be a direct or an indirect measurement and it is usually performed in a distributed way by using a smartphone as a sensing device. The typical way to deal with the data quality issue is by estimating the uncertainty associated with each measurement (as is customary in physical sciences [172]). Instead, we propose an approach for estimate the quality of data collected by MCS systems which takes into account the distributed nature of the process. We cast the problem of data quality assignment as an evaluation of the uncertainty of the underlying measurement process. Briefly, the measurand (i.e., the quantity associated with a phenomenon we are measuring) estimate has to be associated with a measure of uncertainty. The uncertainty is typically estimated by looking at how the dispersion of the measured values is (the larger the dispersion, the bigger the uncertainty). The measurement error is estimated by the dispersion around the actual value of the measurand which is determined by the interaction among error-prone sources. In MCS systems, the value is typically the result of a composition of different measurements. Therefore, the uncertainties associated with each measurement need to be propagated computing their combination [172]. Lately, user trust principles an uncertainty propagation techniques have been used as sensing uncertainty metrics in the sensor network context by Asmare and McCann [5].

Typical methods to gauge uncertainty propagation commonly imply some modeling assumptions about the measurement process, especially for those approaches concerning the probability density function of measured values [172, 38, 39]. The distribution described by probability functions could be employed either by analytically deriving the distribution of the measured quantity (given) its functional dependence from the input quantities or by performing Monte Carlo simulations to obtain an estimate of the measurand's output distribution [38, 39, 12].

Monte Carlo numerical simulation is a suitable solution if the input distribution is known or reasonable assumptions can be made about it. Unfortunately, this is not the case of MCS applications where measurements are performed by many devices in unsupervised conditions, making the formulation of a sound statistical model nearly impossible. Besides, having an adequate mathematical model of the input variable would not suffice in any case, as their composition function makes it difficult to derive an analytical expression of the output (e.g., aggregation phases in MCS usually output quantities that are results of recursive composition functions [2]). Additional issues are encountered when the temporal dimension is taken into account. In fact, the composition of several measurements over longer periods is a common approach used to smooth and progressively update the collected information. This poses the challenge of how to manage such an update of measurements at given checkpoints.

To solve the issues mentioned above, we propose to employ statistical non-parametric bootstrap which is a largely known data-driven technique for empirically estimating the uncertainties of measured unknown quantities. Bootstrap has been proven to be a suitable solution when modeling approaches and analytical methods are hardly applicable [155, 100]. In simple terms, bootstrap is a Monte Carlo simulation method that approximates the sampling distribution by sampling the original observations with replacement. The statistic of interest is computed for each resample and the resulting distribution — named *bootstrap distribution* — can be used as a replacement of the actual sampling distribution in order to study its shape and spread [155, 100, 48, 49, 91]. Statistical bootstrap as a tool to characterize uncertainty propagation has been used by Kass et al. in the framework of neural data analysis [91] while Lee et al. exploited it as a method to refine estimation of a system for blood pressure measurement [104].

In this Section we try to give the following contributions:

- A method that frames the problem of assessing data quality in crowdsensing platforms into a formal, technically sound approach, by casting it as an uncertainty evaluation problem.

- The design — based on statistical non-parametric bootstrap — of a data-driven strategy which could be used to get rid of complex interactions among the (potentially many) variables affecting the measure estimate and the related uncertainty propagation.

- The evaluation of the applicability on a MCS application, characterized by periodic updates of data and associated quality.

We validated the proposed approach against numerical simulation and synthetic benchmarks. We also implemented the bootstrap-based method in a real-world mobile crowdsensing platform for road surface roughness monitoring. Our experimental results suggest that our method is a suitable tool for evaluating data quality in a complex sensing tasks by estimating the uncertainty of the overall process measurements.

In the following Section, we describe the architecture of the MCS platform used in our case-of-study. Then, the proposed method is detailed and, in the last two Sections we outline the experimental setup, we present empirical results obtained, and finally, we draw our conclusions.

2.1.1 Reference System Architecture

In the following Sections we will often refer to a MCS platform called SmartRoadSense (already cited in Chapter 1). The system has been developed to provide quantitative estimations of road network surface roughness [2, 57, 45] and in this Section we will briefly describe its features and architecture. The SmartRoadSense's approach at data sensing and processing is general enough to be employed also in different contexts. As many other MCS platform, sensing tasks in SmartRoadSense are performed by multiple distributed sensing devices by means of which volunteers contribute to gauge the quantity of interest in a specific location, within a specific time-window.

As shown in Figure 2.1, the architecture of the SmartRoadSense platform is characterized by the following three layers:

- An app running on users' smartphones during a given car trip. The application makes use of the smartphone's accelerometers and computation capabilities to collect and process acceleration values the device is subject to. The result, representing the estimated roughness of the travelled road in a given point at a given time, is geo-referenced, time-stamped, and transmitted to a server by means of radio connectivity.

- A cloud-based back-end service in charge of collecting, aggregating and storing data from multiple users. According to Figure 2.1, this layer is in charge of two tasks:

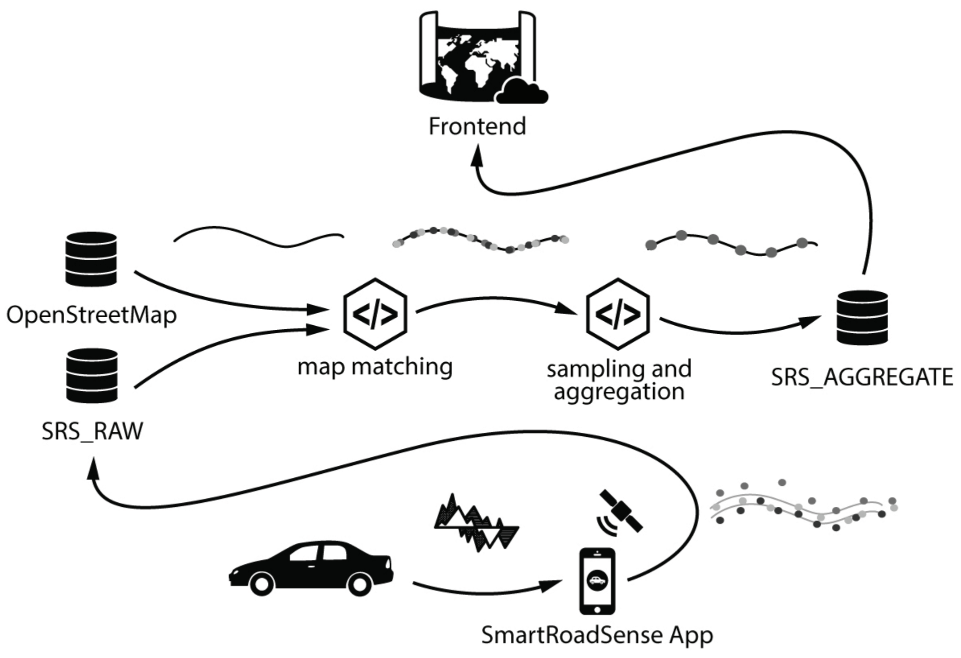


Figure 2.1: SmartRoadSense architecture

  - *Map-Matching*: georeferenced roughness indexes stored in the database of raw-data (*SRS_RAW*) are projected on digital cartography maps, specifically OpenStreetMap[1]. Map-matching entails the association of GPS points to features on a digital cartography maps;
  - *Sampling and aggregation*: data is subsequently aggregated to provide a single evaluation (for a given spatial coordinate) of the roughness index, given the data made available for that point by multiple users. Aggregated data is used to populate the related database (called *SRS_AGGREGATE*).

- A front-end service providing visualization capabilities of the geo-referenced information produced by the SmartRoadSense processing pipeline. The same front-end also allows interested end-users to download a continuously updated version of the database containing all SmartRoadSense aggregated data in a ready-to-be-reused fashion. Each row of the open-data dataset contains a set of information relative to the roughness level, the geo-localization, the quality of the data, and

[1]OpenStreetMap is a collaborative project whose aim it to build a free editable map of the world. Please refer to https://www.openstreetmap.org for more information.

even a indication of the estimated number of occupants of each vehicle that has been involved in the gathering process. The detailed description of each information is shown in Table 2.1.

Column Name	Format	Description
LATITUDE	DECIMAL DEGREES	The latitude coordinate of center of the section of the road where the PPE value has been estimated.
LONGITUDE	DECIMAL DEGREES	The longitude coordinate of center of the section of the road where the PPE value has been estimated.
PPE	DECIMAL	The average roughness level of the road section.
OSM_ID	LONG INT	The ID of the road in the OpenStreetMap dataset.
HIGHWAY	TEXT	The road category according to the OpenStreetMap classification[2]
QUALITY	DECIMAL	A numerical estimate of the quality of this particular PPE value. This quality index has been calculated using our bootstrap-based method, in our case-of-study.
PASSENGERS	DECIMAL	The average of the number of passengers in vehicles involved in the process.
UPDATED_AT	DATE (ISO 8601)	The last update of the data for that particular road section.

Table 2.1: SmartRoadSense open data structure

In SmartRoadSense (and possibly in other mobile crowdsensing systems), the sensing process can be divided into time epochs, during which data is continuously gathered, processed and aggregated. Segmentation of both space and time dimensions (e.g., through the definition of a bi-dimensional grid and the discretization of the time axis) can be in fact considered a common approach to the design of MCS at different spatiotemporal resolutions [82, 83, 204].

At the end of a given time epoch the system updates current information on the status of measured variables and, in case, it performs a composition operation with data collected in previous epochs (in SmartRoadSense an epoch represents a week of monitoring activity). The platform continuously receives values of road roughness from end users. Roads are spatially segmented into landmark points, then all values associated to positions falling within a given range (typically 20 meters) of a landmark point p are aggregated and concur to the overall roughness index of p (the mean value of contributed points is taken by default). At the end of each week current epoch terminates, and the roughness value of each point p is updated by taking the average between the value of current epoch and the value of previous epoch. This processing inherently implements a form of infinite impulse response filter, the aim of which is to progressively downweigh (through an exponential decay of weights) the contribution of older samples to the value assigned to p. Needless to say, different update rules can be conceived, according to different specific needs.

The above description exemplifies the difficulties that could arise when dealing with uncertainty propagation in these settings, since the measurand (the roughness index of p in SmartRoadSense) needs to be tracked along its evolution and the corresponding unknown uncertainty subject to possibly complicated transformations.

2.1.2 Bootstrap Based Uncertainty Propagation

To bypass all the issues related to the propagation of uncertainty in MCS platforms that preclude the adoption of analytical and Monte Carlo methods, we propose to take advantage of the statistical bootstrap.

Figure 2.2 provides an overview of the toolflow of the proposed approach when applied to the SmartRoadSense crowdsensing system, while Table 2.2 summarizes the symbols used along this Section.

As illustrated by Figure 2.2, data produced by terminal devices at a given time epoch t_i (with $i = 0, 2..., n_{nw}$) are collected into a sample of size n_i. Nonparametric bootstrap is applied to this sample (for each time epoch). Data is sampled with replacement, obtaining N_b resamples, each of size n_i. The statistic of interest (i.e. the mean value $\overline{x}$) is computed from every resample and plugged into the processing block labeled as $\mathscr{Y}$, which represents the functional relationship between all variables that influence the measurand. In SmartRoadSense application, this phase encompasses the propagation of uncertainty along the different time epochs, according to the update filter: each mean value $\overline{x}_i$ at time epoch t_i is averaged with the corresponding value y_i computed at time epoch t_{i-1}. The result $y_i = \frac{\overline{x}_i + y_{i-1}}{2}$ is then stored as current value to be composed with a new measurement at next time epoch.

The distribution of values assigned to y is the output bootstrap distribution. Such distribution can be studied to obtain information about its center, shape and spread.

Symbol	Description
p	Generic geo-referenced landmark (aggregated) point
nw	Number of epochs
$t_0, t_1, ...t_i, ...t_{nw}$	Time epochs
$n_0, n_1, \cdots, n_{nw}$	Size of samples and of resamples at epoch t_i
N_b	Number of bootstrap resampling iterations
$\overline{x}$	Mean value of a generic resample
$\overline{x}_i$	Mean value of resample at epoch t_i
$\overline{y}_i$	Measurand variable computed at epoch t_i
$\mathscr{Y}$	Generic function relating x_i to y_i

Table 2.2: Description of notations and symbols used in this Section.

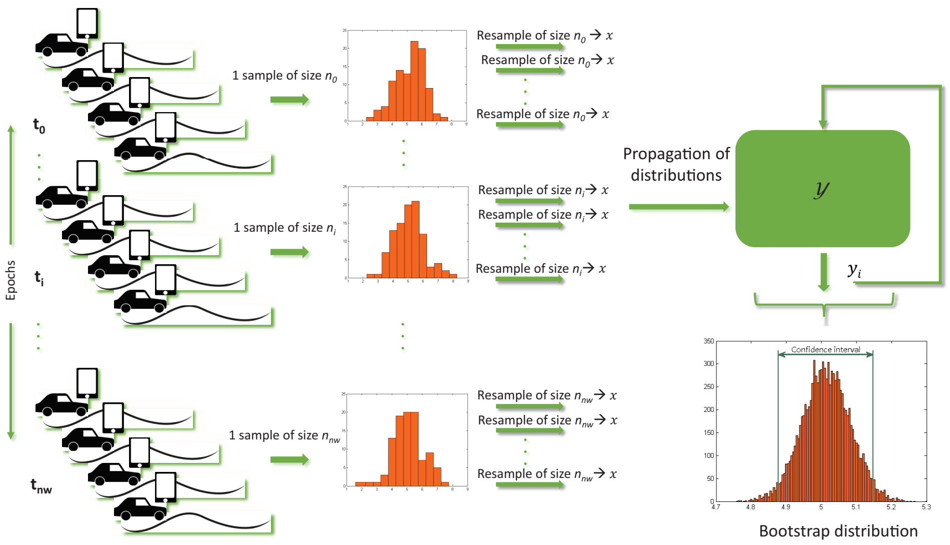

1 sample of size n_0
Resample of size $n_0 \rightarrow x$
Resample of size $n_0 \rightarrow x$
Resample of size $n_0 \rightarrow x$
t_0
t_i
t_{nw}
Epochs
1 sample of size n_i
Resample of size $n_i \rightarrow x$
Resample of size $n_i \rightarrow x$
Resample of size $n_i \rightarrow x$
1 sample of size n_{nw}
Resample of size $n_{nw} \rightarrow x$
Resample of size $n_{nw} \rightarrow x$
Resample of size $n_{nw} \rightarrow x$
Propagation of distributions
$\mathscr{Y}$
y_i
Confidence interval
Bootstrap distribution

Figure 2.2: Toolflow of the bootstrap-based approach.

While the center of the output bootstrap distribution represents the estimate of the statistic under study (the mean value in this specific case), the shape provides effective information about the type of distribution and, finally, the spread conveys information about the output uncertainty (what we are searching for). Needless to say, bootstrap resampling does not lead to any improvement in the accuracy of the estimate of the statistic, since this clearly depends on the accuracy of the initial sample from which all resamples are derived. Nevertheless, important information regarding the sampling distribution is encoded in the bootstrap distribution and this is exactly what we are interested to exploit to infer the uncertainty of measurement. The histogram placed in right-bottom corner of Figure 2.2 illustrates this concept. Indeed, bootstrap distribution values are finally sorted and samples representative of a given percentile are extracted to provide a confidence interval CI to be used as uncertainty estimate.

Other statistics might be evaluated according to the presented method. In fact, one of the major points of strength of the bootstrap is its flexibility in handling different types of statistics, which has to be compared with the difficulties faced to derive analytical results. If we were interested, for instance, at investigating the uncertainty associated to the median value (instead of the mean) the same approach would remain valid and we would only need to change the computation of the mean value $\overline{x}$ from each resample with the respective median value.

The main algorithmic steps of the proposed approach can be summarized as follows:

```
1. for $k = 1$ to $N_b$
    • for $i = 2$ to $nw$
        – Sample with replacement the observation vector collected during time epoch $t_i$
        – Compute mean value $\overline{x}_i$ (or any statistics of interest) of the bootstrap resample at time $t_i$
        – Update measurand $y_i$ according to $\mathscr{Y}$. In SmartRoadSense, $y_i = (\overline{x}_i + y_{i-1})/2$
2. Extract 95% confidence interval from bootstrap distribution
```

For each crowd-based measurement the system will reiterate the whole process described by the above pseudo-code (and shown in Figure 2.2). For instance, the SmartRoadSense platform applies the bootstrap process for each aggregated point in the example. As already detailed in Section 1.3.2, MCS applications can collect large amount of data in little time. The big data flow could raise the question of the scalability of the proposed approach, which should be taken into consideration when a huge number of uncertainty evaluations have to be carried on. Nonetheless,the inherent parallelism of the proposed approach should be remarked. In fact, uncertainty intervals associated to different geo-localized points can be computed independently from each other. Therefore, in principle, they can be split in many processing tasks that can be autonomously executed in parallel, potentially mitigating the impact of the computational burden.

2.1.3 Experimental Results

To validate the introduced technique, several experiments have been conducted:

- First, a set of synthetic benchmarks has been devised to compare the bootstrap based uncertainty evaluation against a standard Monte Carlo method, under the assumption of knowing the input probability distributions, needed to run the Monte Carlo experiments.
- Second, a sensitivity analysis has been performed to evaluate the dependence of the results from the number of bootstrap resamplings (N_b), allowing to explore the tradeoff between accuracy and computational complexity.
- Third, an experiment has been conducted to simulate the case of a system measuring a time-varying quantity.
- Last, the uncertainty of a measurement within the SmartRoadSense crowdsensing platform has been computed to show the applicability to a real world use-case.

2.1.4 Synthetic benchmarks

The rationale of these experiments was to assess the suitability of the proposed approach in terms of accuracy of the confidence interval. The proposed bootstrap-based uncertainty propagation has been validated by comparing it with standard Monte Carlo uncertainty propagation (SMC, for short), assuming the knowledge of input probability density functions. We recall that, while this is an assumption that has to be made if one wants to apply standard Monte Carlo propagation, it cannot be taken for granted. The bootstrap propagation technique, conversely, doesn't rely on any type of knowledge of input data, rather it performs a data-driven Monte Carlo simulation by drawing the so called *pseudo-observations* from the vector of initial observations and generating from it (through resampling) all the information needed for inference tasks.

Three types of distributions have been considered, covering a wide spectrum of possible statistical configurations, namely: a Gaussian distribution of mean $\mu = 5$ and standard deviation $\sigma = 1$, a uniform distribution taking values in the $[4, 6]$ interval, and a Rayleigh distribution with scale parameter $b = 5$.

We included the Gaussian distribution because of its role in statistics and error distributions [155]. As well, we chose the uniform distribution since it is often studied in uncertainty evaluations of measurements [38]. Finally, we also took into consideration the Rayleigh distribution because it is an example of asymmetric distribution, which adds to the significance and coverage of our experiments.

For what concerns the bootstrap based uncertainty propagation, we generated 100 points (according to each of the input distributions) representing the observations. Sampling with replacement has been performed with $N_b = 10^5$ replications, mean values have been computed and given as input to the propagation pipeline representing the update process described in the last Section: the mean of observed values at each epoch has been averaged with the mean of observed values at previous epoch. Three sets of experiments were performed, simulating a time horizon of, respectively 2, 10 and 25 epochs (on a system like SmartRoadSense, characterized by weekly updates, this means simulation on an interval spanning from half a month to around half a year). The approximate 95% confidence interval has been computed by taking the 0.025 and 0.975 quantiles of the resulting bootstrap distribution.

Regarding Monte Carlo simulations, for each type of distribution we generated 100 points, took the mean value and propagated it according to the same rule (i.e. mean of current epoch averaged with the updated value y computed at previous epoch) on the same time horizons (i.e. 2, 10, and 25 epochs). The whole process has been repeated for 10^5 trials, leading to a distribution of values from which the average value and an estimated 95% confidence interval have been computed.

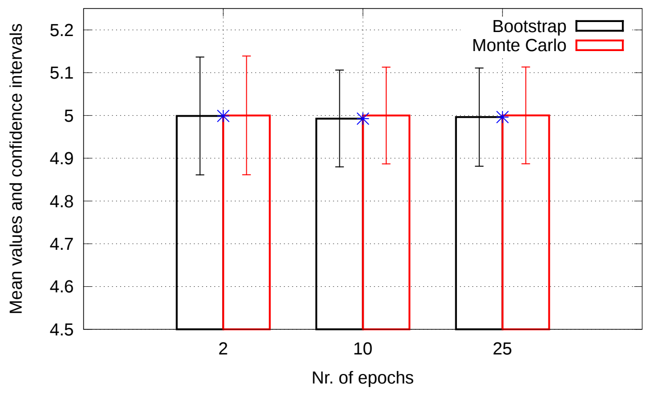
All the experiments have been repeated for 10 runs. Results are represented by the average of the following values: *i)* the mean value at the end of the propagation process (representing the estimate of the measurand); *ii)* the lower bound of the 95% confidence interval; *iii)* the upper bound of the 95% confidence interval.

In Figures 2.3a, 2.3b, and 2.3c we reported histograms providing a comparison of the performance of both methods according to the above mentioned metrics for, respectively, Gaussian, uniform and Rayleigh input distributions. For each Figure, histograms denote the mean value estimate, together with error bars encoding the confidence intervals for each simulated epochs horizon (2, 10, 25). As a reference term, we also computed the values (represented as star markers in Figures 2.3a, 2.3b, and 2.3c) that would be obtained for the measurand if no bootstrap were applied, but only a simple composition of the observations were made epoch by epoch.

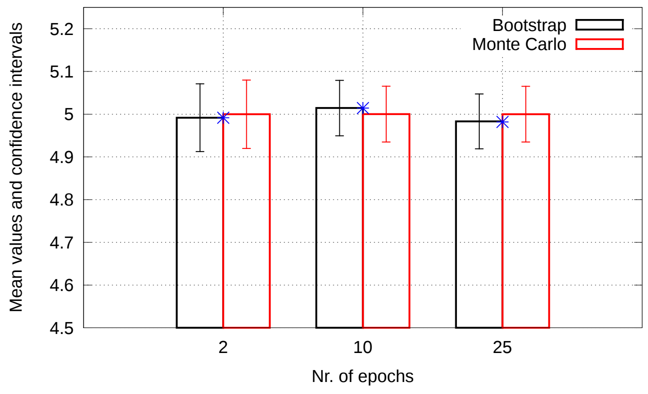
Results provide evidence of a very good agreement between the standard Monte Carlo approach and the proposed bootstrap uncertainty propagation method.

In particular, the width of confidence intervals obtained with our method are within a 1.5% deviation from the intervals estimated by means of SMC, with a maximum 0.15% relative error on the value of the lower bound and a 0.13% relative error on the upper bound for the Gaussian input.

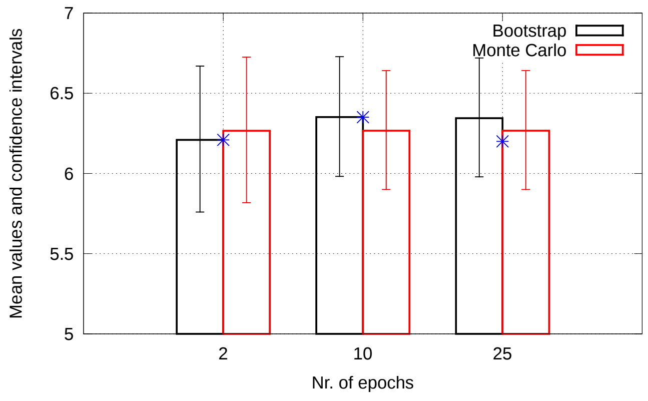
In case of uniform input distribution, we obtained confidence intervals whose width differs at most for a 1.7% from that of SMC, while lower bounds of the intervals are within 0.32% from their SMC counterparts, and upper bounds fall within a 0.35% range.



(a) Mean values and confidence intervals. Normal distribution ($\mu = 5, \sigma = 1$).



(b) Mean values and confidence intervals. Uniform distribution ($a = 4, b = 6$).



(c) Mean values and confidence intervals. Rayleigh distribution ($b = 5$).

Figure 2.3: Synthetic benchmarks

Finally, the analysis of experimental results with input following a Rayleigh distribution showed a 0.81% maximum difference between the confidence interval widths of the proposed approach and those of SMC. The maximum relative error obtained by the proposed approach with respect to the SMC method, amounts to 1.38%, for lower limits and 1.31% for upper limits.

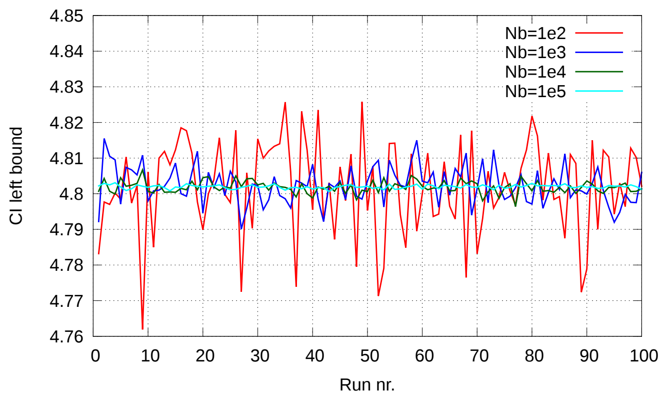
As expected, our technique doesn't lead to any improvement in the accuracy of estimates, which clearly depends on the accuracy of the initial observation vector (an inherent feature of resampling techniques). This justifies the differences seen with uniform distribution and, in particular, with the Rayleigh distribution.

Once the accuracy of the bootstrap based uncertainty propagation has been assessed and demonstrated to be consistent with the one obtained by Monte Carlo approaches (that assume prior knowledge of statistical distribution of input), we turned our attention to other types of experiments. We indeed analyzed the effect of the number of resampling iterations (N_b) on the system performance, by computing 95% confidence intervals with the proposed algorithm for different values of N_b (namely $N_b = 10^2, 10^3, 10^4, 10^5$) along a time horizon of 10 epochs. Input observations were randomly drawn from a normal distribution ($\mu = 5, \sigma = 1$). The results obtained over 100 runs are plotted in Figures 2.4a, 2.4b, and 2.4c for, respectively, the left bound of the 95% confidence interval, the mean estimate, and the right bound of the 95% confidence interval.

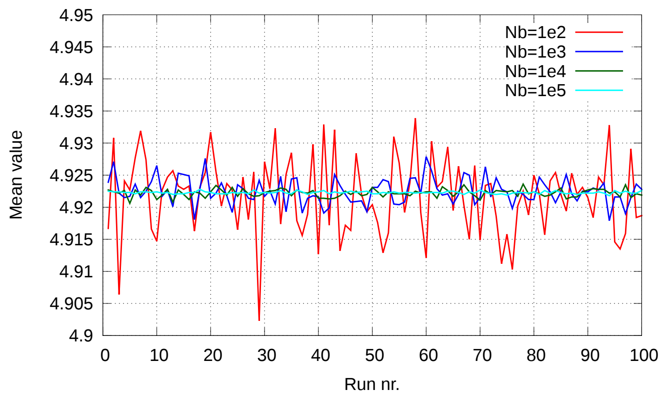
Experiments highlight the variation of the mean estimate and of the confidence intervals as the number of resamplings changes. In particular, albeit not markedly significant (maximum variations are within a 1.7% range), the effect of Monte Carlo random fluctuations across the different runs is clear: higher values of N_b correspond to lower variations across the runs, in accordance to known results in the bootstrap theory [76]. It took on average 36.5s to compute the confidence intervals for a single run when $N_b = 10^5$, and 0.0365s when $N_b = 10^2$. Experiments have been performed on an Intel® i7 CPU, with a 2.80GHz frequency clock and 8GB RAM, running a Matlab®implementation of the bootstrap-based approach.

This empirical experiments highlight a mild variation as N_b decreases: the width of confidence interval changes by up to 8% (when $N_b = 10^2$), with respect to the width computed with $N_b = 10^5$. This empirical evidence shows the potential of alleviating the computational workload by lowering the number of resampling iterations without severely affecting the accuracy. Conversely, when mitigation of stochastic fluctuations is an issue, N_b should be increased.

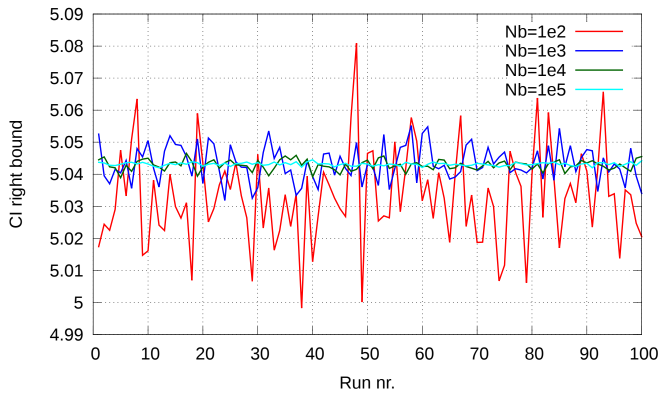
The final experiments on synthetic data have been designed to test the proposed approach on a wide time interval during which the value of the measurand is subject to dynamic change. This experimental set up has been conceived to model situations when a potential drift of the physical quantities has to be monitored and tracked by the crowd-



(a)



(b)



(c)

Figure 2.4: Sensitivity analysis: left bound of 95% confidence interval (a), estimated mean values (b), right bound of 95% confidence interval (c), for different number of bootstrap resamples across 100 runs. Normal distribution ($\mu = 5, \sigma = 1$).

sensing system. For instance, in SmartRoadSense the road surface could progressively deteriorate and, at a given point, could be subject to maintenance or in NoiseTube [118] the noise level of a specific area could suddenly change. The effect of this possible evolution has been evaluated by simulating a *piecewise* linear dynamics of the measurand along different time epochs. In particular, input data was generated according to three types of statistical distributions as follows:

- *Gaussian distribution*: observations were randomly generated with a mean value linearly increased at each epoch (from $\mu = 5$, at epoch 0 to $\mu = 7$, at epoch 50) and then set back to $\mu = 5$ for the remaining 50 epochs. Standard deviation was kept constant for the whole simulation ($\sigma = 1$).
- *Uniform distribution*: data was generated by taking values uniformly at random in an interval that was progressively shifted from $[4, 6]$, at epoch 0 to $[6, 8]$, at epoch 50. From epoch 51 to epoch 100 values were drawn again uniformly in the $[4, 6]$ interval.
- *Rayleigh distribution*: input values were taken from a Rayleigh distribution whose scale parameter b was linearly changed from $b = 5$ (at epoch 0) to $b = 7$ (at epoch 50) and then set again to $b = 5$ (from epoch 51 to epoch 100).

These observations have been then used as input for the uncertainty propagation processing pipeline based on the non-parametric bootstrap. Following the previously described experiments, the update at each epoch was performed by taking the average between the measurand estimate at current epoch and the one at the previous epoch.

Plots of the mean value and error bars representing the associated confidence intervals are reported in Figures 2.5, 2.6, and 2.7 to illustrate the results for, respectively, the normal, uniform, and Rayleigh distribution. As expected, the system can effectively cope with a changing input, by dynamically tracking its evolution. Thanks to the proposed approach, the estimates of the measured variables and the corresponding confidence intervals can be also effectively updated.

2.1.4.A Case study: SmartRoadSense

In order to exemplify the practical applicability of our proposal, we applied the bootstrap-based method to a dataset extracted from the SmartRoadSense project [13]. Data refer to a road segment in Italy composed of 10 monitored points, each one aggregating from 12 to 30 measurements across two adjacent weeks (week 18 and 19, corresponding to the period from May 2, 2016 to May 15, 2016) of the SmartRoadSense monitoring activity. The main features of the dataset are reported in Table 2.3: with respect to the aggregated point indicated in column 1, we reported in column 2 the week (epoch) the values refer

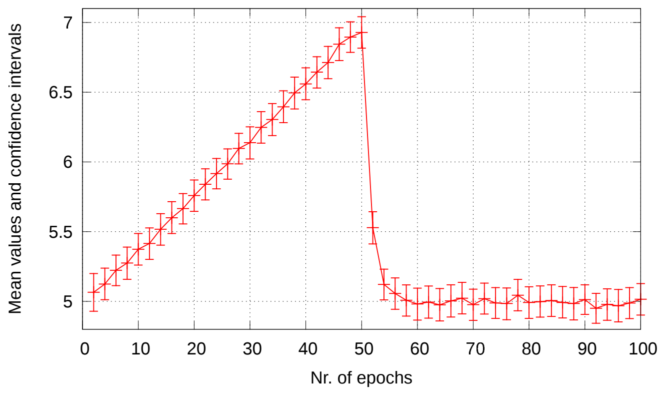


Figure 2.5: Dynamic input analysis: mean values and confidence intervals as function of number of simulated updates. Normal distribution. Nr. of bootstrap resamples = 10^3.

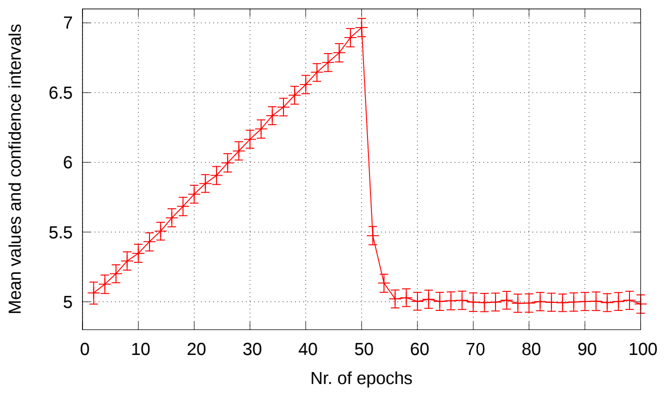


Figure 2.6: Dynamic input analysis: mean values and confidence intervals as function of number of simulated updates. Uniform distribution. Nr. of bootstrap resamples = 10^3.

to, in column 3 the number of points aggregated, and in columns 4 and 5 their mean value and standard deviation.

For each point of the dataset we applied the bootstrap-based uncertainty propagation method to compute the 95% confidence interval at the end of the period spanning the two weeks of observations. The number of bootstrap resampling iterations was set to $N_b = 10^4$.

Results are reported in Figure 2.8, plotted as a histogram with error bars, overlying the piece of map that includes the road under investigation. Each bar is associated to an aggregated point whose roughness index is expressed through a color map (green for low roughness values, red for high roughness levels).

To provide some further details about the sensitivity of the method with respect to the number of resampling iterations, we computed 95% confidence intervals for point

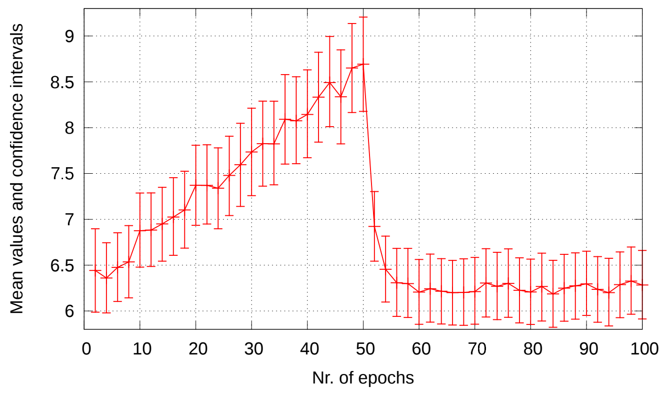


Figure 2.7: Dynamic input analysis: mean values and confidence intervals as function of number of simulated updates. Rayleigh distribution. Nr. of bootstrap resamples = 10^3.

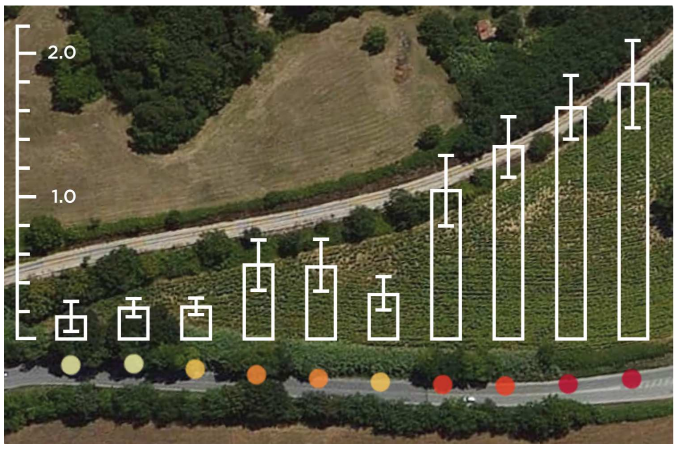


Figure 2.8: Case study: mean values and confidence intervals at [2.5% 97.5%] for 10 points taken from a monitored road in SmartRoadSense. Nr. of bootstrap resamples = 10^4.

P_{10} of the SmartRoadSense dataset with different values of N_b (i.e. $Nb = 10^2, 10^3, 10^4, 10^5$) across a set of 100 runs. Interestingly, the analyzed point clearly represents an example of a small sample size being composed of 10 measurements (in each of the two weeks). Results of this experiment are illustrated in Figures 2.9a, 2.9b, and 2.9c for, respectively, the left bound of the intervals, the mean estimate, and the right bound. Stochastic Monte Carlo variations are, as for the synthetic benchmarks, significantly compressed in a small range when $N_b \geq 10^4$. It is worth noticing a higher variability of the results from run to run for low values of N_b (up to 17.4% for the left bound, $N_b = 10^2$), with respect to the experiments performed on synthetic benchmarks experiments, plausibly because of the effect of the small sample size.

On average, the confidence intervals for a single run were computed in 6.8s, when

Table 2.3: SmartRoadSense dataset: number of raw points, mean value and standard deviation for each aggregated value and monitored week.

Aggregated point	Week	Count	Avg	Stdev
P_1	18	12	0.1587	0.0217
	19	13	0.1318	0.0314
P_2	18	30	0.2278	0.1764
	19	23	0.2017	0.1702
P_3	18	30	0.2023	0.1093
	19	23	0.2421	0.1680
P_4	18	23	0.5260	0.5422
	19	30	0.5055	0.5580
P_5	18	18	0.4093	0.4577
	19	28	0.5954	0.6407
P_6	18	18	0.2312	0.2084
	19	25	0.3863	0.4178
P_7	18	23	1.1319	1.0053
	19	32	0.9361	0.7257
P_8	18	23	1.6216	0.8599
	19	24	1.0610	0.6417
P_9	18	13	1.9805	0.8552
	19	9	1.2424	0.5309
P_{10}	18	10	1.9473	0.8892
	19	10	1.6059	0.5432

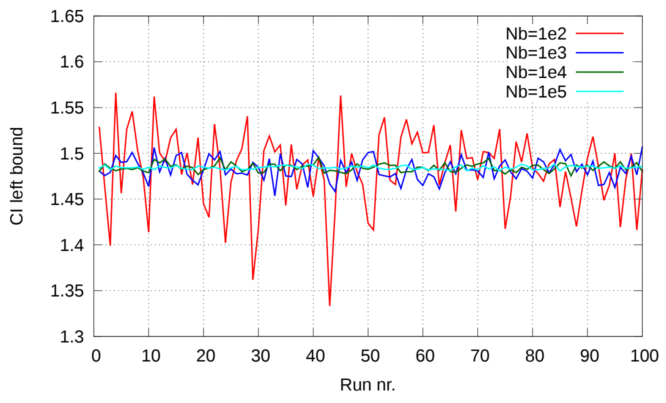
$N_b = 10^5$. The same task, when $N_b = 10^2$, was completed approximately three orders of magnitude faster, i.e. around 7ms (timing results refer to the same hardware configuration and implementation used for synthetic benchmarks).

2.2 Accuracy Improvement in MCS Vehicular Applications

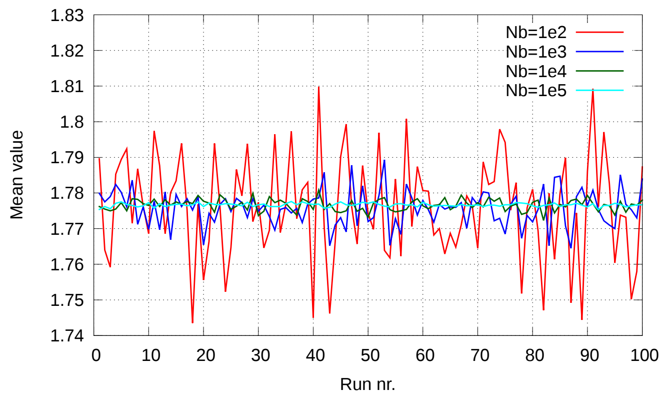
When dealing with geo-spatial data, accuracy is a two-fold issue, in that it concerns both the value of the sensed quantity and the GPS coordinates of the point in which it was measured. When the point needs to be associated with an object on a map, GPS accuracy impacts map matching [131, 175].

This Section focuses on efficient map matching algorithms for crowdsensing road applications. In particular, we provide a classification of map matching issues and propose incremental real-time algorithms to tackle them and improve the overall mapping accuracy.

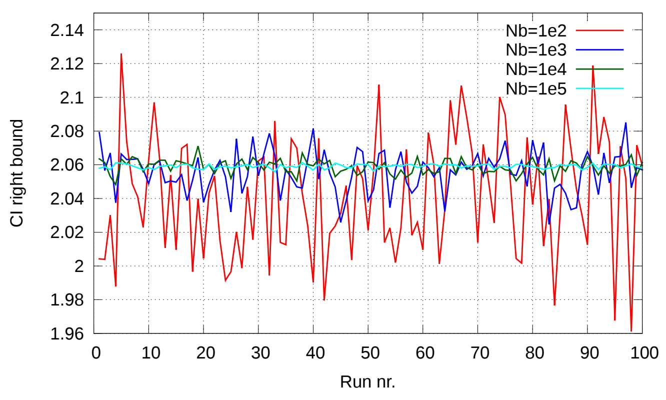
As we already did in the last Section, we are going to take SmartRoadSense as a case



(a)



(b)



(c)

Figure 2.9: Sensitivity analysis: left bound of 95% confidence interval (a), estimated mean values (b), right bound of 95% confidence interval (c), for different number of bootstrap resamples across 100 runs. SmartRoadSense dataset, point P_{10}.

study [2]. Among the wide range of vehicular applications that pose map matching issues, SmartRoadSense has several peculiarities that give rise to new challenges: i) it requires fine-grained sampling, ii) it targets not only recognized main roads, but also not-yet-tagged road segments, and iii) it collects data from heterogeneous devices installed both on public and on private vehicles. Hence, map matching is addressed without relying on a pre-established trajectory (as in the case of public transportation), nor on the knowledge of roads and links among them. Rather, incoming data are sequences of points that need to be matched on road segments based only on geometrical considerations and on first-order reachability analysis performed on the fly. Moreover, we assume map matching to be applied in real time, so that computational complexity has to be kept under tight control, looking for a solution belonging to the category of incremental online methods.

Experimental results, validated against ground truth datasets collected for one month on known bus lines, show that map matching accuracy can be significantly improved by means of incremental linear algorithms applied on the fly without leveraging any topological information.

The rest of the Section is organized as follows: Section 2.2.1 provides a summary of related work on vehicular crowdsensing systems and map-matching algorithms; Section 2.2.2 introduces concepts and definitions related to the map-matching problem; Section 2.2.3 describes the proposed approach; Section 2.2.4 describes the experimental setup and presents the results.

2.2.1 Related Works

In this Section, we report some recent scientific literature related to techniques, methods, and systems prosed so far in the scope of our work, namely: system architectures for mobile sensing (with a particular focus on vehicular applications) and algorithmic approaches for map matching problems.

2.2.1.A Vehicular crowdsensing systems

One of the first applications of vehicle-based crowdsensing is represented by *CarTel*, a system developed with the aim of detecting road potholes utilizing GPS and accelerometers mounted in cars equipped with embedded microprocessors [50]. Mobile sensing for the detection of traffic conditions, bumps, and acoustic events has been investigated through the integration of data from accelerometers and microphones into a system called *Nericell* [129].

Thiagarajan *et al.* proposed in 2009 a system (named *VTrack*) targeting the goal of road traffic delays estimation by means of mobile phones [175]. A particular focus of this work is represented by the reduction of smartphone energy consumption during sensing activities and by the compensation of noise associated with sensors sampling. *CTrack* is a system developed by some of the authors of VTrack to achieve accurate trajectory mapping from GSM fingerprints instead of using WiFi signals or GPS traces [176].

Large-scale mobile sensing has been proposed for air pollution monitoring in *OpenSense* [1]. Data gathering in OpenSense is achieved by means of participative sensing of citizens equipped with enhanced modified smartphones or ad hoc pocket sensors, and by means of special sensor stations placed on public transport vehicles.

Recently, some architectures for road surface collaborative monitoring have been proposed [2, 57, 101]. The more representative is SmartRoadSense, the platform already presented which is designed to integrate mobile sensing and cloud systems to support continuous monitoring of road surface quality. A roughness index is computed on board of end-users' smartphones and then transmitted to be stored, processed, aggregated and visualized in cloud. The SmartRoadSense platform entails several components, resulting in a multi-tier architecture. In particular: *i)* signal processing techniques have been used to process acceleration traces and obtain a numerical value (the roughness index, RI) that correlates to the quality of the road segment spanned by a given vehicle; *ii)* GSM communication is used to transmit RI values, GPS coordinates and timestamp in batch to a remote server; *iii)* a cloud-based back-end is used to store, process and aggregate geo-referenced traces from multiple users; *iv)* a cloud front-end based on CartoDB [3] is adopted for visualization of geospatial data (e.g., overlay on geographical maps such as Google maps [4]).

2.2.1.B Map matching

Map matching is an inference process that reports a sequence of location data onto a map. In mobile sensing applications, data is typically a GPS trace (i.e. a sequence of time-stamped *(latitude, longitude)* values and the map refers to annotated road networks in a digital georeferenced database. Sequence localization data can derive from different sources of information (e.g., GPS devices, GSM fingerprints, WiFi access points position) while target maps could differ in the information content and available details (e.g., topological information, one-way roads annotations, etc.).

Map-matching algorithms are classified into *global* algorithms and *incremental* algorithms. Global approaches [131, 175] process whole input traces in order to achieve a solution, while incremental algorithms [178, 121] work on small segments to be processed in sequential order. On one hand, global algorithms usually result into more accurate solutions but must be inherently run only offline; on the other hand, incremental algorithms

[3] Please see http://cartodb.com
[4] Please see http://maps.google.com

make local choices that could possibly impact the accuracy of mapping but are suitable for online execution. Hence, global strategies are particularly useful to obtain reliable data from inaccurate and/or sparse input, while online algorithms become a mandatory choice for real-time applications and in all cases when efficiency is an issue.

In this framework, many different techniques have been proposed to tackle the map-matching problem. Some authors investigated the use of computational geometry techniques posing the problem as a pattern matching between curves under Frechet distance [16, 28]. Others proposed the use of signal processing methods (e.g., Extended Kalman Filters) [136] or Bayesian estimators [121]. According to recent scientific literature [64, 184], the best performances in terms of accuracy are obtained by algorithms based on *Hidden Markov Models* (HMMs). HMMs allow to test multiple possible mappings and find the maximum likelihood solution by means of dynamic programming (i.e. Viterbi algorithm). This statistical approach grants robustness to the matching algorithm, resulting in enhanced accuracy when faced with noisy inputs [131, 176]. A local, incremental variant amenable for an online implementation has been recently presented by Goh *et al.* [64].

Despite significant advancements obtained, the above mentioned algorithmic approaches present some issues with respect to the mobile crowdsensing scenario. In fact, global solutions incur high computational overhead that makes them unsuitable for vehicle-based applications where timeliness is often required. Furthermore, most state-of-the-art methods make some assumptions on input data (e.g., the availability of topological information) which is not always guaranteed. Overcoming these issues is one of the main purposes of this work.

2.2.2 Problem Statement and Scope

This Section formulates the map-matching problem addressed in this paper referring to known definitions [64] and to terms adopted in OpenStreetMap [138].

Trace A trace, $T = (t_n | n = 1, ..., N)$, is a sequence of N samples collected by a vehicle. Each *trace point* t_n is characterized at least by: time stamp (t_n.t), GPS coordinates (t_n.lat, t_n.lon), and sample (t_n.val). Additional fields can be available, like the GPS accuracy (t_n.acc) or the vehicle speed (t_n.v).

Line A line, $L = (p_m | m = 1, ..., M)$, is a M-point polyline representing a road segment as a series of segments connecting vertices p_1, ..., p_M in order. Each *vertex* p_m is represented by its coordinates (p_m.lat, p_m.lon).

Map A *map* S is a set of lines representing all the road segments of interest.

According to OpenStreetMap, we consider a *road* as a relation among lines, possibly annotated with viability information (speed limit, directions, permissions, etc.). A road network is a set of roads complemented by a connection matrix which adds topological information to the map. A road network is defined on a map and, in general, it covers just a subset of the lines of the map. The percentage of lines covered by the road network in a given region depends on the maintenance and updating of the underlying data base. Working on a map (rather than on a road network) maximizes the coverage at the cost of giving up topological information which is usually exploited in map-matching algorithms.

Map-Matching Given a trace T and a map S, the goal of map matching is to find the correspondence between each point of T and a line of S.

In general, reducing the sampling rate (i.e., reducing the number and density of the points in the trace) makes it harder to determine the actual trajectory of the vehicle on the map. Hence, considerable research efforts have been devoted to the development of robust map matching solutions able to cope with sparse traces [131, 175]. In the limiting case in which there is less than 1 sample per line, the key problem is to figure out which path the vehicle took between two points. The topology of the road network is essential in this case.

On the other hand, challenging issues can be raised also by the abundance of points provided by high sampling rates. When the rate reaches the order of one sample per second, there are two new problems to face. The first one is accuracy, in that the distance between subsequent points in the trace might fall below the resolution of the GPS, causing many possible artifacts. The second one is performance, in that the sampling rate poses tight constraints on the time taken to process each sample in online real-time applications.

2.2.2.A Map Matching Issues

Figure 2.10 provides a schematic representation of the basic artifacts possibly produced by map matching algorithms fed by a dense trace. Lines (i.e., road segments) are denoted by labels a, b and c, while dots represent trace points. Bold lines on the left represent the false trajectory inferred by matching points on the closest lines, while those on the right represent the actual (ground-truth) trajectory. All the traces are assumed to go from left to right (meaning that points to the left have time stamps preceding those to the right) along a ground-truth trajectory that goes from a to c. Label b is used to denote a line

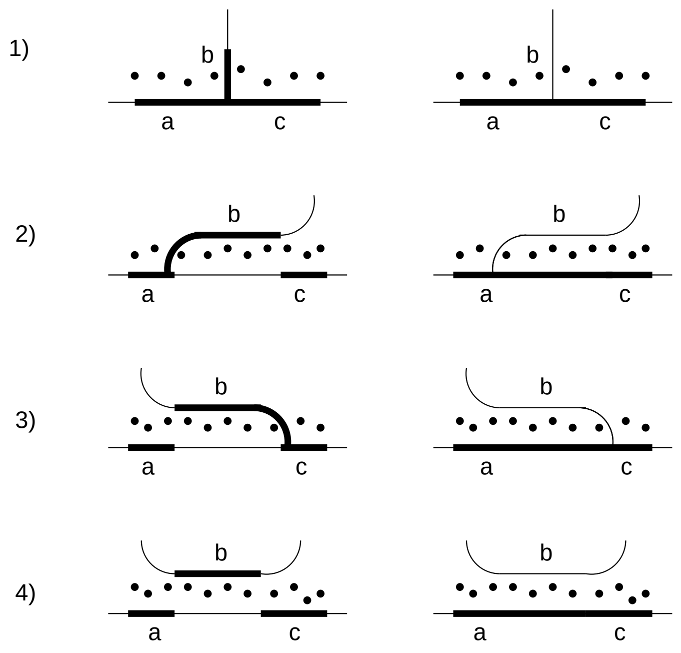


Figure 2.10: Schematic representation of the 4 main artifacts of map matching applied to dense traces. For each case the artifact is represented to the left and contrasted with the ground truth trajectory, to the right.

which is not in the trajectory but passes so close to some of the trace points to induce mapping artifacts.

In *Case 1* line *b* is a crossing road orthogonal to *a* and *c*. The wrong trajectory looks as if the vehicle had taken the cross and then was immediately turned back on the main road.

In *Case 2* line *b* is one of the two roads of a fork which runs parallel to *c* for a non-negligible stretch. If, after the fork, most of the trace points are closer to *b* than to *c*, it looks as the fork was taken. But when *b* and *c* diverge, the remaining points are mapped back to *c* without a viable link.

In *Case 3* a fork similar to that of Case 2 is encountered in the opposite direction, so that the fake trajectory jumps from *a* to *b* as the two roads get close to each other, and then it rejoin the main road at the fork.

In *Case 4* some of the samples are mapped on line *b* even if it never encounters lines *a* and *b*.

All the artifacts possibly encountered when map-matching a dense trace can be expressed as a combination of the four listed above. Without loss of generality, in cases 1, 2, and 3 we assume the road to switch from line *a* to line *b* exactly when it crosses *b*. All other situations can be easily obtained by considering *a*=*c*.

For the sake of explanation, the points in Figure 2.10 are represented as equidistant from each other and laying on an ideal trajectory, even if misplaced with respect to the map. Real traces look much worse, in that the points usually jump on both sides of the ground-truth lines and sometimes they overlap and locally reverse the order. However, persistent errors like the ones schematically represented in Figure 2.10 are the most difficult to detect and correct.

It is also worth mentioning that, in the absence of viability or topological annotations, the artifacts of Figure 2.10 occur whenever there are lines that cross or get close to each other on the map, even if there are no paths between them in the road network.

The purpose of our approach is to find an incremental map-matching algorithm able to detect and correct matching artifacts when dealing with dense traces in the absence of topological information.

2.2.3 Proposed approach

The output of map matching is the association of each trace point to a line. We use t_n.line to denote the property of point t_n that represents its association to a given line. Hence, map matching reduces to setting all the t_n.line properties to the appropriate line id.

Restricting the range of candidate solutions to the ones that can be executed incrementally in linear time, we start by matching each point to the closest line, as determined by issuing a query to the underlying geospatial data base. To speed up the refinement of this first-cut matching we make use of *run length encoding* (RLE) to represent contiguous subsets of trace points mapped on the same line [159].

Run A *run* $R = (r_k | k = 1, ..., K)$, is a sequence of K contiguous points taken from a given trace, such that all the points in R are mapped on the same line (denoted by R.line) and the points that precede and follow R in the trace (if any) are mapped on a different line.

Runs are computed on the fly based only on proximity matching, and then incrementally processed to correct artifacts. A sliding window of three runs, denoted by *previous* (R_p), *currrent* (R_c) and *next* (R_n), is used to this purpose. At each step a decision is taken on the correctness of the matching of R_c, based on the consolidated matching of R_p and on the first-cut matching of R_n.

The decision process is based on the following conditions, which minimize the number of additional queries:

- A) Reachability from previous run. R_c.line and R_p.line cross in a point close to the first point of R_c and to the last of R_p (only two points and two lines involved in the query);
- B) Reachability of next run. R_n.line and R_c.line cross in a point close to the first point of R_n and to the last of R_c (only two points and two lines invlved in the query);
- C) Fork. R_p.line, R_c.line and R_n.line cross in the same point, close to the first and last points of R_c (only two points and three lines involved in the query) ;
- D) Short run. The number of points in R_c is lower than a given significance length L (no geospatial queries required).

	A	B	C	D	Action
Case 1	T	T	T	T	split R_c between R_p and R_n
Case 2	T	F			merge R_c in R_p
Case 3	F	T			merge R_c in R_n
Case 4	F	F			split R_c between R_p and R_n

Table 2.4: Truth table of the artifact compensation algorithm.

Table 2.4 shows the decision criteria used to detect the 4 cases of Figure 2.10 and the actions taken to correct each of them. Rows are associated with possible artifacts, while columns are associated with conditions. Empty entries represent don't care conditions.

In Cases 1 and 4 the current run (R_c) is split between the previous and the last ones. This is done by assigning R_p.line to the line property of the first points of R_c, and R_n.line to the remaining points of R_c. In Cases 2 or 3 the current run is merged either in the previous one or in the next one.

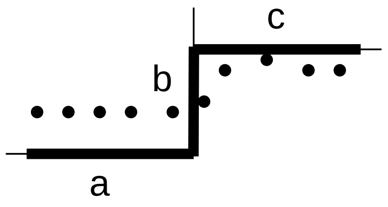


Figure 2.11: False positive case: actual trajectory that risks to be recognized as a case-1 artifact if the link road b is very short.

Looking only at the truth Table, conditions C and D seem to be redundant. In fact, the two binary conditions A and B provide the 4 combinations required to encode the 4 cases of interest. However, C and D are needed to distinguish Case 1 from false positives, like the one in Figure 2.11, which represents a short link road. Whenever none of the four artifacts is detected, the current run is assumed to be correct and its line labels are left unchanged.

2.2.4 Implementation and experimental results

This Section presents the experimental results obtained by applying the proposed map matching algorithm to enhance the quality of the SmartRoadSense road surface monitoring system.

2.2.4.A Implementation and experimental setup

The architecture of SmartRoadSense is composed of three components: a mobile application running on Android devices to compute every second a geo tagged estimate of a roughness index, a server that gathers roughness traces from all the end-user devices and maps them on OpenStreetMap, and a cloud-based front-end for graphical visualization of geospatial data, based on CartoDB.

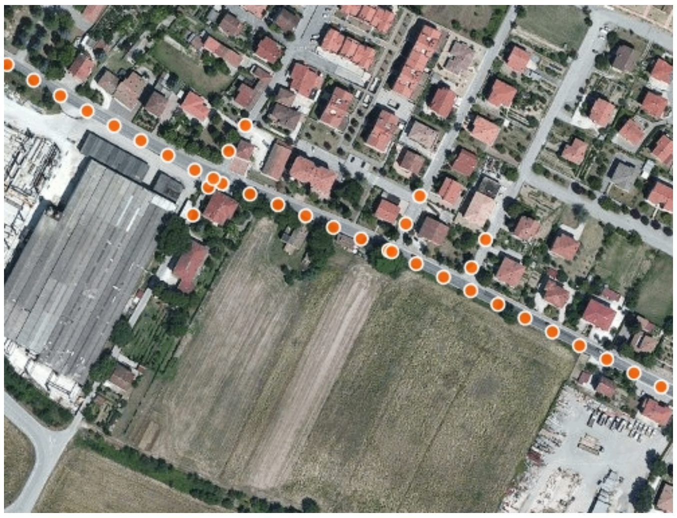
The server-side map matcher of SmartRoadSense, called *Cartesian matcher*, associates each trace point to the closest line, according to the 2-dimensional Cartesian distance. The proposed algorithm is implemented as an additional component, called *match enhancer*, which operates on the runs of trace points annotated by the Cartesian matcher.

Data points are collected by a standard web service endpoint and stored in a PostGIS database system. Both the Cartesian matcher and match enhancer are implemented as PHP scripts which operate on collected data by means of PostGIS stored procedures written in PL/pgSQL. Maps are taken from the OpenStreetMap project, while CartoDB is used to display data and double-check the correctness of the matches.

Although SmartRoadSense was conceived as a mobile crowdsensing application engaging end-users in road surface monitoring, the approach was validated on data collected for one month from Android devices installed on public buses operating on a known line in the Province of Pesaro-Urbino, in Italy. All the map elements along the actual bus line were manually sorted to build the ground truth baseline.

2.2.4.B Results

Figures from 2.12 to 2.16 show the results of map matching in the four misleading cases targeted by the proposed approach. Each Figure refers to a specific case and reports the trace points projected on the line segments in which they are mapped by the traditional Cartesian matching and by the proposed match enhancer. The effectiveness of the enhancer is apparent, in that the artifact are completely cancelled. In particular, Figure 2.12


(a)

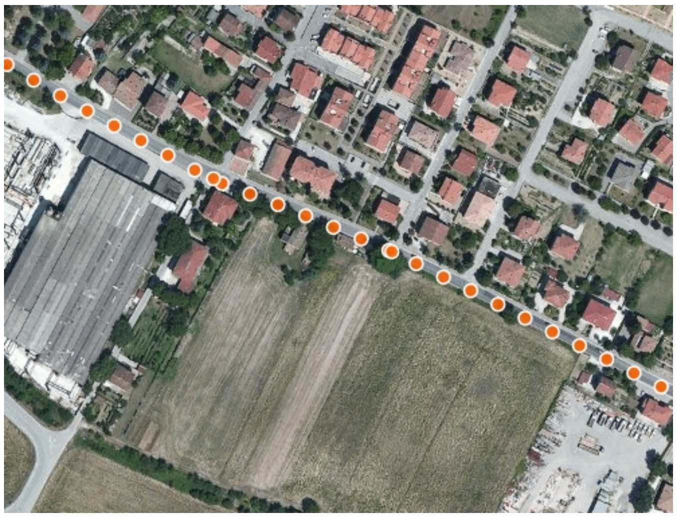
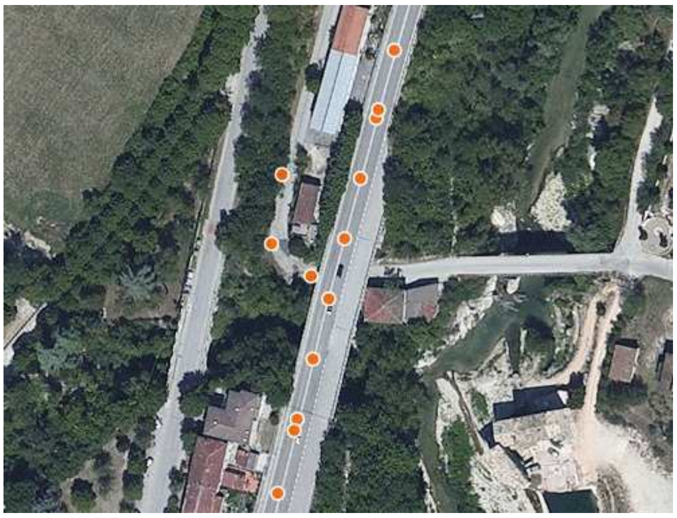

(b)

Figure 2.12: Projected trace points plotted on top of the satellite map showing examples of Case-1: (a) before and (b) after match enhancement.

shows several examples of Case-1 artifacts, Figure 2.13 shows an example of Case 2, Figure 2.14 shows an example of Case 3, while both Figure 2.15 and Figure 2.15 provide examples of Case 4.

Two metrics can be used to estimate the accuracy of map matching relative to ground truth: the percentage of trace points mapped on lines that do not belong to the bus line (miss rate), and the ratio between the number of wrong lines and the number of ground-truth lines involved in matching (fake line ratio).


(a)

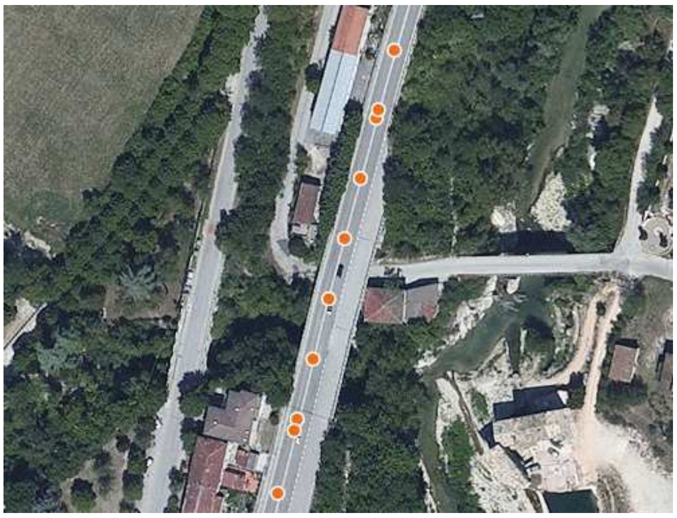
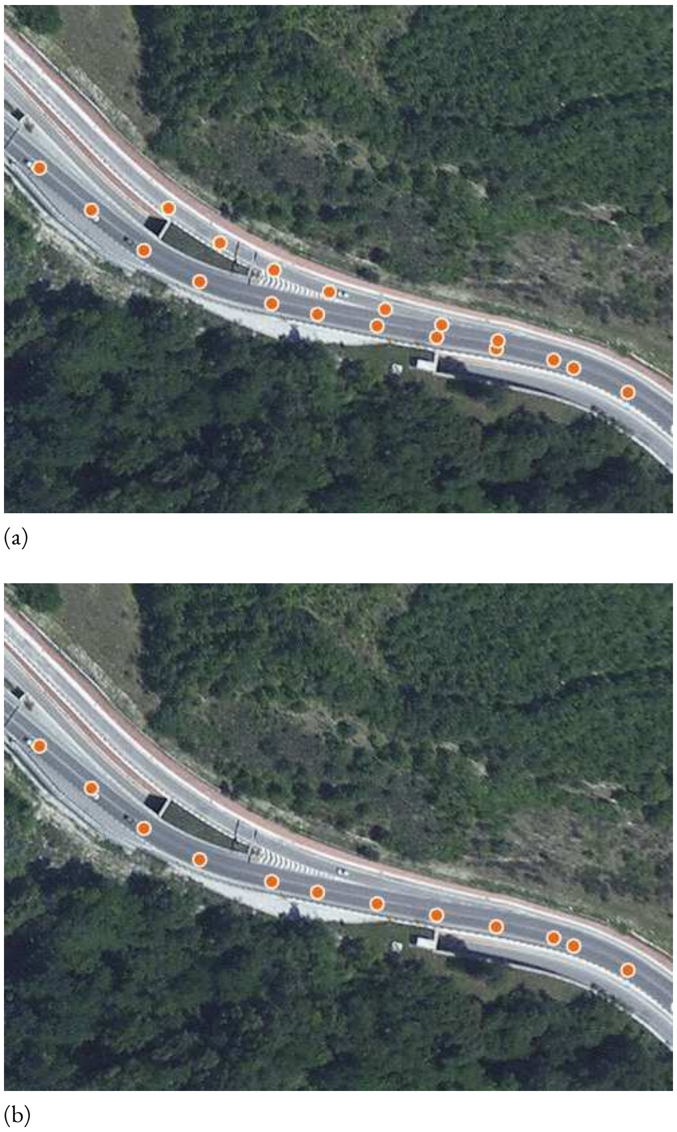

(b)

Figure 2.13: Projected trace points plotted on top of the satellite map showing examples of Case-2: (a) before and (b) after match enhancement.

Cartesian matching provided a miss rate of 2.85%, but in site of the small percentage of wrong matches, the fake lines involved in matching exceeded the number of ground-truth lines (108 against 102), leading to a ratio of 1.06. The difference between the two metrics is explained by the nature of the trace and of the map: the trace includes extra urban roads with a limited occurrence of artifacts, while the map is highly fragmented, containing many unclassified very short segments which are the most error prone ones.

The application of the match enhancement provided sizeable advantages, reducing



(a)

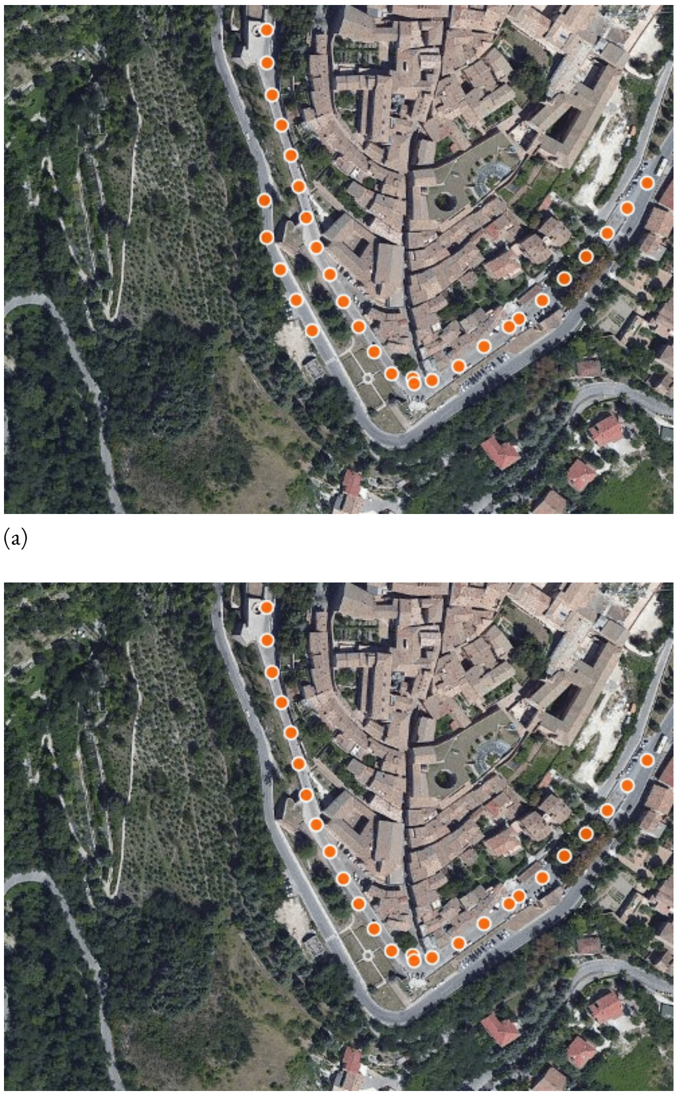
(b)

Figure 2.14: Projected trace points plotted on top of the satellite map showing examples of Case-3: (a) before and (b) after match enhancement.

the miss rate to 0.90%, and the fake line ratio to 0.47.

2.3 Final remarks

The MCS paradigm is inherently incline to producing heterogeneous data whose data quality can fluctuate. In this context, providing suitable approaches to deal with low or unknown data quality—in which case the collected data must be supplied with a quality


(a)

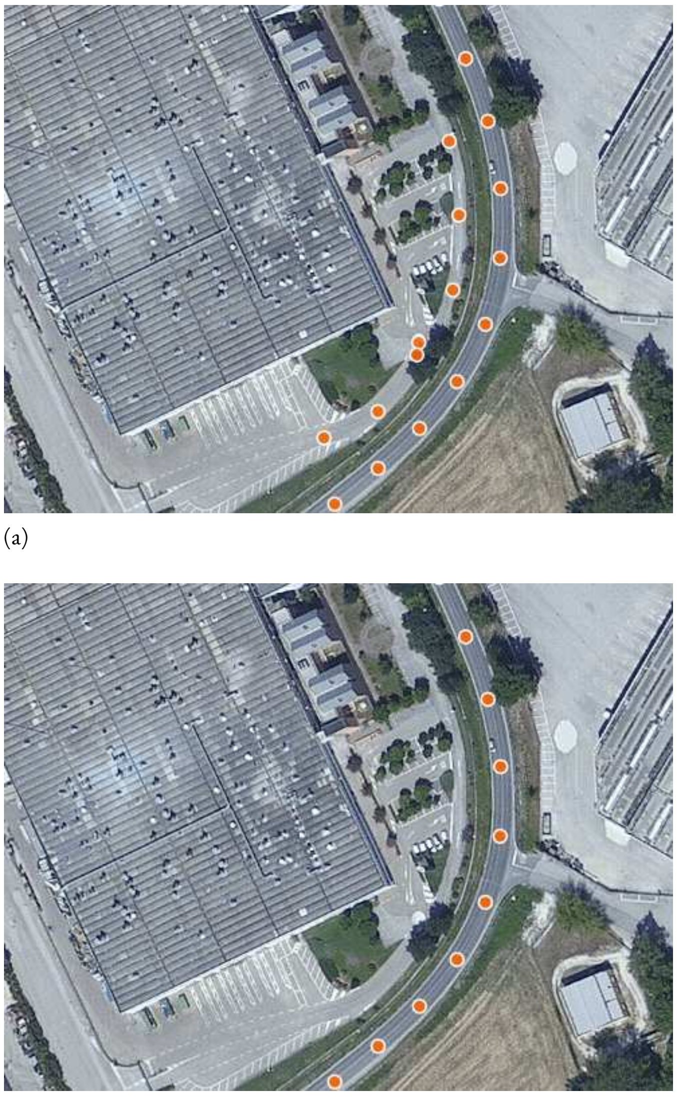
(b)

Figure 2.15: Projected trace points plotted on top of the satellite map showing examples of Case-4: (a) before and (b) after match enhancement.

indicator—is crucial. The addition of an index of data quality may also affect other stages of the system's internal processes.

In the first part of this Chapter, we treated the mobile crowdsensing activity as a distributed measurement process, in that, the estimate of the quantity to be measured is a composition of multiple distributed measurements performed by multiple users through their personal devices.

We reformulated the problem of associating a quality index to a measured value as


(a)

(b)

Figure 2.16: Projected trace points plotted on top of the satellite map showing examples of Case-4: (a) before and (b) after match enhancement.

an evaluation of uncertainty within a propagation framework. The complex interactions among variables together with the difficulty of determining their statistical features hamper the feasibility of applying analytic techniques or classical Monte Carlo simulations. We therefore proposed the use of a statistical tool called non-parametric bootstrap as a data-driven method to cope with the uncertainty propagation process without any preexisting knowledge (or hypothesis) about the statistical properties of data.

The effectiveness of our approach, regarding its accuracy, flexibility, and consistency, has been extensively demonstrated by empirical results we described. We also tried to demonstrate the potential of the proposed approach in an actual crowdsensing application by testing our method against data gathered by SmartRoadSense.

In the second part of the Chapter, we made use of the same MCS application to present an incremental real-time algorithm to enhance the accuracy of map matching in crowdsensing applications dealing with dense traces to be mapped on maps with unknown topology.

The issues raised by the abundance of tracepoints have been discussed and classified and efficient solutions have been proposed to handle them on the fly.

The effectiveness of the match enhancing techniques presented in this paper has also been demonstrated using SmartRoadSense as a real-world case study.

Experimental validation performed on known trajectories show that all types of artifacts are appropriately handled by the proposed algorithm with significant improvements in the overall matching accuracy. In particular, the percentage of wrong matches reduces from 2.85 % to 0.90 %, while the ratio between fake lines and correct ones decreases from 1.06 to 0.47.

Chapter 3

Privacy Preserving Incentives

Volunteer participation is pivotal to the success of any crowdsensing initiative. But taking part in a distributed mobile sensing task entails an investment in term of time, effort and mobile device resources (e.g., mobile data traffic, battery, and computational resources). Moreover, many MCS applications require a long-term commitment and even the willingness to share some personal data without providing any direct benefit to the participant. Offering some reward to contributors represents a crucial feature for MCS applications, where a lack of volunteer participation or interest can constitute a major obstacle to the application's success [115]. On the other hand, systems providing rewarding mechanisms are usually based on the concept of *user profile* and can require the collection of personal user data. That is to say that users willing to participate in order to be rewarded can lose interest in joining the sensing endeavor if they have to share their private data.

Table 3.1: Rewarding schemes classification in terms of motivation provided and anonymity of user information.

Motivation	Non-anonymous	Anonymous
Interest	Social inclusion, belonging, social good.	Enjoyment, entertainment, gamification.
Community	Reputation, trust.	Altruism.
Monetization	Virtual or *fiat* currency exchange, bidding, monetary transfers.	Vouchers, credits.

In Chapter 1 we gave a widely adopted classification of cooperation incentives commonly adopted in crowdsensing initiatives. In this Chapter, we analyze the same topic taking into account another aspect often crucial in MCS application: privacy.

To date, different rewarding schemes have been proposed. Some of them include privacy-preserving mechanisms preventing information about participants to be inferred during the reward assignment [88] Table 3.1 shows how most cooperation incentives can be broadly characterized in terms both of the type of motivation they are based on and whether they are compatible with anonymity-preserving mechanisms or not.

We speak of incentives based on *interest* if they rely on the volunteer's intrinsic motivation. Volunteers can join a sensing task because they are interested in the task itself or because they take enjoyment from participating. The latter can be the case when the task is playful by itself or because gamification elements have been employed in designing the task [46]. *Community* incentives are based on moral or ethical motivations. Sensing task can be promoted by communities by leveraging immaterial rewards such as reputation and trust while altruism and sense of community are the only reasons behind anonymous contributions. Rewarding mechanisms based on exchanging sensed data with money (either real or virtual), credits or any other type of financial-based rewards go under the *Monetization* category. As seen in Section 1.3.5, adopting this kind of incentives may expose the system to user misconduct as they may be willing to deceive the system to get higher rewards [72].

In this Chapter we will focus on incentives which allow anonymity-preserving mechanisms. In particular, gamification techniques and the exchange of vouchers. The following Section will discuss gamification as a method to make the sensing task more appealing and playful, as to engage as many volunteers as possible. In particular, we will see how gamification and anonymity can get along as major features of an MCS application. We will demonstrate how an MCS application with strict privacy requirements could be enhanced and made more appealing by adding a gamification layer without compromising the privacy-preserving techniques already in place.

Generally speaking, most MCS platforms exploit the participation of a number of volunteers to generate social value, pursuing goals oriented toward the common good. Incentive mechanisms are usually an essential part of any MCS application, but external stakeholders may also be interested in the common cause the platform is aiming to without wanting to play an active role in the sensing process. In the second part of this Chapter, we will introduce an inclusive rewarding platform designed explicitly for MCS endeavors. The proposed rewarding system acts as a bridge between volunteers, the MCS application, and third-party stakeholders, putting them in communication to trigger network effects and positive externalities that can help the pursuing of the common good. Our rewarding protocol is based on voucher exchanging mechanisms and will be described together with some implementation details.

3.1 A gamified approach to motivation incentives

Collective awareness is defined as an attribute of communities that helps them solve collective action problems, equivalent to the way that social capital is defined as an attribute of individuals that helps them solve collective action problems [15]. Collective awareness is a critical aspect within communities: it promotes collective action and crowdsourcing through citizen participation, which may lead to the accumulation of data and information essential for informing decision making and for solving difficult problems [143]. However, engaging a huge number of the general public in a crowdsourcing initiative is a mammoth task. There are existing initiatives that are exploiting the use of game mechanics for encouraging people to engage with tasks that produce real outcomes, such as Foldit [42], a citizen science game that uses the mechanics of puzzle solving for folding protein. Engaging citizens with tasks associated to real meaning and purpose taps into autonomy and agency, which are key for fostering self-directedness when it comes to education, social awareness, and civic participation. With these perspectives, we aim to explore an existing initiative for facilitating a user-engagement strategy for public-led, collective awareness platform for facilitating self-directed participation in stealth tracking of data within the context of transportation and urban planning. This paper specifically discusses the approach taken by the Crowd4Roads (C4Rs) project[1], which aims at combining smart sensing, ride sharing, and gamification applications to harness collective intelligence for providing open data towards boosting traffic conditions in Europe. Domains, such as transportation lies at the core of society, providing safe and easy solutions to individuals and businesses to connect actions and interactions that would otherwise be inaccessible [56, 89]. The infrastructure of road transportation in particular plays an important role in the global economic growth and activity, providing easy access to health and education, social benefits and offers connecting pathways for commercial ventures [150]. Passenger cars account for 73.7% of total intra-EU passenger transport [34], which indicates a strong public sector usage and dependence on the international network of road systems to pursue daily activities. With greater accessibility and improved transportation services, a global increase has followed in the demand for production services, which has consequently seen the rise of national and international consumption of resources [124]. Consumption practices, alongside substantial public usage of the road systems in the EU, has led to concerns of the impact that road transportation has had, and will lead to,

[1]The Crowd4Roads (C4RS) project "combines trip sharing and crowdsensing initiatives to harness collective intelligence to contribute to the solution of the sustainability issues of road passenger transport, by increasing the car occupancy rate and by engaging drivers and passengers in road monitoring." [141] For more information about the project please see http://www.c4rs.eu

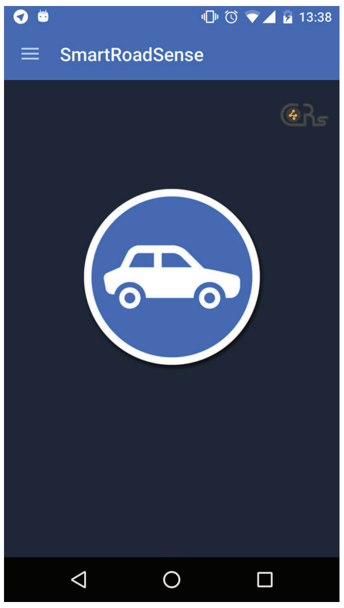
on the surrounding ecology and its effects on the environment. Current trends and predicted growth in traffic and road use have been estimated globally as an unsustainable venture, leading to an increased urgency of developing coordinated approaches to sustainable travel policies for preservation of the environment, business, health and social mobility for future generations [65]. A white paper issued by the British government on sustainable travel strategies, discussed these needs further and proposed working incentives to help bring about public behavior change to harmonize with government policy, the reduction of unnecessary road travel [47]. The goal of the C4Rs project is thus to engage the general public, i.e. car drivers and passengers, in making road transport more sustainable by exploiting SmartRoadSense, a crowdsensing system we already described in Section 2.1.1 that exploits the accelerometers of car-mounted smartphones as non-intrusive sensors of road surface quality, and increasing crowd participation in the initiative via gamified interfaces.

In this Section we also describe the games and gamification strategy for engaging end-users, which will provide both a direct effect, allowing the community to benefit from the service provided by individuals, and an indirect effect, being for sure the most effective way of raising awareness. Being involved in decision making when it comes to a local community and surrounding may nurture attitudes and behaviors required for a greater awareness of local transportation issues and roles of citizens in urban planning. SmartRoadSense has been already described throughout this work but in the next Section more details about it will be added in order to set the context for the following part of the Section. Then, the gamification strategy proposed for providing a more playful and engaging interface to the application will be outlined. This Section is concluded with reflections on the development considerations, providing relevant insights for the development of game-based interventions that rely on smart sensing and citizen participation on a mobile platform for sourcing big data.

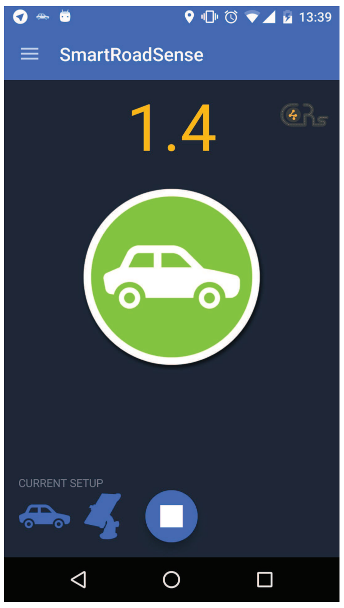
3.1.1 Smart sensing of road quality

SmartRoadSense is a crowdsensing platform, which aims to provide administrators, policy makers and citizens with up to date open-data on the roughness of the road surface network, to empower them to make better-informed road maintenance decisions. In Chapter 2, we already described the platform multi-tiered architecture sporting both a multiplatform mobile application client and a complex cloud-based infrastructure. SmartRoadSense enables users to collectively monitor the status of roads through their smartphones, just activating the mobile app while they travel on board of a vehicle, without any further interactions during the driving. The user interface of the mobile app has been designed trying to reduce as much as possible the number of actions required from users as they usually interact with SmartRoadSense just before and right after their ride.

The most common usage pattern is that users launch the app (Figure 3.1a) and push the main button to start tracking their journey (Figure 3.1a). Once the journey is over they push the stop button and close the app. Even if they forget to stop the tracking session, the app detects that they are not in a moving vehicle anymore and automatically stops the tracking. Recorded data will be further analyzed in a background process and sent to the server as soon as a suitable network connection is available. The collected data are anonymously transmitted to the server which stores and aggregates them extracting useful data about road surface status. These data are subsequently published as seen in Section 2.1.1.



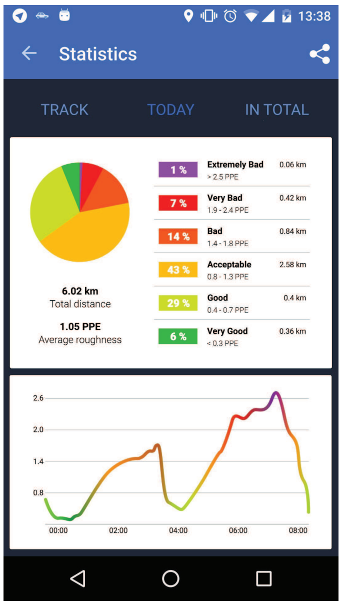
(a)



(b)

Figure 3.1: (a)The SmartRoadSense app initial screen (b)Tracking activated in app.

Each data represents a single measure of the roughness of the road in a particular time and point. These data are grouped in ordered sets called tracks, each one representing a portion of the roads travelled during the user trip. A track can be conceptually seen as an array of single data, ordered according to the instants when they have been recorded, each one containing a roughness measure along with the corresponding geographical position, the timestamp of the measurement and other minor information as the device model and the mobile operative system on top of which the app is currently being executed. The app does not collect, at any time, user personal information (e.g., name, email, address) as this may be harmful to the user privacy. All the tracks are temporarily stored in the smartphone internal memory with restricted access permissions. As data are collected, the app continuously updates a local database containing aggregated indexes about tracks. The database includes data on the SmartRoadSense usage such as the distance covered, the number of individual measurements elaborated, the time spent tracing, the average and maximum roughness levels registered, and the overall distribution of roughness values per level of driving comfort. This information is calculated as daily, monthly and overall user's statistics and are presented in a dedicated view of the app (Figure 3.2). Although the numerical values are not sent to the central server in any way, as this would mean to communicate data related to user's activity, through the SmartRoadSense mobile app users willing to do so can share their statistics as screenshots among major social networks.



1 %	Extremely Bad > 2.5 PPE	0.06 km
7 %	Very Bad 1.9 - 2.4 PPE	0.42 km
14 %	Bad 1.4 - 1.8 PPE	0.84 km
43 %	Acceptable 0.8 - 1.3 PPE	2.58 km
29 %	Good 0.4 - 0.7 PPE	0.4 km
6 %	Very Good < 0.3 PPE	0.36 km

Figure 3.2: Statistics view in SmartRoadSense Android mobile app

Also, the entire communication system between the mobile SmartRoadSense app and the server has been designed to allow users to gather and deliver data in a privacy-preserving manner. Once a track has been collected and stored, the app will try to send it to the server as soon as the connection of the mobile device allows it. To further weaken the relatedness between data and the user who provided them, tracks collected during a single journey are split into multiple chunks of consecutive measurements. Each chunk is then individually delivered to the server through a communication protocol inspired by zero-knowledge techniques [51]. This kind of method allows a prover to prove to the verifier that he knows a secret without revealing information about that secret. In the specific case of SmartRoadSense, data are delivered to the server without sending any sender's personal information, and the server responds with a secret token which can be afterwards used by the sender to claim its ownership of that burst of data. Each single communication is exchanged using standard asymmetric cryptography protocols, in order to maintain the exchange confidential.

3.1.2 Engagement strategy using gamification

This Section presents the gamification layer design to be used as a method of engaging a large and varied user base with the road sensing application.

With gamification's rise in prominence, particularly within the business and enterprise sectors, different definitions have been proposed. The more widely accepted include: "the process of engaging people and changing behaviour with game design, loyalty, and behavioural economics" [209], "the use of game design elements in non-game context" [46] and "the craft of deriving all the fun and addicting elements found in games and applying them to real-world or productive activitie" [30]. The use of game techniques has also been applied to a wide range of subjects, such as science [157], maths [63], foreign languages [43], cultural heritage [66], health [59], computer science [106], software engineering [168], business and logistics [154]. Gamification techniques are also employed to pursue transversal objectives, such as fostering participatory approaches and collaboration among peers [106], self-guided learning [185], completion of homework assignments [63], making assessment procedures easier and more effective [128], integration of exploratory approaches to learning [66] and strengthening student creativity [8].

American game designer Jane McGonigal, in her book [122] argues that advancing a *gameful* mind-set in the real world can produce effective and measurable change, leaning on modern research in positive psychology, to promote games as an integral factor contributing to human happiness, motivation, meaning and community development. Despite becoming "the public face of gamification", with this book McGonigal has distanced herself from the denomination (favouring the notion of *gameful design*) and its then emerging negative connotations. Indeed, with more and more scientific research and data complexifying the field after the initial (marketing oriented) enthusiasms, gamification is, as of now, a contested field of studies, raising objections from three main directions: 1) efficacy 2) ethical acceptability, and 3) techno-determinism. In 2016, Arnab suggested that "gamification is essentially an experience design tool focusing on motivations and emotions of the target group" [171], emphasising on the relationships between people, context and activities. It is essential not to de-humanise "the target users by using trivial mechanics in hope of engaging them as a common entity by performing player profiling instead and emphasizing motivations and emotions in order to establish an engaging user experience" [4]. The potential of games and gamification to foster change of perceptions and views and nurture positive attitudes and behaviors open up opportunities for the techniques to be used to achieve positive outcomes. Social games communities for instance are changing social interactions, leading to greater capabilities for social learning and interactions and importantly more fun in everyday contexts. There are existing games for social change and change of behavior such as Darfur is Dying[2], Evoke[3], PR:EPARe[4] [32], RecycleBank[5], Foldit[6], The Walk[7], FitBit[8], Khan Academy[9], CrowdRise[10], and SuperBetter[11] that are focusing on sensitive, subjective and/or current issues. For instance, Foldit exploits collective awareness and citizen participations in solving challenges associated to protein folding, which have helped scientists to further their research and Evoke encourages the general public to undertake real missions that are linked to real world issues and real life applications.

The C4Rs project hopes to draw on the ideas of developing positive psychological motivation to engage users to interact with the road sensing application. Future pilot studies will be conducted to evaluate the engagement factors of the app with and without the gamified layer to provide an analysis on the success of using gamification in the Crowd4Roads project. Presented below is a first iteration of the game design template which will inform the gamification layer.

3.1.2.A Harnessing the potential for narratives and role playing

The engagement strategy relies on the simple mechanics of pervasive quests and the engaging characteristics of stories and narratives towards fostering self-directedness. The gamification layer for C4Rs is based on the genre of Role Playing (RP) game systems with elements of interactive narrative fiction games. The core concept is that the player will take on a semi-customizable character (player choice from a few pre-defined characters), and complete game missions to level up their character's reputation and gain in-game collectable items.

Following the outcome that our targeted users for the C4Rs app will likely be a very varied user group with no discernable connected interests, it is intended to make the gamification layer as open to different tastes and preferences as possible, in a similar development style of a MMORPG (Massively Multi-player Online Role Playing Game) such as World of Warcraft[12]. Players will be tasked with completing missions in 'space'

[2] Darfur is Dying: http://www.gamesforchange.org/play/darfur-is-dying/
[3] Evoke: http://edition.cnn.com/2010/TECH/03/01/evoke.game.africa.poverty/
[4] PR:EPARe: http://www.seriousgamesinstitute.co.uk/games/prepare.aspx
[5] RecycleBank: http://www.recyclebank.com
[6] Foldit: https://fold.it
[7] The Walk: http://www.thewalkgame.com
[8] FitBit: https://www.fitbit.com
[9] Khan Academy: https://www.khanacademy.org
[10] CrowdRise: https://www.crowdrise.com
[11] SuperBetter: https://www.superbetter.com
[12] World of Warcraft: https://worldofwarcraft.com

using different real world locations as unlockable areas for new in-game world quests. The game will use narrative, humor and player choice to encourage player investment and drive story. In-game items are connected to the quest story's and are given a value system based on rarity, encouraging players to try and retrieve higher value in-game items for character reputation and re-sell for real world rewards. Presented below are three core considerations of game mechanics that are to be used in the development of a RP game system for the C4Rs' road sensing application

3.1.2.B Pervasive Quest/Story Unlocks: Road Distance Travelled

Exploiting the current popularity of the pervasive nature of games, such as Pokémon Go[13], the direct connection between the gamification layer and the road sensing applications is the trip information, where game quests are unlocked through travelling around in the car in the real world. Much like, location-based services will be used to track the distance a player has travelled in the car with the road sensing app on. The design aims to increase and improve the player's interaction and agency within the gamification as well as considering other gameplay changes which we intend to make. It will cover the basic flow of *Quest Packs*, how the player will be asked to make choices and the basics of *Encounters* as well as talking about *Rewards* and *Items*, and their use within the gamification system.

Users of the SmartRoadSense application will be awarded credits, which can be used within the SmartRoadSense Gamification system, whenever they collect data that is then uploaded to the SmartRoadSense data servers. There are two types of credit which can be collected, these are defined as *Quest Credits* and *Gold Credits*. Quest Credits are awarded for collecting data on *Standard Roads*; these are roads which are not considered to be special by the application. Gold Credits are awarded for travelling on *Special Roads*. Special Roads are roads for which little or no data has been collected. Each type of credit will be gathered at specific rates (see Figure 3.3):

Quest Credits One Quest Credit will be gained by the user for every 1 kilometre of data collected on a Standard Road.

Gold Credits One Gold Credit will be gained by the user for every 1.5 kilometres of data collected on a Special Road.

[13]Pokémon Go: http://www.pokemongo.com

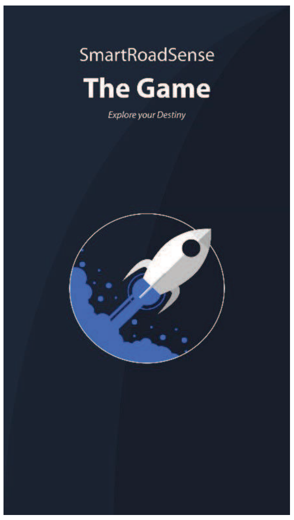
Table 3.2: Number of Quests in a Quest Pack vs. Quest Pack Cost in Quest Credits.

No. of Quests	Quest Pack Cost
3	150
4	200
5	250
6	300
7	350
8	400
9	450
10	500

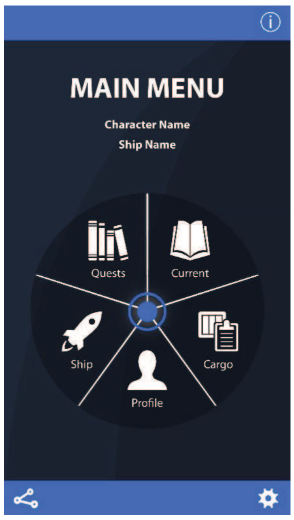
Quest Packs form the backbone of the SmartRoadSense gamification. Each Quest Pack will contain from three to ten *Quests* which form a storyline. As the player progresses through each Quest they will be confronted with *Options* and Encounters. The player must surmount these to progress to the next part of the storyline and so on towards eventual victory. The player will be provided with one to three unlocked Quest Packs when they first download the application. Any remaining Quest Packs will need to be unlocked using Quest Credits. Quest Credits are gained by the player whenever they travel and use the application to gather data. Each of the Quest Packs costs a specific amount of Quest Credits which is determined by the amount of Quests it contains. Table 3.2 shows the relationship between the amount of Quests within a Quest Pack and its cost; this will be modified during playtesting.

Additional to the distance travelled mechanic associated to Quest credits, there will be another travel/location based mechanic of unlocking new/special quests through players tracking information to places that are less well travelled in the real world (Gold Credits). Gold Credits will allow the player to purchase quests that will have a higher than average chance of receiving rare items. This mechanic will be used to encourage players to travel on less known roads for higher gain quests and to feed-back data on areas that little is known about.

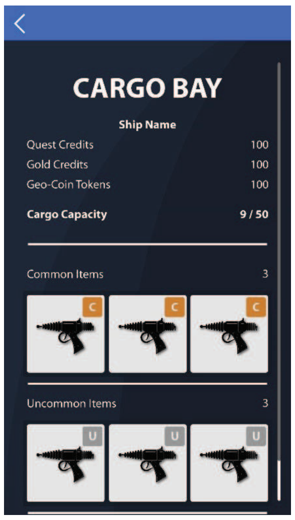
These mechanics, that are connected to the tracking system of the C4Rs application, use scarcity through item rarity, collecting, chance and narrative to drive player engage-



(a)

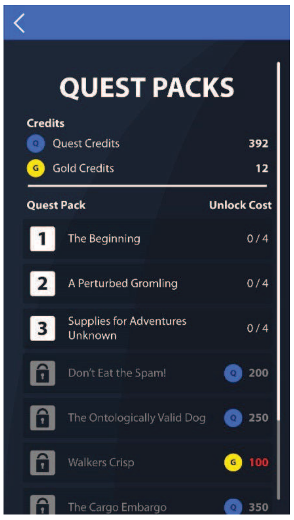


(b)



Quest Credits	100
Gold Credits	100
Geo-Coin Tokens	100
Cargo Capacity	9 / 50
Common Items	3
Uncommon Items	3

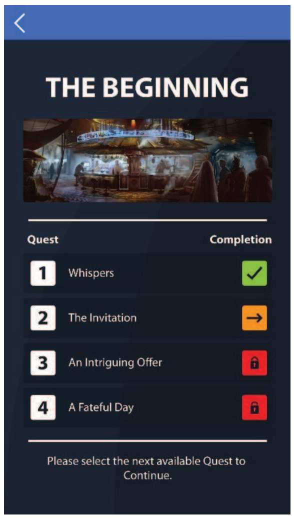
(c)



Quest Credits	392
Gold Credits	12

Quest Pack	Unlock Cost
1 The Beginning	0 / 4
2 A Perturbed Gromling	0 / 4
3 Supplies for Adventures Unknown	0 / 4
Don't Eat the Spam!	Q 200
The Ontologically Valid Dog	Q 250
Walkers Crisp	G 100
The Cargo Embargo	Q 350

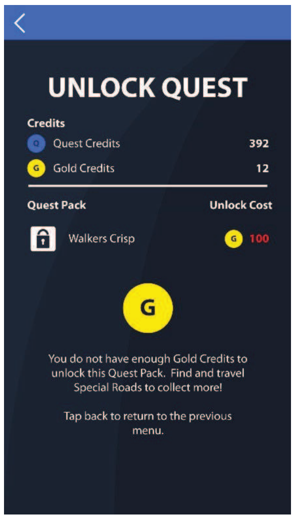
(d)



Quest	Completion
1 Whispers	✓
2 The Invitation	→
3 An Intriguing Offer	🔒
4 A Fateful Day	🔒

Please select the next available Quest to Continue.

(e)



Quest Credits	392
Gold Credits	12

Quest Pack	Unlock Cost
Walkers Crisp	G 100

You do not have enough Gold Credits to unlock this Quest Pack. Find and travel Special Roads to collect more!

Tap back to return to the previous menu.

(f)

Figure 3.3: Examples of in-game screens. From (a) to (f) Loading Screen, Loading Bay, Cargo Bay, Quest Packs, Quest List, Story Encounter and Quest Unlock screens.

ment. Primarily, players should be motivated through their desire to gather and complete collections of in-game items to travel greater distances using the application and to explore routes of travel that they may not normally have considered. These mechanics should appeal to the collector, explorer and social player types. Competition could also be a driving factor of motivation for players as once items are gathered into a player's collection, player's main gain will be "social recognition" for acquiring these items amongst friends.

Initially, the team considered using the road roughness level data to equal the games quest difficulty score, however, after much deliberation it was decided that distance travelled may help gather more quality data due to issues surrounding ability to stop players cheating. The team was concerned that players could alter the data more easily in relation to the road roughness and therefore the game rewards could be open to manipulation from players, skewing the data that the C4Rs server received. In light of this, the team decided that it was much easier to track and control manipulation of distance travelled, than to control the road roughness level of the application.

To formally set distance parameters to the quest difficulty gauge, the team will has looked at pre-pilot data already available on the Crowd4Roads server to determine average travel distances. We shall also highlight less travelled areas and use this as a basis to develop a prototype of a special quest.

3.1.2.C Player Choice (Quest Stories & Outcomes)

Players are presented with various difficulties of quests throughout the game. Each quest starts with a background scenario which will be presented as a written screen with the potential for a voice over/audio option. The background scenario will present the player with a scenario in which they have a choice of how they may respond. Players are then given the option of what they would do in that scenarios situation. Choosing from one of two/three options, players are then rewarded/penalized based on their choice. This mechanic is often used in board games and referred to as a *crisis deck* (examples of these board games include Battlestar Galactica[14] and Dead of Winter[15]).

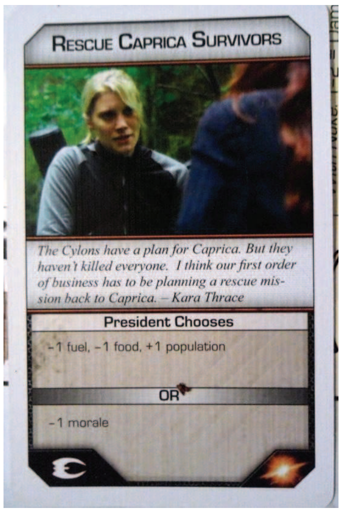
By using non-linear narrative mechanics to support the motivations of player choice, players feel a greater sense of involvement in the games story and players can control how they are developing their character. Players also feel a greater sense of achievement when rewarded with a higher value item due to the fact that they have had a direct input into acquiring that item. Players who make poor decisions through the quests offered may learn from their previous decisions and start to choose options that they would not have previously considered.

3.1.2.D Core Reward System

Listed below is a set of perceived core rewards systems that will engage and reward the player for interacting the C4Rs application.

- Treasure In-Game Items: Items are awarded from quests and completing quest packs. Each Item will be assigned a rarity factor and a value factor. The rarity fac-

[14] Battlestar Galactica: https://www.fantasyflightgames.com
[15] Dead of Winter: https://www.plaidhatgames.com/games/dead-of-winter


RESCUE CAPRICA SURVIVORS

The Cylons have a plan for Caprica. But they haven't killed everyone. I think our first order of business has to be planning a rescue mission back to Caprica. – Kara Thrace

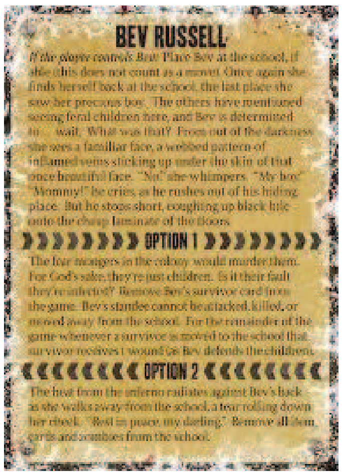
President Chooses

–1 fuel, –1 food, +1 population

OR

–1 morale

(a)


BEV RUSSELL

If the player controls Bev: Place Bev at the school, if she (this does not count as a move). Once again she finds herself back at the school, the last place she saw her precious boy. The others have mentioned seeing feral children here, and Bev is determined to ... wait. What was that? From out of the darkness she sees a familiar face, a webbed pattern of inflamed veins sticking up under the skin of that once beautiful face. "No," she whimpers. "My boy." "Mommy!" he cries, as he rushes out of his hiding place. But he stops short, coughing up black bile onto the cheap laminate of the floors.

OPTION 1

The fear mongers in the colony would murder them. For God's sake, they're just children. Is it their fault they're infected? Remove Bev's survivor card from the game. Bev's standee cannot be attacked, killed, or moved away from the school. For the remainder of the game whenever a survivor is moved to the school that survivor receives 1 wound (as Bev defends the children).

OPTION 2

The heat from the inferno radiates against Bev's back as she walks away from the school, a tear rolling down her cheek. "Rest in peace, my darling." Remove all item cards and zombies from the school.

(b)

Figure 3.4: Board Game examples of Player Choice/Narrative Mechanics.

tor will be connected to quest difficulty. Rarer Items can also be located through completing quest packs and exploring unknown areas. Items are also assigned an item value for an in-game monetary system. Items can be traded in or exchanged with other game rewards based on the value of the item. A concept of this rarity and value system is shown in Table 3.3.

Items are the core reward system within the C4Rs gamification strategy, targeting collectors (completing sets, gathering items), merchants (players interested in financial/monetary systems in games) and social exhibitors (social media sharers).

- Vouchers collection: The exploiting of a novel voucher exchange system is an additional idea for supporting the transfer between in-game rewards to real world rewards. It was suggested that items and Gold Credits could be turned into a value of Geo-Coins that would be stored in another separate application known as a "wallet". Vouchers could then be traded for real world rewards such as free parking or vouchers. This would all be dependent of collaboration with external stakeholders and interested parties for providing these real-world rewards. In-game Items gathered from quests can be sold if the player does not want to retain specific items or has multiples of same item. This may provide an extrinsic motivation for players who are not interested in the in-game rewards system. The voucher reward platform will be further discussed later in this Chapter.

- Pets/Oddity/Collectibles: Pets are another in-game item that is used by many RP games. Pets are collectable items that often grant bonuses to a character or pro-

Table 3.3: Concept of Item Rarity and Item Values.

Quest Difficulty	Item Rarity	Item Value
Easy	Common	10 Coins
Medium	Uncommon	50 Coins
Challenging	Rare	100 Coins
Hard	Epic	250 Coins
Epic	Mythic	500 Coins

vide an aesthetic value for the player. Pets could be included as rewards on certain quests, completing quest packs or exploring unknown areas. Much like treasure items they add another level to the collection mechanic.

- Social Media Sharing: To ensure that any in-game items have extra external value and promote the sense of competitive collection in players, all items/pets once gained, should be made available to be shared on most social media platforms. This will help promote competition elements within the gamification layer and encourage social acceptance in player motivations.

3.2 A flexible rewarding platform

Independently from its particular purpose, any public and successful MCS system is able to put the exceptional sensing and communication capabilities of mobile devices to the service of a common goal shared by a community of users, thus encouraging active citizenship. MCS platforms generate an intrinsic social value which can be capitalized by the application provider, by the participants, or by third-party stakeholders. Indeed, external organizations and communities could be interested in the results of the crowdsensing process without giving relevance to the process itself.

End-user engagement in MCS systems has been treated in several pieces of research [203, 109, 86, 61, 188, 181, 196, 88, 115]. Most of these studies address the cooperation incentive mechanics as one (essential) feature of a crowdsensing platform, just like any other integrated feature (e.g., data sensing and gathering). However, even if closely related, binding together aspects like data collection and reward mechanisms could restrain the network effect and reduce positive externalities, keeping potential stakeholders out of the loop [164].

Rather, the separation of the crowdsensing process from reward mechanics could allow external stakeholders to allocate incentives in order to stimulate the citizen's motivation in contributing to one or many MCS initiatives. In this framework, the decoupling is intended to help in applying platform economy concepts to collective aims [140].

Using real currency to reward participants is the most flexible and direct method of paying back user efforts, but dealing with *fiat money* can lead to possible complications. For instance, Ogie have demonstrated that "an inverse relationship between privacy and monetary reward can be gleaned i.e. the higher the money involved, the greater the need to collect personally identifiable information of participants and hence, the lesser the privacy" [137]. Moreover, when money is involved, volunteers are more likely to trick the system to get more benefits [72]. Ensuring trustful contributions while preventing misbehaviors (e.g., forgery, double spending, cheating, or speculation) can be cumbersome [14].

In this context, another fair and homogeneous way of evaluating and attesting the effort devoted by individuals to the collective aims is required, making public engagement work as *fiat* money [137]. A variety of third-parties could adopt this kind of effort-based currency while eschewing the complications of using real currency.

In the following, we propose a general purpose rewarding platform based on a voucher exchange system, specially designed for public MCS application users. The platform has been conceived in respect to the peculiar features of contributions in typical MCS applications (outlined at the beginning of Chapter 2), while respecting user anonymity and enabling a thorough decoupling between MCS instruments and participation incentives.

3.2.1 Proposed system

The slim architecture of our proposed system makes it suitable to provide a general and inclusive framework, to implement cross-platform policies and strategies.

Contributions are evaluated in terms of time and resources consumed in the volunteer's effort, and on its geographical location (to make those who provide rewards able to identify areas of major interest). Contributors are awarded vouchers, as a compensation for the resources spent in the sensing activity. These vouchers are intrinsically tied to a geographical position (i.e., where was the contribution generated?), a timestamp (i.e., when was the contribution made?), and a purpose (i.e., which MCS application was used by the user and what is its goal?).

From a privacy-oriented perspective, these vouchers are conceived to be entirely anonymous and not to include any additional information about the user. They can be freely exchanged among users as they are not bound to any to a particular individual. To prevent double spending and user misbehaviors the consumption of vouchers is overseen by a central authority, but their ownership is neither tracked nor exclusive.

3.2.1.A Definitions

Our proposed system takes care of the following operations:

(a) Generating vouchers;

(b) Making vouchers available to the intended recipient;

(c) Verifying voucher validity;

(d) Exchanging vouchers between peers.

Definitions we will refer to in the following description of the platform's operations are shown in Table 3.4.

Aim	One of the goals of a crowdsensing platform taking part in the proposed voucher system.
Volunteer	Individual that invest time and effort in order to pursue the common aim.
Registry	Central authority that issues and registers vouchers.
Instrument	Tool provided by a crowdsensing platform and used by volunteers in order to perform work, which is then rewarded through the generation of vouchers.
Merchant	Third-party interested in one or more aims pursued by platforms of the voucher system, which may exchange vouchers for goods, services, or other rewards.
Pocket	Tool controlled by the volunteer that collects vouchers.
Point of Sale (POS)	Technical end-point that allows merchants to accept vouchers.

Table 3.4: Definitions used to describe the proposed voucher system.

3.2.1.B Protocol description

The protocol can be split into two main parts. *Voucher generation* includes operations (a) and (b) mentioned above. *Payment* includes operations (c) and (d). Both procedures are explained in more detail below.

Voucher generation. The procedure is triggered by the user contributing to a common aim. Contributions can be provided in terms of data transmissions or any other form of trackable effort performed by the user or the instrument. As shown in Figure 3.5, to compensate the volunteer's efforts a number of vouchers must be generated.

The generation procedure is started by the *instrument*, which requires new vouchers from the *registry* after being used by the *volunteer* in order to contribute some work towards the platform's *aim*. The *registry* is responsible for issuing the vouchers and delivering them to the volunteer's *pocket*.

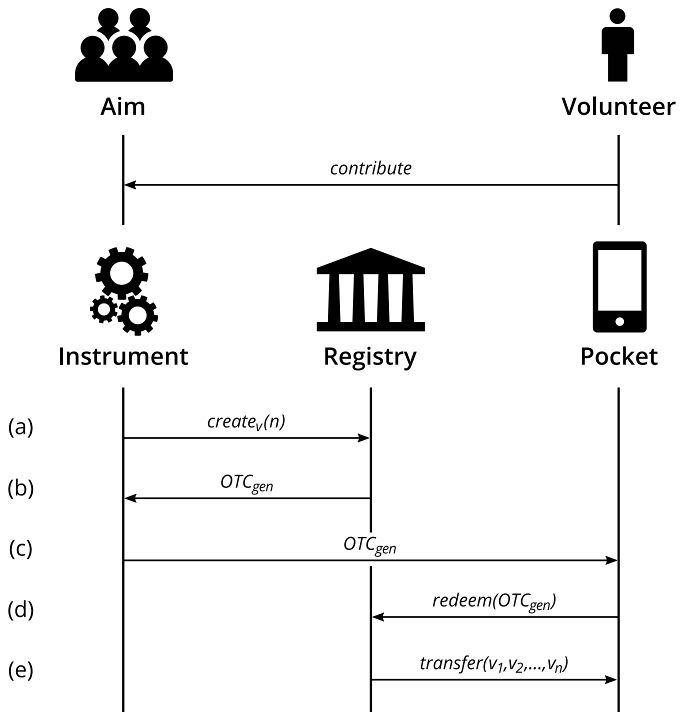


Figure 3.5: Voucher creation protocol and transfer to the volunteer's pocket.

Following the actions enumerated in Figure 3.5, the generation protocol is articulated as follows:

(a) The *instrument* registers the *volunteer*'s contribution and requests the generation of n vouchers.

(b) The *registry* permits the emission of the new vouchers and generates a one-time code, OTC_{gen}, in the form of an URL with including a unique code.

(c) The *instrument* forwards OTC_{gen} to the volunteer's *pocket* (e.g., a mobile application).

(d) The *pocket* uses the one-time code to redeem the vouchers from the *registry*.

(e) The *registry* verifies that the one-time code is valid and that the request can be satisfied. The vouchers are generated and transferred to the *pocket*.

Payment. Vouchers earned by *volunteers* has to be exchanged with a third-party *merchant* to be of any use. The *merchant* can provide the *volunteer* with goods or services in exchange for vouchers. Figure 3.6 shows how this exchange happens between the merchant's *point of sale* and the volunteer's *pocket*.

The procedure can be detailed as follows:

(a) The merchant's *point of sale* creates a new "payment instance" through the *registry*. This can happen before or after that a *volunteer* manifests his or her desire to obtain some goods or to access a service. The POS indicates the voucher *filter* (see Section 3.2.1.D), the number of requested vouchers k (i.e., the "cost"), a payment location URL_p, and an (optional) acknowledge location URL_{ack} (see Section 3.2.1.E);

(b) The *registry* generates a new one-time code OTC_{pay} for the payment;

(c) The *point of sale* forwards OTC_{pay} to the volunteer's *pocket* (i.e., a mobile application);

(d) The one-time code is verified by the *registry* which also add data about the payment's *filter* and k are retrieved. The *pocket* determines whether the payment can be satisfied (i.e., enough vouchers satisfying *filter* are present) and allows the user to accept the payment;

(e) k vouchers are transferred to the *registry* and are considered to be lost to the *pocket* and the user;

(f) The payment is confirmed by invoking URL_p;

(g) Optionally, the *registry* invokes URL_{ack} to acknowledge the payment to the *merchant*.

3.2.1.C Voucher format

The *registry* generated vouchers upon request by *instrument* in the form of simple textual JSON-encoded files. Each file represents a voucher signed with the private key of the *registry* to have a proof of its authenticity.

A single voucher v_i includes the following information fields:

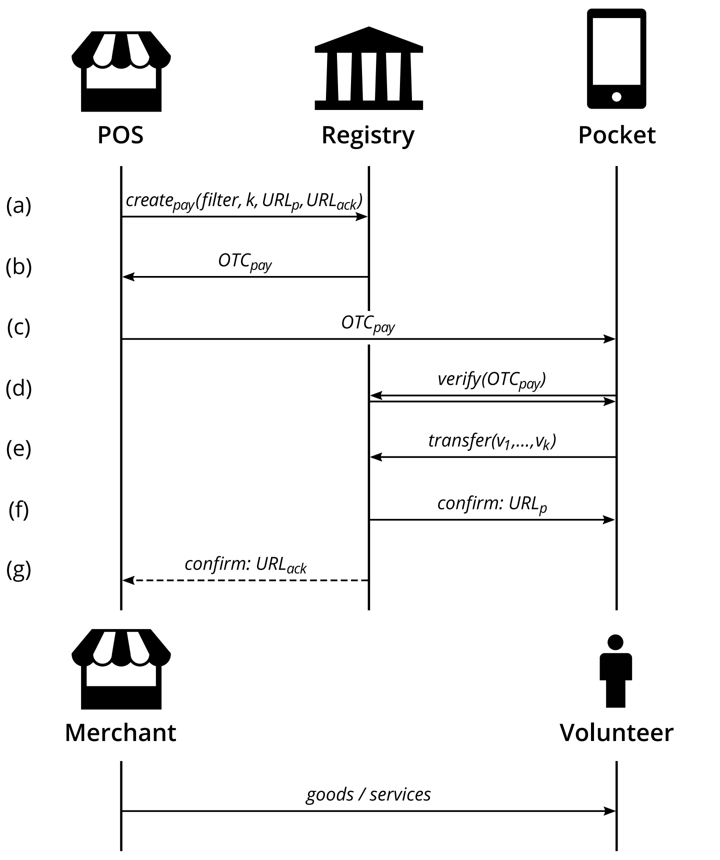


Figure 3.6: Payment protocol.

$$v_i = K_{priv}(ID_i, lat_i, lng_i, ts_i, URL_{gen})$$

Where K_{priv} represents the signature operation performed on the voucher payload utilizing the *registry* private key. The ID_i is a globally unique identifier (GUID) for the i-th voucher. lat_i, lng_i respectively represent the latitude and the longitude value of the voucher's geographical position. ts_i stands for the voucher's timestamp. It indicates when the contribution happened (i.e., data are received by the *instrument*'s server). The *instrument* responsible for issuing the voucher is represented by URL_{gen}, i.e., an URL known and registered by the *registry* that uniquely identifies the *instrument*.

3.2.1.D Voucher filtering

Figure 3.6 shows how new transactions (in the form of a one-time code) are triggered by the merchant's *point of sale* and generated by the *registry*. The *POS* is also responsible for specifying transaction details, such as which kind of vouchers and how many of them are requested as payment.

In the proposed system each voucher has a unitary non-fractional value. To simplify the system, single vouchers cannot be further split and can either be spent entirely or not at all (i.e., the payment procedure does not contemplate changes). For each payment, the required amount of vouchers has to be specified. In the protocol description, described in Section 3.2.1.B, such amount is indicated as k.

Merchants can be interested in rewarding efforts carried out, with through a certain *instrument*, within a specific geographical region, and during a particular timespan. For instance, local authorities could be interested in data collected in their urban areas while services releasing continuous data updates may decide to incentivize work done in the last few hours. All these *merchant*'s preferences can be expressed by a voucher filter. The role of such a filter is to exclude non-compliant vouchers from being used for a payment. A *voucher filter* uses a *white-list* approach and can include any number of the following criteria: (a) identifier of the *instruments* that generated the voucher, in the form of an URL (see previous Section); (b) geographical structure in the form of a GeoJSON[16] polygon; (c) time reference as an absolute timespan or a relative expression.

Looking at the step (d) of Figure 3.6, we can see how the *pocket* itself applies the *filter* on the vouchers available to the user and determines whether the payment can be completed or not. After which, the *registry* confirms that the vouchers selected by the *pocket* are compliant with the filter.

The *Pocket* is responsible for selecting the vouchers requested for a payment. Vouchers can be picked randomly among those that satisfy the criteria, or they may be selected manually by the user.

3.2.1.E One-time codes and URLs

Figures 3.5 and 3.6 show how both voucher creation requests and payment instances waiting to be completed make use of *one-time codes*. In both cases, one-time codes identify a pending voucher operation on the *registry*'s side. In the generation protocol, the user makes use of them to redeem new vouchers, while in the payment protocol, codes are used to complete a payment.

More specifically, a one-time code is a textual unique identifier associated to an unique operation. A one-time code could take the form of a URL using HTTPS as a schema and point to a registry-owned host. The URL path may contain the unique operation identifier.

For instance, a simple one-time code could have the following shape:

[16] `GeoJSON:http://geojson.org/`

```
https://registry.com/payment/7d9bd006
```

When *instruments* and *pockets* are implemented as mobile applications, having one-time codes as HTTPS-based URLs is particularly useful since most mobile app platforms allow applications to register URLs as *deep links*. In that, on mobile platforms such as Android or iOS, a deep-link can be used to launch the target mobile application automatically, also supplying custom launch parameters (i.e., the one-time payment unique code).

In case the mobile client application is not installed on user's smartphone, having one-time codes implemented as URLs provides a fallback mechanism since codes will be opened as common URLs in the default mobile browser (in those cases the URL could link to a landing web-page that prompts the user to install the app). Security for mobile deep links can be ensured by using "App Links" on most modern mobile platforms [107].

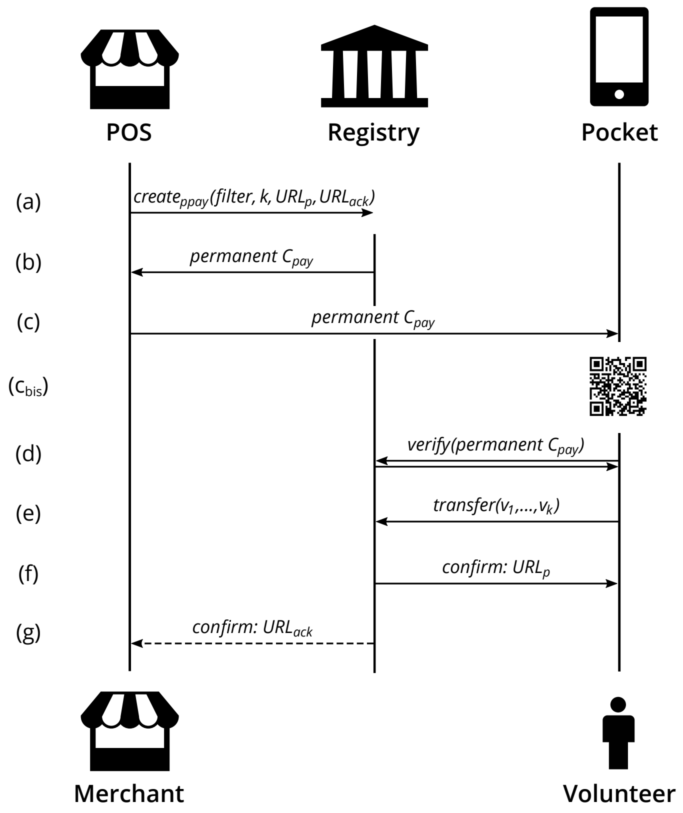


Figure 3.7: Offline payment protocol.

The mechanism of one-time code, used for voucher generation and payment protocol, relies on the fact that they are discarded right after being used for the first time. However, the payment protocol shown in Figure 3.6 can be changed to be used in offline mode. In this scenario payment instances are created once and marked as "permanent" (without the need of an active *point of sale*). Figure 3.7 shows how the *registry* returns a *permanent payment code* which does not expire after the use and can be reused multiple times, by different users. Being a persistent URL, the *merchant* could encode the code into a QR Code and display it in physical locations where users can scan it at any time — *(*c_{bis}*)* in Figure 3.7 — without requiring an online back-end service.

3.2.1.F Platform growth

The proposed voucher-based rewarding platform is designed in order to attract both crowdsensing systems (whose aims usually are related to pursuing the common good and are backed by a community of volunteers) and third-party stakeholders (which may be interested in supporting a cause or rewarding active citizenship without taking an active part in the MCS process).

In our platform, voucher creation is linearly dependent on the amount of effort devoted by volunteers. That means that the system does not impose an upper bound on the number of existing vouchers. Consequently, the risk of inflation is high then some control over the behavior of voucher creators must be exercised (also to guarantee a fair treatment of volunteers in spite of the diversity of the aims they pursue and in spite of the MCS instruments they use).

On the other hand, hoarding is discouraged by this inflationary risk and by the fact that vouchers are rigidly tied to the units of work performed by volunteers.

The central authority managing the rewarding system is responsible for approving new crowdsensing platforms (and their aims). A transparent ethical committee should be in charge of such evaluation, with the purpose of assessing the pertinence of any new MCS system joining the platform.

To ensure that generated vouchers are proportional to the effort provided by volunteers, *instruments* provided by joining MCS must be checked for technical correctness. Also, *instruments* have to be registered to *registry* in order to request new vouchers and to be associated with a unique public identifier.

On the other hand, there is no need of an evaluation process for *merchants*. They can freely choose how many (and which kinds of) vouchers they require in exchange for the service provided. However, while any merchant's *point of sale* can freely join the system by exploiting the platform's open protocol, technical requirements impose that they need to register themselves to the *registry*. Upon such registration, the *registry* provide them with an API key *points of sale* use for communicating. The *registry* could disable a misbehaving POS if needed.

3.3 Final remarks

In this Chapter, we confronted the problem of providing users with appealing incentives to engage them in the sensing endeavor. We discussed different approaches from the privacy point of view, proposing an original rewarding schemes classification in terms of motivation provided and anonymity of user information. We then discussed the two most promising paradigms between those identified as compatible with strict privacy-preserving mechanisms: gamification techniques and voucher-exchanging systems.

In Section 3.1 we presented an overview of the potential of engaging the general public to participate in providing useful data on roads through the road sensing app and the proposed gamification layer. Following the C4Rs research conducted on identifying potential target end users, the gamification method that has been proposed utilizes a number of core game mechanics such as collection, trading, and choose your own adventure stories to appeal to a wide-reaching audience. This is due to no *core* target audience being identified and the need to garner appeal on a scale similar to an MMORPG. We predict that the gamification design proposed in this paper will, however, appeal to younger members of the community, with a hope that this may inspire more consideration on developing healthier travel behaviors for future generations.

It is worth mentioning that all the game mechanics discussed only rely on data already present on-board of the player's smartphone. In that, the game requires no further data disclosure other than that already requested by the C4Rs platform functioning. In this framework, the social media sharing mechanic is not essential to the gameplay. Moreover, volunteers/players may willingly share their in-game achievements to the public but the game metaphor is abstract enough to conceal any real-world user personal data, and in doing so, it helps in preserving the anonymity of the contribution.

We acknowledge the theoretical and design proposal nature of what presented so far. Future works will include pilot testing and feedback of the proposed gamification strategy against that of the use of the non-gamified approach of the Road-Sensing application. These findings will be analyzed and put forward to document the outcomes in future works.

In the second part of this Chapter, we presented an original and open rewarding platform based on a voucher exchange system.

Thanks to a framework for implementing cross-application incentive strategies, the platform permits a real decoupling between MCS instruments and rewards provided by third-party stakeholders. The protocols presented and the vouchers' format allow to maintain user anonymity throughout the process. Stakeholders are free to filter the vouchers they request for providing services or goods (for aim, location, and time) and are therefore enabled to implement social policies by fine-tuning incentives and filtering criteria.

The platform described is currently under development and will be adopted within the *CROWD4ROADS* project already mentioned in this Chapter.

Several pilots will be deployed, making use of *SmartRoadSense* (in the role of the *instrument*) and the parking systems of selected municipalities that have shown interest in participatory road monitoring (which will act as *merchants* and provide incentives, for instance in the form of free parking for contributors).

The platform will also be made available to the Collective Awareness Platforms for Sustainability and Social Innovation (CAPSSI) Community[17], in order to be thoroughly evaluated and exploited at large.

[17] CAPSSI: `http://capssi.eu`

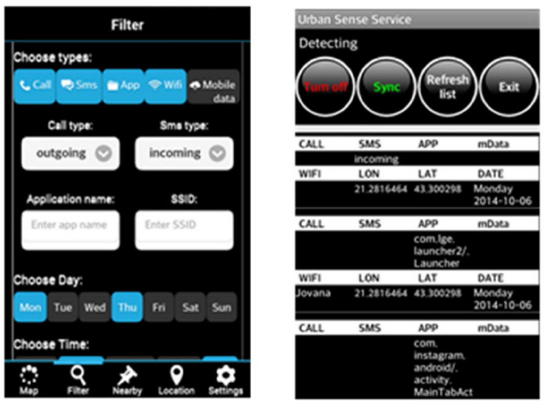
Chapter 4

User Interface

Most of the times, the interaction between users and a mobile crowdsensing application happens through a user interface (*UI*). Even if some MCS solutions have no interface on the client side and they only process collected data and send them to the server, the majority of MCS platforms provide volunteers and end-users with various kind of interfaces. The sensing task and the way end-users consume processed data determine which type of user interface the application may provide. For instance, volunteers may be enabled in their sensing activities by a mobile application, whose main purpose is to let users control the sensing operation and to relay all collected data back to the server. Along with these basic functionalities, mobile clients usually allow participants to select or join the sensing task, to configure sensing parameters associated with the sensing tool (e.g., frequency of sensing, precision of sensing, contexts in which the sensing should be activated or deactivated, conditions of data transmission, ecc.), to receive feedbacks on their activity, and to manage the rewards the user collects. As specified before (see Section 1.2.2), users are not just exploited to collect data, but many MCS applications let them create sensing tasks or consume final aggregated data produced by the crowdsensing process. It is in the interest of a successful MCS to provide users with a suitable UI for all the aforementioned activities.

Unfortunately, most of the MCS platforms used in real-world application still struggle to provide volunteers with a usable and easy-to-understand UI. As shown in Figure 4.1, the simple act of configuring the sensing tool parameters can become cumbersome when such settings are numerous and very specific. This issue can be ascribed to the inherent complexity of the task or to the lack of expertise among the research team components (who usually develop this kind of applications). Nevertheless, the MCS paradigm is based on participants and, commonly, applications aim to engage as many volunteers as possible in order to maximize the coverage of the sensed area or to improve the data redundancy (and therefore the aggregated data quality). As a consequence, the ability to attract and recruit users in joining the crowdsensing task is fundamental: having a well-designed user interaction is paramount to lowering the barriers preventing users from joining the sensing endeavor [193]. To make an MCS application as inclusive as possible the user interface of its clients should be easy to understand, but at the same time, it should be powerful enough not to hinder or obstacle the sensing process itself. In this regard, UIs primary focus should be on finding the perfect balance between providing users with a complete but possibly verbose or cluttered interface and offering a smooth but less powerful instrument to participate in the sensing process.

In the heterogenous MCS ecosystem, many applications are just prototypes (e.g., Anonymity [198], Anonysense [169], and Medusa [151]), simulators (e.g., MCSGame [192], Matador [24], and Context [133]), or demonstrators (e.g., Ear-Phone [152] and GPS-Less [167]). Among those platforms that are real-world implementations of a full-fledge distributed sensing system many have been implemented on top of dedicated frameworks (e.g., Sparse [182], MoreWithLess [194], SmartRoad [80], and MCSaaS [125]) or middlewares (e.g., MDPPS [161], Sahyog [7], and Neighbor [77]). Few contributions proposed MCS platforms designed and implemented from scratch (e.g., SmartRoad-Sense [2], MCS-Foster [22], and Ecosystem [81]). Focusing on the UI aspect, virtually all of the proposed applications have a server-side interface based on web technologies (with few exceptions [152]). Users can manage their sensing task, do queries and access aggregated data via standard websites or web services. As regards the client-side, almost all the proposed systems only provide participants with a graphical UI designed for the Android mobile operative system. Only a few platforms allow users to participate in sensing task and control their contributions using other operative systems. In particular, only PRISM [44] offers a Windows Mobile application clients, while LineKing [17], TYT [149], and CrowITS [3] sports both an Android and a iOS mobile clients. To the best of our knowledge, no real-world application provides a *web application* interface, even if this kind of mobile clients could allow users from all of the three major mobile operative systems (Android, iOS, and Windows Mobile) to join the sensing process. Web application frameworks usually provide limited control on smartphone embedded sensors. This is probably the reason behind the absence of web apps in the MCS scenario. Providing a client interface in the form of a mobile application seems to be the method of choice for almost any MCS platform. Nonetheless, mobile applications, especially those designed and implemented from scratch, entail serious drawbacks. Mobile applications are hard to implement and — as already seen in Chapter 1 — are rarely installed and kept by users [35]. Moreover, local storage on mobile operating systems is known to be insecure [144]. It is clear that the sensing tasks should not be a burden but, as also outlined by Zamora et al. [201], there are still many issues concerning the client side of MCS which still need to be properly resolved.



Filter

Choose types: Call, Sms, App, Wifi, Mobile data

Call type:	Sms type:
outgoing	incoming

Application name:	SSID:
Enter app name	Enter SSID

Choose Day: Mon Tue Wed [✓ Thu] Fri Sat Sun

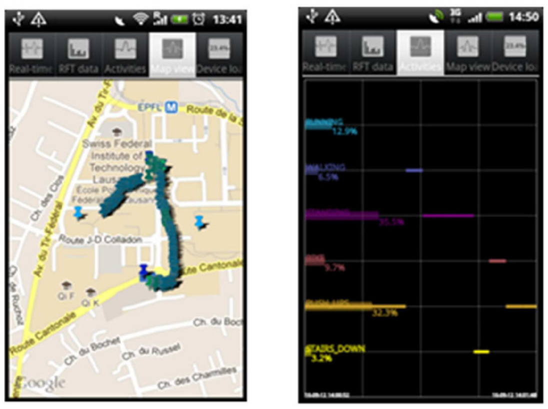
Choose Time:

Urban Sense Service

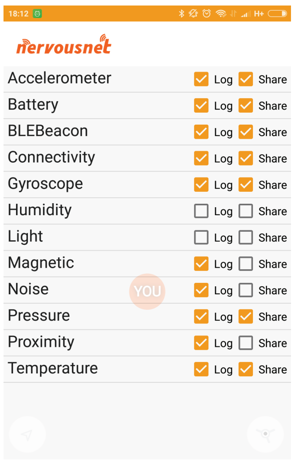
Detecting

CALL	SMS	APP	mData
	incoming		
WIFI	LON	LAT	DATE
	21.2816464	43.300298	Monday 2014-10-06
CALL	SMS	APP	mData
		com.lge.launcher2/.Launcher	
WIFI	LON	LAT	DATE
Jovana	21.2816464	43.300298	Monday 2014-10-06
CALL	SMS	APP	mData
		com.instagram.android/.activity.MainTabAct	

(a) UrbanSense [113] data collection and filtering configurations



(b) ExposureSense [148] map and timeline activity views



nervousnet

	Log	Share
Accelerometer	✓	✓
Battery	✓	✓
BLEBeacon	✓	✓
Connectivity	✓	✓
Gyroscope	✓	✓
Humidity	☐	☐
Light	☐	☐
Magnetic	✓	☐
Noise	✓	☐
Pressure	✓	✓
Proximity	✓	☐
Temperature	✓	✓

(c) NervousNet [147] lets users explicitly set logging and sharing configuration

Figure 4.1: To date, most of the MCS platforms used in real-world applications still struggle to provide a usable UI

In this Chapter, we will propose an original solution to the MCS UI issue. Our proposal is especially suitable for those MCS applications which require an explicit sensing style [72] and a significant commitment by the volunteer, that is to say where empowering users with an effective tool to contribute to the sensing task is crucial. We propose an original approach to tackle the aforementioned UI issues in MCS. Our approach is based on the emerging paradigm of conversational interfaces, and on the *chatbot* phenomenon in particular.

In the following Section, we give an overview of the chatbot phenomenon detailing its peculiar characteristics and its advantages (and disadvantages). In Section 4.2 a case of study based on a rather simple but representative MCS application is presented. In Section 4.3 we will discuss the approach and draw our conclusions.

4.1 The rise of bots

The reach of third-party applications for mobile operating systems — usually marketed as "apps" — has grown dramatically over the last years. A decade after the launch of the *App Store* for iPhone in 2008, the number of apps available to smartphone users now reaches into the order of millions. Their reach, impact, and economic significance cannot be ignored.

However, recent statistics have shown that users frequently make use of a limited set of popular apps. Moreover, most smartphone owners are accustomed to installing nearly zero new apps on their devices on a monthly basis [35]. For app developers it has become more and more difficult to make their products visible in an already crowded application store, where competition is harsher than ever.

Instant messaging (IM) and social network applications consistently take up the top spots among the most used applications. IM apps in particular have shown tremendous growth over the last years, recently taking over the lead in terms of number of users, growth, and user engagement [190]. Messaging platforms are getting immense attention and — as their success depends on the network effect of their users — ferociously compete for market share.

Starting in 2014, many messaging systems have introduced support for so-called "bots": enhanced conversational agents that can chat with users, right inside the messaging app itself. These bots live inside a very familiar place: in a conversation thread, right next to private conversations with friends and relatives, which is increasingly the most used feature of a user's smartphone. Most users in fact use messaging apps several times a day and have a well-rounded understanding of their interface and manner of working.

Instead of trying to attract people to new apps, bots provide an incredibly convenient way for services and developers to engage with users where they already are, making use of the existing conversational paradigm. Even if the conversation follows the familiar and recognizable conventions of a messaging system, the exchange does not need to be text-based. Thanks to the richness of the development frameworks made available by many messaging platforms, bots can also exchange information using complex messages and UI primitives, which allow the conversation to be more efficient.

Bots are growing fast in number and many IM platforms have started offering *bot stores*, just like mobile OS platforms do for apps. In fact, the robustness of the messaging ecosystem — encompassing its existing user base and distribution channels — make bots the perfect bridgehead to transform instant messaging systems into software delivery platforms. This could be a lucrative endeavor for messaging apps, bringing them into competition with the mobile platforms on which they operate.

However, to the best of our knowledge, a thorough and detailed appraisal of this phenomena is still missing.

In this Section, we examine this trend with a history of existing chatbots and a survey of popular IM platforms. We then propose the definition of "Botplications": conversational agents designed to follow a set of guidelines that can serve as functional replacements of apps. The distinguishing characteristics of such agents are identified, including differences and advantages over traditional applications. We then speculate about the future trends of these technologies and conversational interfaces.

4.1.1 History of conversational interfaces

The idea of building a computer — or better, a program — able to talk with humans, giving the illusion of a true human-to-human interaction, can be dated back to the '50s, when Alan Turing proposed his seminal "Imitation Game" [177]. Better known as the *Turing test*, it aims to determine if a machine can give the impression to other humans of being human itself. Today the Turing test is still used as a test for evaluating to which extent a program, a *bot*, is human-like: the Loebner Prize is annually assigned to the best computer system pretending to be human.

4.1.1.A Chatterbots

One of the first examples of such a program was *ELIZA*, created by Weizenbaum in 1966 [186]. ELIZA did simply answer all the user's utterances with other vague questions, giving a rough impression of a Rogerian psychotherapist. Weizenbaum's first example of a so-called "chatterbot", and its crude attempts at fooling users into believing they were interacting with another human, laid the foundations for chatterbots and bots in the following 50 years, until today.

Another example of a popular and more recent chatbot is *ALICE* (Artificial Linguistic Internet Computer Entity), which won the Loebner prize in 2000, 2001, and 2004. It was developed by Wallace in 1995, inspired by ELIZA. ALICE relies on a simple pattern-matching algorithm: it uses pattern templates to represent input and output and makes use of recursive techniques to apply different rules [180]. The underlying intelligence is based on the Artificial Intelligence Markup Language (AIML), which makes it possible for developers to define the building blocks of the bot's knowledge [179].

4.1.1.B Personification

From the initial attempts with ELIZA and ALICE, to the ambition of developing "intelligent" agents able to fully converse with humans using natural language, bots have been increasingly employed in many fields with a discrete success, yet with limited spread. They have been applied in e-commerce applications [94], like *Nicole*, a virtual assistant with customer service tasks, or *Anna* by IKEA; in education, like CHARLIE [126], a bot that lets students communicate with the online learning platform INES, or TQ-bot [55], dedicated to student tutoring and evaluation; information retrieval [160, 165], and e-government services [117]. Most of those bots are based on AIML and ALICE [162, 166].

Along with chatterbots, supported by the evolution of Automatic Speech Recognition (ASR) systems in the '80s, Spoken Dialog Systems (SDS) started drawing the attention of academics and the industry [123]. The conversation did move from a textual to a spoken channel, as a presumably easier and more natural interface for humans. ATIS [74] was a telephone-based flight reservation system, funded by DARPA, developed in 1990, in the USA; in the same years, SUNDIAL was developed in Europe [142], with the same aim of giving information about flights via telephone. The latter system was able to understand between 1.000 and 2.000 words in four different languages. Mercury was a similar, more recent, undertaking in this field [163]. These systems addressed several difficult technological issues, including speech recognition, understanding the user's requests, and giving the system the means of offering satisfactory answers.

A further step toward the "personification" of intelligent systems has been achieved with the development of Embodied Conversational Agents (ECA) [123], in the late '90s. Animated characters, with human features, able to simulate emotions with facial expressions and gestures, have been employed in different fields to interact with humans. They seem to be perceived as more trustworthy and agreeable, and are thus employed with the hope of being more readily accepted in everyday life applications than simple textual or spoken interfaces. Cassell et al. developed REA (Real Estate Agent) [25], a humanoid expert in real estates, that interacted with users and could sense them by means of cameras. Kopp et al. give another example of an ECA that has been used as a museum guide, in order to engage visitors and to interactively chat with them. In a real world study it was observed that users were inclined to use human-like communication strategies and that they perceived the agent as being sociable [98].

4.1.1.C Instant messaging

After the success of browser chats at the beginning of the '90s, like IRC (Internet Relay Chat), the end of that decade has seen the spread of instant messaging services, such as AOL Instant Messaging (known as AIM), Yahoo! Messenger, and MSN Messenger. Many of these platforms allowed users to add bots to their contact lists as if they were real people: they could then exchange simple text messages and receive different kind of information [94]. A famous example of such a bot was *SmarterChild*, that could converse about breaking news and the weather.

4.1.1.D Virtual Private Assistants

Only few years ago, "smart assistants" have started drawing the attention of the greater public. According to McTear et al., different reasons have influenced this new success of conversational interfaces [123]: the progresses in artificial intelligence assistive technologies, like speech and image recognition; the emergence of the semantic web; the increasing availability of connectivity and improvements in device hardware; and the renewed interest of big technology players. Apple's *Siri* (generally considered to be the the first public voice-enabled Virtual Private Assistant), Microsoft's *Cortana*, *Google Now* and *Google Assistant*, Amazon *Alexa*, and Samsung *S Voice* are the main actors of the last five years in the field of conversational interfaces.

Most notably, the reasons of the growing success of VPAs could also be explained by the many differences in respect to older personal assistants. First of all, smartphones have become pervasive in everyday life and are perceived as more personal than any other device. The same goes with the assistants, which are always present for the user. Modern assistants are perceived as more flexible: they are not limited to a very narrow task, but are able to interact with a plethora of applications, internal and external to the device [10]. The interaction is more "human-like", using simple, yet impressive tricks: the effort made to provide direct answers, in integrated and often spoken dialogs; improvements to the inference of the user's intents and the correction of ambiguities; better interpretations of the input's semantics; the capability of answering sassy questions, giving the illusion of a synthetic and likable personality. All these factors have probably made the luck of modern VPAs and the failure of older ones, like for instance *Wildfire*, a VPA of the mid '90s, multi-modal and phone based, but with poor human-like interaction capabilities [87].

Even though basic services of the aforementioned personal assistants are somewhat similar (including web search capabilities, event planning, voice calls and messaging, music playback, personalized notifications, weather information retrieval, etc.), some peculiar aspects can differentiate them. Siri and Cortana are the more similar ones, acting as personal assistants with a well defined "personality" that transpires in many of their answers. Google Now is also similar to these first two, but lacks a well defined personality and pulls information directly from the user's online Google Account, possibly raising some privacy questions. Samsung S Voice is probably the more "classical" one among the others and may not be perceived as personal like the other ones. Lastly, Amazon Alexa is going in a somewhat different direction: developed for Amazon Echo, a smart speaker, it brings the Internet of Things (IoT) closer to everyday life and it can be integrated with different other devices and services, especially in regard to home automation and entertainment.

It is also worth mentioning the IBM statistical responding agent called "Watson" [53], which stepped into the limelight in 2010 for participating in the "Jeopardy" television program. In that occasion Watson beat both his two human competitors only relying on an off-line database of unstructured contents. In more recent years, IBM further developed Watson, turning it into a powerful Artificial Intelligence (AI) assistant employed in health-care and custom-care scenarios. Watson can also be exploited as a cloud-based cognitive services of composable AI building blocks [195]. Developers can train the Watson AI to answer to answer questions posed in plain natural language about particular intent, and use it to build applications like chatbots.

4.1.1.E The rise of a new bot-based paradigm

In the last couple of years a new approach to conversational interfaces has been reclaiming prominence: an apparent return to the classic *chatterbot* texting interfaces has been observed, with the main difference that these new bots have gained additional capabilities and now "live" in the cloud.

Starting in 2014, many online messaging systems (like Kik, Telegram, and WeChat) have opened up to third-party developers, offering the means to building bots and programmatically exchanging messages with users through their platforms. Application Programming Interfaces (APIs) for bots expose high level services (messaging, payments, bot directory, authentication, etc.) and UI elements (buttons, locations, images, etc.) giving developers the possibility of implementing innovative services through a conversational user experience.

Services offered by these new bots are of a higher level than the ones offered by their predecessors. Bots often feature access to other services that have utility in everyday life, such as ordering food, managing an e-commerce purchase, booking restaurants, ordering a cab, and so on. Examples include health care bots such as *Nombot* [67], a bot helping users to track their daily food consumption on Telegram; educational bots, like the one by Chaniago et al. that let parents track their children's presence at school [26]; educational help, like *MOOCBuddy* that proposes MOOC courses over Messenger [85].

It is also of note that the possibility of building more helpful and practical bots is in good part due to the increased availability of service APIs open to third-parties and the rise of the "platform" business model [31]. The recent decade has seen a growing attention to *open data* as well, which is increasingly made available by governments or administrative entities, also providing a useful foundation for many bots.

Since data and services are more and more accessible through programmatic interfaces, and given that bots often offer a simpler development platform than apps in terms of development and maintenance efforts, the task of offering access to such services through a conversation interface is very approachable.

Many of the major messaging platforms[1] lately have started offering *bot directories*: repositories similar to app stores, listing the available bots that can be accessed through the platform.

4.1.1.F Beyond "Chatbots"

In light of the recent resurgence in popularity of chatbots, we argue for a new and more significant taxonomy of autonomous conversational agents.

Firstly, the very nature of the term "chatbot" sets certain expectations about the agent's interactions: users may assume that a chatbot will be able to "chat" with them using natural language, but only a subset of autonomous agents are designed to simulate natural conversations. Agents that are not will inevitably fall short in their understanding of humans, who have been generally shown to tend to impatience when interacting with software agents [92, 10].

In fact, the "chat" verb trivializes the potential of bots, leading back to chatterbots as silly novelties with which to exchange messages on a command-line, mimicking human idiosyncrasies without any further purpose, just like with the aforementioned ELIZA chatterbot. Instead, as we discuss in the next section, bots have the potential to be a powerful and efficient software platform, only incidentally accessible through text messages.

[1]Kik: `https://dev.kik.com/`,
Facebook Messenger: `https://developers.facebook.com/docs/messenger-platform`,
Telegram: `https://core.telegram.org/bots`,
Skype: `https://www.skype.com/en/developer/`,
Line: `https://developers.line.me/messaging-api/overview`,
WeChat: `http://dev.wechat.com/wechatapi`,
Slack: `https://api.slack.com/bot-users`,
all accessed on October 2017

4.1.2 Botplications

Among the plethora of new and old conversational interfaces, we identify a category of conversational agents that, having been designed according to principles of simplicity and effectiveness, can serve as functional replacements of mobile applications.

We call them “Botplications”.

A Botplication is an agent that is endowed with a conversational interface accessible through a messaging platform, which provides access to data, services, or enables the user to perform a specific task.

In particular, a Botplication is generally characterized by the following distinguishing features.

4.1.2.A Thread as app

Botplications are stepping stones in the evolution from an *app-centric* mobile OS experience — where the whole user experience of the device is concentrated in mostly independent applications that serve as an enclave of unique services and functions — to a *thread-centric* experience. That is, a user experience where services and information are provided as streams of messages and notifications, presented using a coherent and consistent set of interface paradigms.

Messaging threads are independent conversations that enclose personal relationships. Exchanging messages is in fact the primary method for users of a mobile OS to keep up relations with other people, in a very natural and intimate way. These personal conversations and relations can be naturally extended to services and businesses.

Each one of these threads is an entity that can send updates and notifications to the user, even in multiple parallel conversations, while taking advantage of the built-in facilities of the messaging system.

Among these facilities, for instance, the ability of users to retain total control over each single thread: they can chose to reply, mute the conversation, or even to permanently delete the thread. Also, threads have the capacity of keeping track of read/unread status and message drafts on multiple devices or platforms. The user has the ability to search for a message across all open threads, instead of having to remember in which particular app or service the information is hidden. Incoming messages on any thread are notified to the user in a familiar way.

Most modern messaging apps are in fact presented as a threaded “inbox”, automatically grouping messages from the same sender and surfacing recently updated conversations that may be of interest. Instead of having new information dispersed across a number of isolated apps, each with its own custom interaction modes, users can rely on the fact that frequently or recently used services are automatically promoted to a visible position.

A user's "inbox" acts as a replacement for the app-centric homescreen of a mobile OS, where the most recent threads serve the same purpose of a dynamic list of favorite apps.

Conversation threads make it easy to provide integrated tools and services that make it easy to accomplish regular tasks, but in a recognizable and familiar place: a personal relationship developed through the exchange of text messages.

4.1.2.B History awareness

Just like mobile apps, Botplications are designed to solve a specific and circumscribed issue. However, unlike most apps, Botplications inherently keep an exhaustive chronological log of past interactions with the user in their thread.

This ingrained feature of a thread of messages allows the user to explore past information in a familiar way, by scrolling through a timeline or by using built-in temporal search. Users can approach the service with more confidence, since past state appears frozen in previous messages and data does not unexpectedly vanish.

History can only change with familiar and explicit user actions, deep-seated in muscle memory for the majority of users used to receive and send text messages. New messages are appended to the history in a predictable and well-known way, while past interactions can be deleted through explicit action of the user.

History also serves as guidance to users, since past commands (and the results they generated) can be easily used as a template for future requests.

Also, past history provides the context in which all future interactions can be evaluated. Information collected by a Botplication *should* be maintained and used in order to streamline requests, skipping questions and automatically disambiguating between different choices if possible. A conversation is a natural and effective way to collect personal information, interests, purpose, and preferences of the user, all of which can be employed in order to improve the quality and accuracy of the service.

4.1.3 Enhanced UI

Despite the fact that Botplications derive from chatterbot-like conversational interfaces, their UI does not *have* to consist of mere plain text messages.

Modern messaging platforms support a variety of messages, including pictures, "stickers" (preset or custom images that convey emotion), videos, and audio. Most platforms also allow the transmission of packaged data, as in the case of geo-locations or contact information, in addition to generic data files. While these platforms of course do not offer the graphical capabilities of apps, it is important to make use of the features available and to exploit the conventions of the messaging platform. For instance, instead of printing out a raw URL, some platforms may display links as nicely formatted cards with a preview.

Moreover, on many messaging platforms even plain text messages can be enriched with a limited subset of rich text formatting. Bold, italic, and embedded links are the most common formatting options, available through markup languages such as a subset of HTML or Markdown. Unicode-encoded *emoji* characters can also be used to efficiently transmit additional meaning or to convey emotion.

Typing should be reduced to a minimum: ideally it should be limited to very short and concise commands and replies, of few characters. A fitting comparison can be done with UNIX commands, which are an example in terseness, as they were designed to efficiently work over teletypewriters — the primeval example of text-based interfaces — and reward users with powerful functionalities through very little typing.

Even if it can be argued that younger generations are getting more accustomed to typing on touch-based soft keyboards, the practice of shortening common words or using abbreviations in everyday messaging by itself demonstrates why full-fledged and verbose conversations should be avoided if possible. Interactions with Botplications should follow linear conversation routes and avoid complicated branching or multiple complex dialogs: anything that cannot be accessed through a couple of taps on the screen will become tiresome to most users. In a user study by Azenkot et al. the average text entry for all participants on a popular mobile OS keyboard was of 41.01 Words per Minute (WPM) [6]. Also, the theoretical maximum typing rate predicted by MacKenzie et al. was of 43.2 WPM for expert users on QWERTY keyboards [116], which is of course far lower to what can be achieved by expert typists on a physical keyboard [68]. Text interactions for mobile users should therefore be kept short and precise.

Many messaging platforms also feature advanced structured message forms, which can further enhance the flow of conversation. Structured messages may, for instance, contain buttons for preset "canned" replies, show different alternatives in a rich representation, or show a list of available commands. Examples of such messages are shown in the next section. The advantages of structured messages are manifold: (1) They constrain the conversation into a limited number of expected outcomes, reducing the possibility of users feeling trapped in a dead end where they have to "guess" their way out. (2) They push the user to use the service, suggesting how the conversation can continue. They also reduce the need for the users to "explore" the interface, making it easier to learn and use. (3) Buttons and quick replies reduce interactions to a single tap instead of requiring complex typing. (4) The service can be implemented more easily.

Botplications try to fill the gap between plain conversational interfaces — which are inherently inefficient to use and offer little way in terms of user experience customization

— and the world of the full graphical interface.

4.1.3.A Limited use of Natural Language Processing (NLP)

Given the aforementioned rising interest in Virtual Private Assistants (VPAs), voice appears to be a natural fit for conversational interfaces. However, to the best of our knowledge, existing bots on messaging platforms avoid voice processing and choose digital text as the most direct and unambiguous form of communication, possibly adopting NLP systems in order to extract commands and intent from the user's messages. Even if natural language understanding is getting progressively more advanced [10], in many scenarios complex dialogs break down because of simple misunderstandings, because of unexpected turns of phrase (human language shows incredible variation and rarely follows a script), or because the user's context cannot be fully elicited from the conversation [123, 9].

Botplications should not attempt to provide the complex experience of a full VPA. Instead, in keeping true to the principles of simplicity and effectiveness, Botplications should almost completely avoid natural language where possible. At the scale of a single bot, going after AI is mostly excessive and counterproductive, when the same results can be obtained with simple text commands using a limited structured language.

Botplications should in fact not pretend to be human: except in cases where this is desirable (e.g., for customer support or as a barrier before an actual conversations with a human operator is activated), it should be clear to users that they are talking to a machine. Even if artificial delays or "is typing" indicators can be used in order to make the conversation more familiar to the user, faking human responses risks to increase the perceived distance between service and user instead of decreasing it. Texting a computer that doesn't understand what the user is saying can be a frustrating experience, in particular when the computer hides its failures inside a dialog that is artificially kept "natural" and "human-like". This hides failure points in the conversation and makes the user feel less in control of the interaction.

However, this does not imply that bots shouldn't show a personality or take advantage of humor and emotional responses to provide a charming and likable interaction with users.

Botplications should rely on the limited — but accurate — interaction tools the messaging platform makes available, while NLP frameworks can optionally be employed to accommodate unforeseen user requests. AI and deep human-like dialogs are red herrings in the current development of conversational interfaces: Botplications should be about accessing services efficiently, a command-line-like interface to cloud-based APIs, not talkers.

Table 4.1: Features of major messaging platforms that support bots (as of October 2017).

Platform	MAU†	Groups	Mentions	Message types	Buttons	Carousel	Quick reply	Payment
Messenger	800 M			Picture, video, file, voice.	✓	✓	✓	✓
	Persistent menus, several message templates (Airline trip, Buy, Receipt, Web link, etc.).							
WeChat	700 M		✓	Picture, video, sticker, voice, location.			✓‡	✓
	Deep-links through QR codes, Rich media and Music messages. Integrated web views§.							
Skype	300 M			Picture, video.	✓	✓		
	Several message templates (Hero image, Thumbnail, Receipt, Sign-in, etc.), phone call support.							
Line	220 M	✓		Picture, video, sticker, voice, location.	✓	✓		
	Imagemap message template (picture with multiple hot-spots).							
Telegram	100 M	✓	✓	Picture, video, sticker, file, voice, location.			✓	
	Persistent commands. Deep-links to conversations.							
Kik	80 M	✓	✓	Picture, video, sticker, voice.			✓	
	Kik code identifiers, browser integration via Javascript.							
Slack	$\gtrapprox$ 3 M	✓	✓	File.	✓			

† Monthly Active Users, based on most recent quarterly report published at the end of 2016. Numbers for Slack are an approximation based on known Daily Active Users.
‡ In the form of custom defined menus shown in the conversation UI.
§ WeUI, HTML/Javascript library that provides web-based UI elements coherent with the WeChat app: `https://weui.io/0.4.x/`.

4.1.3.B Message self-consistency

Each single message sent by a Botplication should contain the full context of the conversation and should embody what a single UI screen represents for mobile apps. Users should not have to browse through their conversation history in order to figure out what they are attempting to do and what the service is expecting. Each message has an atomic meaning and stands on its own.

The scope of each message must be clear, its intent must be explicit, and what action must be taken by the user — if any — must be explicit and unequivocal. Indeed, a message delivered by a Botplication should be conceptually seen as a *micro application*, while the conversation is a timeline of past application screens. As mentioned before, structured forms of messages, which include buttons or commands, should be used where appropriate.

Some messaging platforms have, in fact, the ability to alter messages after they have been delivered. In that case, messages can be changed based on the availability of new data or other changes in state, giving the impression of a living view on the service.

4.1.3.C Guided conversation

An important part of an application's user experience is based on user guidance and, likewise, the same care should be applied when designing conversational interfaces through text messages. In fact, because of the free-form nature of the medium, it is easy for bot users to get lost and not to be certain of what commands or what exact syntax is required to perform the desired action.

A successful Botplication guides the user through a task in order to avoid this impasse. The service proactively suggests actions that are likely to follow up after the current interaction, offers alternative choices when needed, and generally offers a framework in which user interactions feel reliable. This can be achieved using the same UI enhancements mentioned before, that is through the use of buttons, formatted messages, or built-in menus that offer interface guard rails to the conversation.

Also, notice that when starting the first interaction with a bot, many messaging platforms offer a way to show a welcome message to the user. The design of the onboarding experience must take into account the initial user guidance and ensure that all functionalities are readily available.

4.1.4 Technological survey

In this section the most popular messaging platforms that support bots through their APIs will be taken into exam, describing distinguishing features of each one.

All of the platforms mentioned allow third-party developers to register an identity for a virtual agent and to programmatically receive messages from any user of the platform, either by accessing an API end-point (*pull* mode) or by being called back by the platform itself using a "web-hook" (*push* mode). All platforms, in any mode, make use of the HTTP protocol.

The *Kik* messenger service launched in 2010 and introduced bots in 2014. On April 2016 the platform launched a "botshop", which includes a directory of available service bots. *WeChat* is a popular IM system in China, first launched in 2011, that always included more features than simple messaging. The platform includes support for "Official accounts", which can provide services to users. It includes a payment system that allows the exchange of monetary gifts, known as "red envelopes". *Telegram* published its bot support API on July 2015, further expanding it with the 2.0 API in April 2016. *Line* also launched its first messenger API in 2015, while Facebook's *Messenger* and *Skype* joined the game only later, including support for bots in April 2016. The former also added support for payments in September 2016.

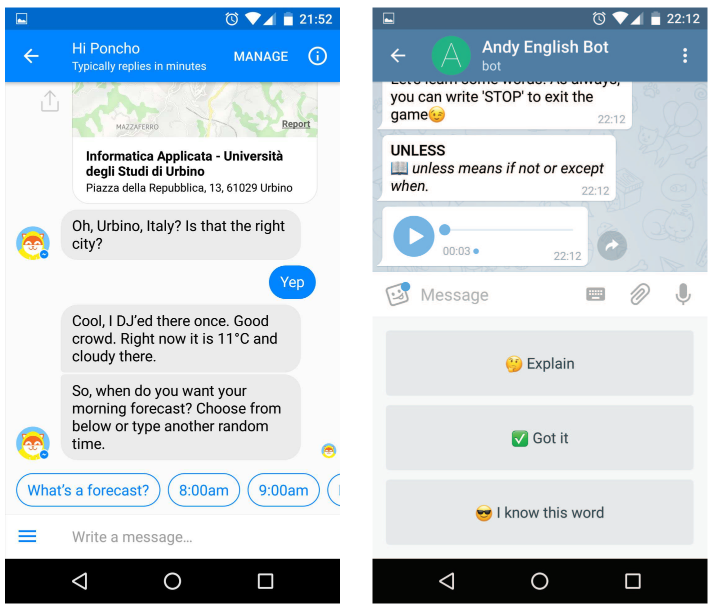
Given the rising interest in bots, most platforms are rapidly iterating and evolving their APIs and services to third-party developers. This survey is updated to March 2017.

In Table 4.1 the platforms considered in this study are listed, firstly reporting the amount of *Monthly Active Users* (MAU), which gives an approximation of the relative popularity of each system. Furthermore, the Table shows whether the platform supports *group messaging* (i.e., if one or more bots can be added to a group conversation between real users) and *mentions* (i.e., if bots can be "called into" a conversation using a special combination of characters). Both of these features allow bots to be used in order to perform specific tasks for a group of users instead of only one. The Table also reports the different *message types* supported by the platform (including pictures, voice, video, stickers, and structured messages, discussed in the next Section) and other significant features.

4.1.4.A Interface features

Messaging platforms distinguish themselves not only for their underlying technical aspects, but also because of the different user interface elements they offer to bot developers and, ultimately, to the end-users. While sending and receiving text and basic multimedia messages is a common feature, more structured messages are available only on some platforms and often differ in key aspects.

Many platforms offer a way to suggest canned replies, which can be sent with a single tap. For instance, on Messenger bots may display a selection of *quick replies* that appear on the bottom of the conversation and remain valid for one interaction, as shown in Figure 4.2a. On Telegram and Kik, bots have the ability to replace the keyboard with



Informatica Applicata - Università degli Studi di Urbino
Piazza della Repubblica, 13, 61029 Urbino

Oh, Urbino, Italy? Is that the right city?

Yep

Cool, I DJ'ed there once. Good crowd. Right now it is 11°C and cloudy there.

So, when do you want your morning forecast? Choose from below or type another random time.

you can write 'STOP' to exit the game😉 22:12

UNLESS
📖 *unless means if not or except when.* 22:12

00:03 22:12

(a) (b)

Figure 4.2: Features that allow users to pick suggested replies. (a) *Quick replies* on Messenger. (b) *Button keyboard* on Telegram.

suggested responses, as shown in Figure 4.2b.

While these preset replies can change in the course of the conversation, other UI features can immutably be added to the chat. In Figure 4.3a a list of *commands* are shown. On Telegram, like on Slack, commands are characterized by starting with a '/' character and are always available to the user to perform tasks in a command line-like experience. A similar system, but with a hierarchical structure, is shown in Figure 4.3b: WeChat allows developers to add a fixed menu to the chat interface, showing a maximum of 3 first-level options and several second-level ones. The same system has been also adopted by Messenger in March 2017. Commands and menus alike give direct access to a bot's main features.

Other UI features are not bound to the overall conversation, but are instead tied to a specific message. For instance, *carousel* messages shown in Figure 4.4a make it possible to include multiple rich cards, provided with an image, a description, and a button, and make them horizontally scrollable to the user. In Figure 4.4b a message with *embedded buttons* is shown: in this case the actions provided to the user are not persistent throughout the conversation, but are limited to one single interaction. Both message formats allow bot developers to show available alternatives to the user, providing a well defined path for the conversation.

Several other message formats are available to specific platforms, among which mes-

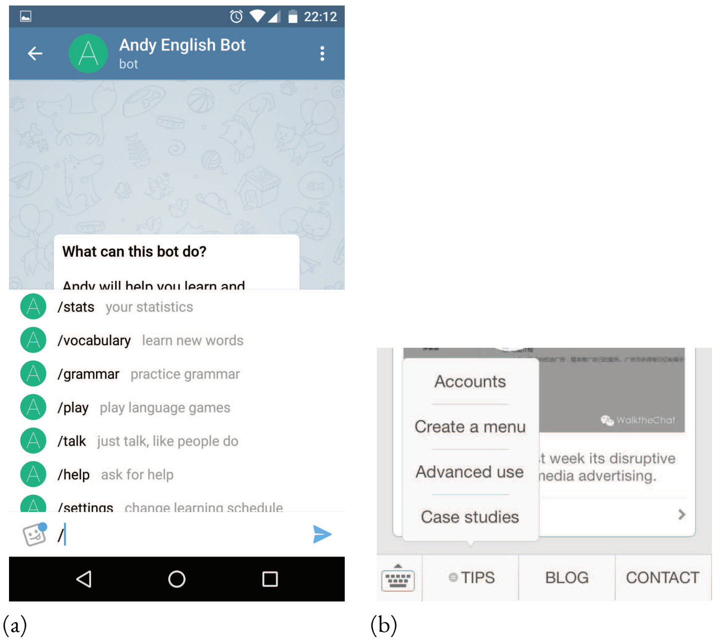


Figure 4.3: Structured commands available during the whole conversation. (a) *Commands* on Telegram. (b) *Custom defined menu* (with an open second-level menu) on WeChat.

sages including a "Buy" link for shopping-oriented bots, messages with music support in WeChat, and flight travel itinerary messages on Messenger.

In terms of making bots interoperate with external systems, it is noteworthy that several messaging platforms have included support for QR Codes or variations thereof. On WeChat, code scanning can be used internally to the system in order to send payments of a preset value to a given contact, which makes the platform compelling for sellers of physical goods as well. On the other hand, Skype, Telegram, and Messenger allow web sites and apps to launch conversations through the use of a specially marked URL. Such URLs include a custom data payload that is sent to the bot in order to track user entry points or to invoke specific actions. This mechanism — also known as "deep linking" — can be used to embed bots in innovative and complex workflows [96].

4.1.4.B Advantages of Bots for users

Instant availability Bots do not need to be downloaded and installed: they are immediately up and running as soon as a conversation is started inside the messaging app. Device storage capacity, installation, and complex configuration steps are not required. This makes bots fast and lightweight, if compared to traditional mobile apps.

"Instant Apps", a technology for Android still in development, will bring a similar experience to apps as well, making them instantly available to users without installations[2].

Gentle learning curve Texting has been used since the dawn of mobile phones and this makes it quite natural for users of any skill level. Provided that the bot provides adequate guidance through its features, learning how to interact with it is never an outlandish experience.

Since all bots on the same messaging platform rely on the same UI building-blocks (such as buttons, carousels, quick-replies, and so forth, down to the basic text messages), all bots share a common UI vocabulary. Knowledge in the use of one bot can easily be transfered to the usage of other bots.

Notifications Many mobile applications implement their own notification system, used as a mean to re-engage inactive users. Instant messaging applications however already include an efficient and functional push notification system, which is available by default without any additional implementation effort.

According to statistics by Localytics[3], more than 50% of users find push notifications annoying. However, receiving notifications through the messaging apps could be perceived as being less invasive.

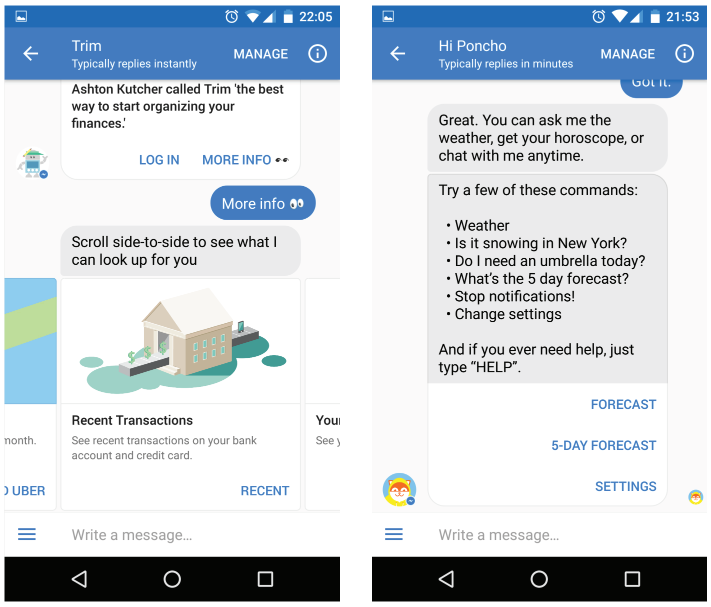
Social graph and contacts Users are already accustomed to voluntarily sharing and storing contact information in messaging apps. In fact, as soon as a bot conversation is initiated, the bot automatically receives the user's basic personal data. When participating in a group conversation with multiple users, bots also gain access to the contact information of all participants.

Moreover, all contents generated by an interaction with the bot exist in the form of messages, which can be directly forwarded and shared with friends. Thus contact to the bot can be spread through a user's social graph, without leaving the messaging app or installing new applications.

Platform independence Bots live inside instant messaging applications without having to worry about which mobile platform they are running on. This makes bots independent from the underlying host operating system: each bot is inherently available on all operating systems the messaging app supports, without any graphical or functional adjustment.

[2]Android Instant Apps: `http://developer.android.com/topic/instant-apps/`

[3]Localytics: `http:/info.localytics.com/blog/the-inside-view-how-consumers-really-feel-about-push-notifications` [last accessed 10 Oct 2017]


22:05
Trim
Typically replies instantly
MANAGE

Ashton Kutcher called Trim 'the best way to start organizing your finances.'

LOG IN MORE INFO 👀

More info 👀

Scroll side-to-side to see what I can look up for you

Recent Transactions
See recent transactions on your bank account and credit card.
RECENT

Write a message...

21:53
Hi Poncho
Typically replies in minutes
MANAGE

Got it.

Great. You can ask me the weather, get your horoscope, or chat with me anytime.

Try a few of these commands:

- Weather
- Is it snowing in New York?
- Do I need an umbrella today?
- What's the 5 day forecast?
- Stop notifications!
- Change settings

And if you ever need help, just type "HELP".

FORECAST
5-DAY FORECAST
SETTINGS

Write a message...

(a) (b)

Figure 4.4: Structured message templates in Messenger. (a) *Carousel* presenting multiple choices through horizontal scrolling. (b) *Message buttons* embedded in a message.

On the contrary, native mobile applications must be adapted or rewritten for each mobile operating system, often with great effort. Users sometimes perceive the same application as being different if they are using it on different operating systems, which can frustrate user engagement. Also, mobile applications must be frequently updated in order to keep up with upgrades to the host system and its features.

Authentication Usually, for each new application or service, users need to register and go through the steps of creating a new account. This is a known obstacle to user onboarding, who notably dislike long registration processes and often do not remember new account credentials.

In contrast, user authentication is not needed in bots. The hosting messaging platform already provides and guarantees the user's identity reliably. Users are uniquely identified by default and they do not have to create additional accounts and passwords in order to interact with the service.

Payment support Some of the analyzed platforms include payment services, integrated in the messaging system. Often users will have already connected their credit cards or bank accounts to such services, allowing bots to use the existing payment methods in a safe and reliable manner, for both user and developer. In contrast, for each brand new app requiring payment capabilities, users have to re-connect their payment accounts and have to trust the newly installed app.

Discoverability As seen previously, discoverability is a very real problem for mobile application on application stores: fierce competition and very small economic margins often make it very hard to compete and gain visibility. The same issues exist for bots. Even if many messaging platforms offer "bot stores" with a selection of available bots, emerging and gaining a solid user base remains one of the main development issues. However, bots have some advantages considering that they can be easily integrated in group conversations or shared just like any other contact.

Asynchronicity Exchanging instant messages is an asynchronous task: after sending a message, users do not have to wait for a reply. Conversation threads store the context, letting user free of leaving a conversation and going back to it later, picking up the dialog from the very last interaction. Multiple conversations can be easily carried forward in parallel.

Limited data requirements Downloading a new app can have a sizable impact on a limited mobile data plan, whereas starting to use a new bot only requires to initiate a new thread in an existing messaging app, with a negligible impact in terms of data traffic. The data size of exchanged messages in an instant messaging app is limited and usually more predictable. This makes bots very attractive for users with limited data plans or in countries with less developed technological infrastructure [130]. In fact, Facebook recently developed "Lite" versions of its applications, specifically targeted at emerging markets, where limited data plans are more common [191].

4.1.4.C Advantages of Bots for developers

Communication reliability Instant messaging applications are naturally designed for fast and efficient message delivery. They are capable of dealing with all kinds of network issues, automatically retrying transmissions and ensuring that the message is either delivered or that failure is handled adequately or correctly reported to the user. Security and privacy of transmitted messages are also guaranteed by many platforms.

On the other hand, mobile applications must implement most of these networking features independently, incurring in the likely risk of bugs or security issues. These issues are multiplied in case of multi-platform applications.

Fast iteration Since a bot's logic is implemented server-side and no code must be deployed onto user devices, deployment is effectively effortless. Similarly to what happens for web pages or web apps, bot updates are also almost immediately propagated to all users. This form of deployment also completely avoids issues with app stores, for instance due to rejections or delays during application review. This allows for fast and uncomplicated development iterations.

Lack of version fragmentation The immediate propagation of updates to all users also means that no user risks staying stuck with an old version of a mobile application: just like web apps, bots are always updated to the latest version.

Limited design efforts Bots heavily rely on the instant messaging application UI and usually have quite limited possibilities of graphical customization. This reduces the interface design time, limiting it to minor customizations of interface elements and focusing on a rich user experience using few simple building-blocks.

Development ease As seen in the previous Sections, several important and demanding services are already implemented by the messaging platform. Services such as user authentication or payments normally require major efforts and attention by mobile app developers. Bots however already include many of these services, which are ready to use through developer APIs.

Moreover, having a generally smaller API surface — which can be used through any development platform and language — developing a bot is generally cheaper and less time demanding than the development of a mobile application.

4.1.5 Discussion

As previously examined, the mobile device ecosystem has been characterized by the popularity of mobile apps and app stores as a software distribution platform in the last decade. Recent statistics have however shown that users tend to rely almost exclusively on a very restricted set of consolidated, preferred apps, instead of trying out new ones, leading the number of new downloaded apps nearly to zero in the short-medium period [35]. Moreover, a 2014 report has also shown the surprising overtake of online messaging platforms usage over social network apps [190].

Bots on instant messaging platforms have recently started gaining widespread interest. As discussed above, not only do bots rely on very popular messaging apps, they also eschew many of the difficulties of mobile app development and distribution. In fact, bots can increasingly be considered as a novel approach to the software distribution problem.

From this point of view, bots show many similarities with mobile web apps, as a distribution system competing with native mobile applications [27]:

1. they allow the development of multi-platform services using a commonly available platform (messaging apps or the web),

2. they are partly limited in terms of what features of the mobile device they have access to (this is particularly true for bots, which however have a different usage paradigm with different expectations from users),

3. the "look and feel" is different from a native app (however, while the user experience of mobile web apps may be different from what users are accustomed to, bots offer a very familiar interface that feels native to the messaging app at least),

4. both approaches favor developers in terms of development, distribution, and maintenance efforts, however bots also ensure a high ease of use to end-users.

These advantages notwithstanding, there are different drawbacks to bots that must be taken into consideration when evaluating their merits as a software platform.

In the first place, not every application is well suited for conversational interfaces. There are several tasks that are inherently more convenient to be tackled by means of a dedicated app, with access to local computational resources and data storage, instead of a remote bot. Other tasks simply work better with a rich visual interactions that goes beyond what a conversational interface can offer.

Even if bot development is very approachable by developers and offers many advanced features at essentially very little effort, a bot is tightly bound to its messaging platform, which may lead to design constraints. There are, very noticeably, inherent limits to the customization of UI elements, the look and feel, and finer details of the user experience.

Bots require an active Internet connection in order to work (in contrast to modern mobile web apps, which may provide an offline experience). This can create discontinuities in service availability and negatively affect the bot's perceived reliability, in case of lacking network coverage or limited data plans, especially in growing markets [130].

Despite the appearance of "bot directories", discoverability is still a crucial issue that must be tackled both by developers and bot platforms. Also, the risk of an overpopulation of bots that hinders the discovery of new services, like for app stores, is equally likely.

Many recent efforts in the field of bots have made use of natural language processing in order to create a generic conversational interface, supporting natural human-like interactions. There is however an inherent distinction between conversational virtual assistants (such as Amazon *Alexa* or Apple's *Siri*), which are well-suited to interactions through spoken natural language, and service-oriented bots, that currently benefit from structured input. While the first kind of bots may provide a more convenient, and in some cases instinctive, interface, the latter can focus on providing a more efficient and lean experience. Also, NLP capabilities are presently not always sufficient to fulfill natural-feeling conversations and they still impose a learning curve on users [10, 9].

Even though some of these issues can be addressed and mitigated in the near future, there are inherent limitations to what bots can offer, which limit how far they can serve as a replacement to native and web apps.

The relative novelty of bots must also be taken into account: the first roots for the spread of bots have been put down, but we are still in the very early days of this new era. A "killer application" for bot platforms, which goes "viral" and fully justifies the rising interest by clearly showing the full potential of bots, is yet to be seen.

In this Section the spread of the new generation of bots on messaging platforms has been discussed, describing how the concept of "bot" has changed from its very first stages to the present days.

A definition of *Botplications* has been given, as a conversational agent living on existing messaging platforms, that follows a set of principles of simplicity and purposefulness in providing access to services and data.

Features and limitations of the most prominent messaging platforms have been presented and compared. A technological survey, depicting in detail the peculiar available user interface mechanisms, and advantages of bots, for both users and developers, were discussed in detail.

In conclusion, even if bots will not substitute the whole mobile application ecosystem in the near future, close attention must be paid to the "rise of bots", as a new software platform for delivering services and data to users.

4.2 A MCS Botplication case study

In the previous Section, we thoroughly discussed the Botplications phenomenon. In this Section, we will cast the conversational interface paradigm in the MCS framework as an alternative method to provide participants with a simple and usable client interface. Our aim is to find an approach able to lower the barriers to adoption and to increase the user base of suitable MCS application.

To demonstrate how a Botplication can be used to effectively reach volunteers of an MCS platform we develop a basic MCS application whose mobile client component is a simple bot. *Da-qui-a-lì* is a Botplication we use as a toy example. It will allow us to empirically investigate and, then, discuss how convenient is the use of conversational interfaces as MCS UIs. In the following of this Section, we will overview the MCS platform we implemented and its aims. After that, the conversational interface will be described, and in the last part of this Section, we will try to analyze advantages, limitations, and drawbacks of adopting a bot-based client in designing an MCS platform.

4.2.1 Platform's Aims

Da-qui-a-lì is an elementary MCS application whose aim is to collect data about pedestrian accessibility within residential areas and produce a map of best and fastest paths throughout the city. "Da qui a lì" is the italian for *"from here to there"*, and the platform name reflects its working principle. Users are tasked with a simple sensing task: to provide an estimate of the duration and comfort of paths they cover within an urban area. Each participant can record as many journeys as he/she wants and the same track can be logged many times. Users provide both the starting and the ending location of their journey, its duration, and few other meta-information such as a score representing how they evaluate the accessibility of the covered path, whether they needed any external help in traveling, and which kind of physical impairment (if any) they suffered along the path. Collected data are processed offline in order to build a pedestrian accessibility map of the covered area[4].

As a proof of concept, the software architecture of Da-qui-a-lì directly mirrors the common MCS application architecture seen in Section 1.2.2. In this simple application, the application server is responsible for collecting data from participants as such information will be processed offline afterward. The mobile client is implemented as a Telegram bot that volunteers use to communicate data they gather during their journeys. The bot core logic is implemented as a web service and the telegram bot's backend refers to it whenever a user sends a message through the mobile messaging platform.

Both the application server and the bot core logic has been developed as PHP/MySQL applications. As already discussed in Section 4.1, Telegram is a cloud-based mobile and desktop messaging app with a focus on security and speed [173]. In this work, we choose to implement our application on top of the Telegram instant messaging platform since its adoption rate is growing fast [19] and because both its communication protocols and its Bot's APIs are extensively documented and easy to be used [174].

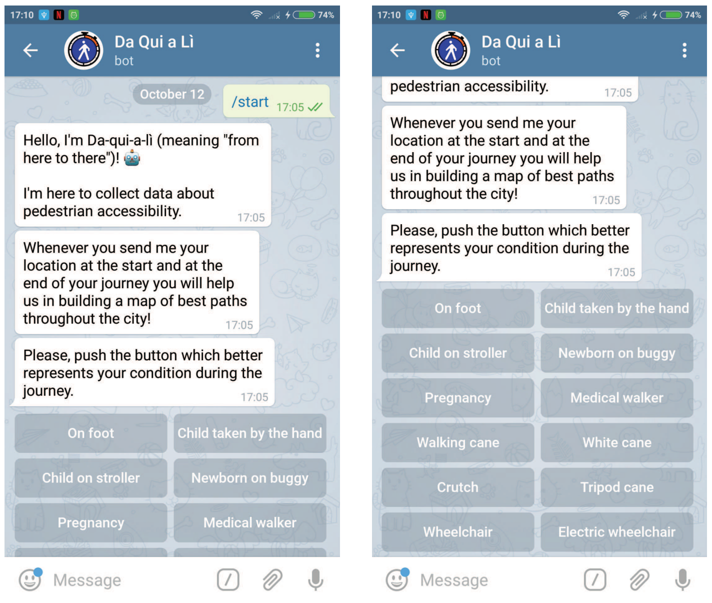
4.2.2 A message-based sensing style

In this Section, we describe the typical usage of Da-qui-a-lì. Volunteers who want to join the MCS application doesn't need to install a dedicated mobile app instead they just have to search for the Da-qui-a-lì bot[5] using the standard Telegram app (Telegram platform

[4]Da-qui-a-lì has been developed in order to demonstrate the suitability of modern conversational interfaces as user interfaces for MCS platforms, thus the implementation of an application including all the MCS phases described in Section 1.2.2 is out of the scope of this work.

[5]The bot is currently available at https://t.me/daquiali_bot

functionalities are also available through a Desktop client, but only the mobile version of the client allows users to share their GPS-based location Da-qui-a-lì aims to collect).



17:10 74%

Da Qui a Lì
bot

October 12

/start 17:05

Hello, I'm Da-qui-a-lì (meaning "from here to there")! 🤖

I'm here to collect data about pedestrian accessibility. 17:05

Whenever you send me your location at the start and at the end of your journey you will help us in building a map of best paths throughout the city! 17:05

Please, push the button which better represents your condition during the journey. 17:05

(a) (b)

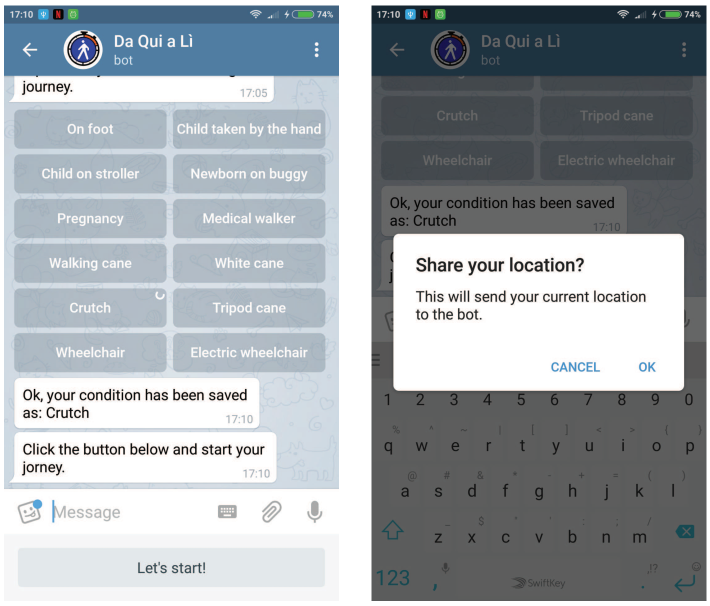
Figure 4.5: *Da-qui-a-lì* first interaction

Telegram's bots are commonly started by typing the \start command. On its first interaction with a new interlocutor, the bot greets him explaining the platform's aim and asking to select the user current condition by indicating any physical impairment he suffers (Figure 4.5). The bot memorizes this information which will be added as metadata to any journey the user logs. If the user condition changes, the relative record can be modified before each recording session.

Each *sensing session* starts when the user sends its current starting position to the bot. The bot presents the user with a convenient interface with just one virtual button a brief explanation of what the user has to do in order to contribute. Thanks to a specific Telegram API, when the user pushes the virtual button (represented in Figure 4.6), his location is shared with the bot along with a textual command. On the server side, the user location is memorized together with the message timestamp.

Once started, the sensing session can be completed by clicking the virtual button "Destination reached!" in Figure 4.7. When the button is clicked, the new user's location is shared once again, and the application server memorizes the new position and the new timestamp.

At the end of each session, the user is asked to provide feedback about the new record. Firstly, data about the length and the duration of the journey have to be confirmed. Then


Da Qui a Lì
bot

journey.

Ok, your condition has been saved as: Crutch

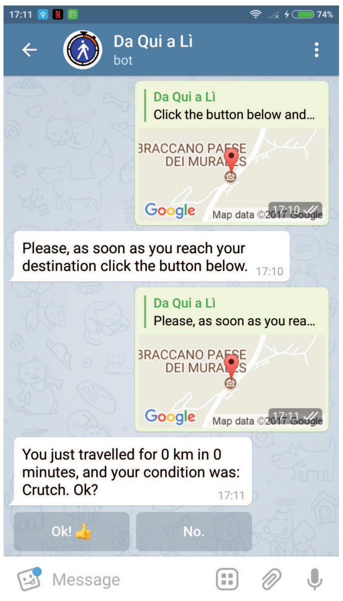
Click the button below and start your jorney.

Share your location?

This will send your current location to the bot.

(a) (b)

Figure 4.6: *Da-qui-a-lì* starting a new journey


Da Qui a Lì
bot

Da Qui a Lì
Click the button below and...

Please, as soon as you reach your destination click the button below.

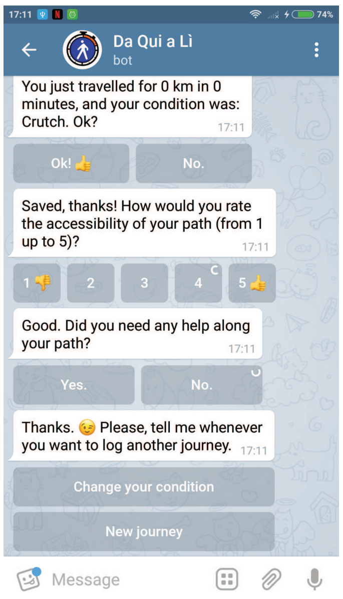
Da Qui a Lì
Please, as soon as you rea...

You just travelled for 0 km in 0 minutes, and your condition was: Crutch. Ok?

Figure 4.7: *Da-qui-a-lì* stopping the recording phase

the volunteer rates the path accessibility and specifies whether some help has been needed to complete the travel or not (Figure 4.8).

At this point, the interaction can be carried on either by starting new journeys click-



You just travelled for 0 km in 0 minutes, and your condition was: Crutch. Ok? 17:11

Saved, thanks! How would you rate the accessibility of your path (from 1 up to 5)? 17:11

Good. Did you need any help along your path? 17:11

Thanks. 😉 Please, tell me whenever you want to log another journey. 17:11

Figure 4.8: *Da-qui-a-lì* evaluating last journey

ing the related button or by selecting another condition to be associated with future records.

The Da-qui-a-lì bot has been released in beta testing version in July 2017 and, to date, many journeys within urban areas have been logged by volunteers. As already mentioned, in its current shape the bot doesn't sport a data processing phase which is currently run offline. The application is still under development, and an hackathon will be organized by the University of Urbino (with the partial support of the Rotary Club Urbino) by the end of November 2017. During the hackathon participants will be able to experiment original methods to deal with data collected until that day. By the end of 2017, a final version of the platform will be released. It will include a data processing and a data distribution phases. Computed best paths will be available as an automatically-updated online map. Further enhancements described in Section 4.3 will be added later on.

4.3 Final remarks

In Section 4.1 we described the emerging phenomenon of Botplications as a new software platform for delivering services and data to users. In the current Section, we demonstrated how a modern-day bot could also act as the mobile client of a distributed MCS application. The design and the implementation of Da-qui-a-lì gives us the chance to describe which are the pros and cons of such a methodology and under which circumstances casting our MCS application as a Botplication could be convenient.

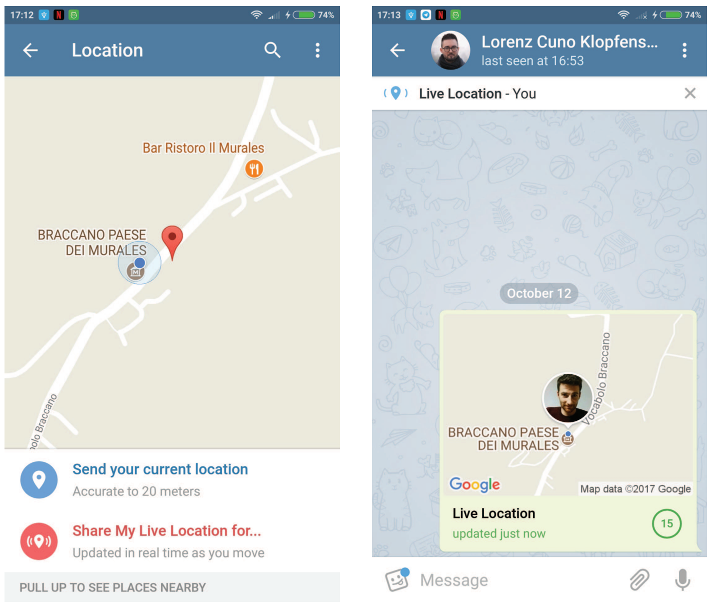
Designing and implementing a multiplatform mobile application can be extremely time and resource-consuming. Modern mobile operative systems allow high-level programming and take care of abstracting hardware and low-level software features to provide a developing platform on top of which is easy to implement even complex applications. On the other hand, the programming framework remains relatively complicated. Its usage learning curve could be quite gentle for an experienced developer while the implementation of a production-ready application is still a hard task for an inexperienced one. Conversely, implementing a bot is a task quickly achievable even for non-experts as most of the major messaging platforms provide ultra-high level programming interfaces. This allows a coding style similar to the declarative programming (at least for what concerns the UI): the user interface is built by the messaging platform following the directives it receives from the web services associated to a particular bot. The core logic of the Botplication can be implemented using any programming framework offering HTTP communication primitives.

Another advantage of designing an MCS app as a bot is that privacy and security communication aspects are already managed by the messaging platform, at least on the client side. Moreover, major messaging systems provide mobile clients for Android, iOS, and Windows, covering almost the totality of the smartphone users. Building an MCS application on top of a messaging system environment, allows developers to program a bot just once and obtaining a multi-platform app *by design*. Botplications live inside other messaging systems. As already mentioned in Section 4.1, mobile messaging clients are becoming more and more popular, and almost every smartphone user have one of them installed on their device. In order to lower the barrier of adoption — an essential task for MCS systems — develop a client as a bot is a reasonable approach as it means no need of app-installation at all. Volunteers just have to search for the desired bot inside the messaging application. Even better, systems like Telegram and Messenger, through the deep-linking mechanism, enable the use of physical means like QR Codes to make a bot directly reachable by the users.

Nonetheless, the proposed approach presents few limitations and disadvantages. First of all, bot-based client rely on many external components, over which developers have little or no control: Internet access, DNS, the IM platform's back-end, the IM mobile clients and more. Each component could be a potential failure point and can introduce performance issues.

The development of a bot-based MCS application interface is relatively easy thanks to the availability of many resources, which are provided by the IM platforms themselves. On the other hand, the bot's interface is hard-connected to the messaging platform system. This strongly inhibits app customizations and makes the app not suitable for MCS tasks needing data pre-processing and local data storage. Moreover, bots are not the best tool for implementing the mobile client of an MCS application based on an implicit sensing style since many sensors cannot be used directly. IM programming interfaces don't provide bots with handlers to communicate to the underlying hardware so, the user should always act as middle-person in feeding bot with sensed data.

One of the key benefits of using the conversational interface from instant messaging systems is that the basic UI paradigms of messaging are well known, both by novice and expert users. However, we found that in many cases users tend not to read any text at all [134]. This may be worsened by the general verbosity of conversational interfaces and by the lack of recognizable UI elements in text messages (e.g., colors, modal dialogs, etc.). In many cases, the effectiveness of messages can be improved using terser language and text emojis to convey tone or status (e.g., a flashing light or a 'stop' sign for errors, sad smiles for wrong answers, etc.). Having to interact mainly through textual messages, the design of any Botplication, and in particular, for MCS bot-based application, should find a way to provide users with useful information avoiding wordiness.



(a) (b)

Figure 4.9: Telegram brand-new *LiveLocations* feature: Figure 4.9a shows how users can share their live location. Figure 4.9b illustrates how live locations looks like once shared (please note the reminder above the chat UI).

It is worth mentioning that this research is still under active development. In the next months, we plan to refine our application and to make some improvements. The next phase of this study will be to replicate the implementation as a Messenger's bot as this platform is currently the most widely used among those we analyzed (at least in the western part of the world). In doing so, we will try to broaden the user base and to increase the number of feedbacks collected. In the following, a focus group will be organized in order to evaluate the convenience of the proposed approach in terms of UI efficacy, in a controlled environment.

In recent days, Telegram released the new *LiveLocations* feature[6]. In Figure 4.9, a mechanism that allows receiving continuous updates about a user's location. The same functionality is available for bots and it could constitute a great enhancement in our application usage paradigm, and, likewise, for any other MCS application focused on user's movement patterns or user's location history. The mechanism will be deeper investigated, and it could be added in a future release of Da-qui-a-lì.

To fully explore the UI resources that the Botplication paradigm unlocks, our application will be used to study the effectiveness of a mixed-reality interaction approach. We will exploit the ability to scan and decode QR Codes using third-party mobile apps together with the already mentioned deep linking functionalities of some IM platforms. The aim would be to lower the adoption barriers even more, by giving to participants the chance to initiate the interaction with the Botplication not only by searching it in dedicated but crowded directories (as it commonly happens today) but also by scanning printed QR Codes, they can find along their paths.

[6]Please see https://telegram.org/blog/live-locations

Chapter 5

Conclusions

In this work, we examined the Mobile Crowd Sensing paradigm. We started analyzing the process and the research progresses that led to a modern and shared definition of what MCS is and what differentiates it from other types of distributed sensing and processing initiatives (e.g., crowdsourcing, participatory sensing, etc.). Then, we surveyed a number of applications in order to pinpoint the distinguishing features of most MCS platforms and to model their common high-level process architecture. The introduction to the MCS topic concluded with the examination of the major open issues that a hypothetical designer of MCS applications would have to face, including preventing privacy leaks, dealing with limited data quality, and facing scalability issues. Throughout this work, we separately examined three open issues, every time trying to look at them from a fresh perspective and always proposing original solutions.

MCS systems are based on the concept of a distributed and unsupervised data sensing activity. In such frameworks, low data quality could hinder or bias the data analysis and the emerge of collective intelligence crowd intelligence extraction. In Chapter2, we discussed how characterizing the quality of sensed data is essential for any crowdsensing initiative. We proposed a novel method to evaluate the data quality based on a simple statistical tool (namely "bootstrap"). This technique enables the estimation of relative quality in large amount of data. It is especially suitable for MCS applications as it does not require any prior knowledge or assumption on the data source's behavior.

We then focused on the typical structure of an MCS data particle, which, along with application-specific values, commonly entails data representing the timespan and the geographical position of when and where the sensing activity has been performed. Those metadata are usually essential for the data processing phase: their quality can also positively or negatively influence the overall intelligence extraction outcome. To this purpose, we proposed a new map-matching algorithm to be applied on dense traces of data, whose aim is to remove errors derived from low-quality GPS recordings. Our approach has been tested in a real-world environment and empirical results show how it can enhance the data quality (and the subsequent association between data points and features on a map) of applications interested in gathering data from moving vehicles.

In Chapter 3 we tackled the problem of providing users with substantial incentives in order to engage them in the sensing endeavor. Distributed sensing processes not only leverage user resources as computational power, battery, and communication capabilities, but also could require a strong commitment from volunteers, in terms of sharing private information such as location history, profile information, and personal social network. Even if cooperation incentives have been in the limelight for about thirty years, we cast the problem of finding ways to motivate users to collaborate toward a shared cause as a privacy issue. Safeguarding the user's privacy and anonymity should be paramount for any MCS platforms but constrictive policies could hinder the implementation of effective motivation mechanisms. In Chapter 3 we proposed a rewarding schemes classification in terms of motivation provided and anonymity of user information. We found that gamification techniques and voucher distribution are mechanisms suitable to be used in anonymous systems in order to engage users. Our analysis went further as we presented a real-world design of a gamification layer built on top of an MCS application implementing strict privacy-preserving techniques. In the following, we also introduced a novel voucher-based rewarding platform, explicitly designed for MCS endeavors. Our system acts as a bridge between volunteers, MCS applications, and third-party stakeholders allowing the latter to be part of the rewarding paradigm without the need of participating in the crowdsensing procedure. The platform uses a voucher exchanging system as a tool for triggering network effects and positive externalities that can help the pursuing of the common good. The main protocols are outlined along with technical details in order to demonstrate the validity and the flexibility of the approach. The platform is currently in the implementation stage, and further developments will be the object of future works.

In Chapter 4 we discussed client UI approaches used in MCS instruments. The client interface is the sole point of interaction between users and platform for most crowdsensing applications. Since having a good number of volunteers is fundamental in MCS, joining a sensing task should not be a burden. In particular, the client UI should not constitute a barrier to platform adoption by users but, unfortunately, most of the MCS platforms used in real-world application still struggle to provide a usable and easy-to-understand UI. In recent times, instant messaging and social network applications consistently take up the top spots among the most used applications. Most of these platforms already provide programming interfaces that allow the implementation of a new generation of conversational interfaces. Our main idea was to exploit the popularity of IM platforms, and the flexibility of the conversational interfaces they host, to empower MCS volunteers with an effective tool for contributing to the sensing task. We first studied the new phenomenon, outlining its main features, pros and cons. A definition of "Botplication" to identify those new approaches in designing mobile applications providing a conversational interface. Also, we implemented a proof of concept in the form of a simple MCS application with a botplication client demonstrating the feasibility of our approach. We analyzed the prototype's interface and drew conclusions on the proposed approach.

In summary, we proposed and discussed many solutions to both well-known and relatively new issues in the context of MCS. In order to keep their treatise as clear as possible, we examined each approach in isolation, providing actual implementations or at least concrete designs of our ideas. Nonetheless, discussed methods are inherently general and could be applied in many different scenarios. As a matter of fact, they could also be employed all together on the proper MCS platform, and they will. Work is currently being done in order to provide an actual integrated implementation of proposed solutions in the C4Rs project.

More in detail, the map-matching algorithm and the methodology to estimate the data quality index presented in Chapter 2 are currently being implemented into the C4Rs data processing server. While the gamification layer we presented in Chapter 3, will be added to the current implementation of the C4Rs mobile client. The MCS Botplication presented in Chapter 4 will be further improved and refined. Once mature enough, it will act as *Instrument* in the context of the voucher-based rewarding platform outlined in Chapter 3.

The development of the rewarding system will be another project's remarkable achievement as it will allow third parties to easily contribute to the common good. The rewarding platform will indeed be a key element since — even if it is specifically designed for MCS contributions — it will allow contributions from outside the crowdsensing context. The overarching intent is to open the platform and use it for external projects pursuing public interests, such as public campaigns about digital skills and inclusion, cultural and scientific literacy actions.

Acknowledgments

This work has been partially by the European Union's Horizon 2020 research and innovation program under grant agreement № 687959.

Bibliography

[1] Karl Aberer, Saket Sathe, Dipanjan Chakraborty, Alcherio Martinoli, Guillermo Barrenetxea, Boi Faltings, and Lothar Thiele. Opensense: open community driven sensing of environment. In *Proceedings of the ACM SIGSPATIAL International Workshop on GeoStreaming*, pages 39–42. ACM, 2010.

[2] G Alessandroni, LC Klopfenstein, S Delpriori, M Dromedari, G Luchetti, B Paolini, A Seraghiti, E Lattanzi, V Freschi, A Carini, et al. Smartroadsense: Collaborative road surface condition monitoring. *Proceedings of the UBICOMM*, pages 210–215, 2014.

[3] Kashif Ali, Dina Al-Yaseen, Ali Ejaz, Tayyab Javed, and Hossam S Hassanein. Crowdits: Crowdsourcing in intelligent transportation systems. In *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*, pages 3307–3311. IEEE, 2012.

[4] Sylvester Arnab, Madhav Nalla, Casper Harteveld, and Petros Lameras. An inquiry into gamification services: Practices, experiences and insights. In *Proceedings of the International Gamification for Business Conference 2015*, pages 34–45, 2015.

[5] Eskindir Asmare and Julie A McCann. Lightweight sensing uncertainty metric—incorporating accuracy and trust. *IEEE Sensors Journal*, 14(12):4264–4272, 2014.

[6] Shiri Azenkot and Shumin Zhai. Touch behavior with different postures on soft smartphone keyboards. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*, pages 251–260. ACM, 2012.

[7] Garvita Bajaj and Pushpendra Singh. Sahyog: A middleware for mobile collaborative applications. In *New Technologies, Mobility and Security (NTMS), 2015 7th International Conference on*, pages 1–5. IEEE, 2015.

[8] Gabriel Barata, Sandra Gama, Manuel J Fonseca, and Daniel Gonçalves. Improving student creativity with gamification and virtual worlds. In *Proceedings of the First International Conference on Gameful Design, Research, and Applications*, pages 95–98. ACM, 2013.

[9] Emanuele Bastianelli, Daniele Nardi, Luigia Carlucci Aiello, Fabrizio Giacomelli, and Nicolamaria Manes. Speaky for robots: the development of vocal interfaces for robotic applications. *Applied Intelligence*, 44(1):43–66, 2016. ISSN 1573-7497. doi: 10.1007/s10489-015-0695-5. URL `http://dx.doi.org/10.1007/s10489-015-0695-5`.

[10] Jerome R Bellegarda. Spoken language understanding for natural interaction: The Siri experience. In *Natural Interaction with Robots, Knowbots and Smartphones*, pages 3–14. Springer, 2014.

[11] Linus Bengtsson, Xin Lu, Anna Thorson, Richard Garfield, and Johan Von Schreeb. Improved response to disasters and outbreaks by tracking population movements with mobile phone network data: a post-earthquake geospatial study in haiti. *PLoS medicine*, 8(8):e1001083, 2011.

[12] IEC BIPM, ILAC IFCC, IUPAP IUPAC, and OIML ISO. Evaluation of measurement data—guide for the expression of uncertainty in measurement. jcgm 100: 2008. *Citado en las*, page 167, 2008.

[13] A. Bogliolo, A. Aldini, G. Alessandroni, A. Carini, S. Delpriori, Freschi V., L. C. Klopfenstein, E. Lattanzi, G. Luchetti, B. D. Paolini, and A. Seraghiti. The SmartRoadSense website, `http://smartroadsense.it/`, 2015. URL `http://smartroadsense.it/`.

[14] Alessandro Bogliolo, Paolo Polidori, Alessandro Aldini, Waldir Moreira, Paulo Mendes, Mürsel Yildiz, Carlos Ballester, and Jean-Marc Seigneur. Virtual currency and reputation-based cooperation incentives in user-centric networks. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2012 8th International*, pages 895–900. IEEE, 2012.

[15] Aikaterini Bourazeri, Jeremy Pitt, and Sylvester Arnab. Social mpower: An educational game for energy efficiency. In *International Conference on Serious Games, Interaction and Simulation*, pages 133–140. Springer, 2016.

[16] Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. On map-matching vehicle tracking data. In *Proceedings of the 31st international conference on Very large data bases*, pages 853–864. VLDB Endowment, 2005.

[17] Muhammed Fatih Bulut, Murat Demirbas, and Hakan Ferhatosmanoglu. Lineking: Coffee shop wait-time monitoring using smartphones. *IEEE Transactions on Mobile Computing*, 14(10):2045–2058, 2015.

[18] Jeffrey A Burke, Deborah Estrin, Mark Hansen, Andrew Parker, Nithya Ramanathan, Sasank Reddy, and Mani B Srivastava. Participatory sensing. *Center for Embedded Network Sensing*, 2006.

[19] Matt Burns. Encrypted messaging app telegram hits 100m monthly active users, 350k new users each day. URL `https://techcrunch.com/2016/02/23/encrypted-messaging-app-telegram-hits-100m-monthly-active-users-350k-new-users-each-day/`, 2016. Accessed: 2017-10-12.

[20] Francesco Calabrese, Massimo Colonna, Piero Lovisolo, Dario Parata, and Carlo Ratti. Real-time urban monitoring using cell phones: A case study in rome. *IEEE Transactions on Intelligent Transportation Systems*, 12(1):141–151, 2011.

[21] Francesco Calabrese, Laura Ferrari, and Vincent D Blondel. Urban sensing using mobile phone network data: a survey of research. *Acm computing surveys (csur)*, 47(2):25, 2015.

[22] Giuseppe Cardone, Luca Foschini, Paolo Bellavista, Antonio Corradi, Cristian Borcea, Manoop Talasila, and Reza Curtmola. Fostering participaction in smart cities: a geo-social crowdsensing platform. *IEEE Communications Magazine*, 51(6):112–119, 2013.

[23] Iacopo Carreras, Daniele Miorandi, Andrei Tamilin, Emmanuel R Ssebaggala, and Nicola Conci. Crowd-sensing: Why context matters. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*, pages 368–371. IEEE, 2013.

[24] Iacopo Carreras, Daniele Miorandi, Andrei Tamilin, Emmanuel R Ssebaggala, and Nicola Conci. Matador: Mobile task detector for context-aware crowd-sensing campaigns. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*, pages 212–217. IEEE, 2013.

[25] Justine Cassell, Timothy Bickmore, Mark Billinghurst, Lee Campbell, Kenny Chang, Hannes Vilhjálmsson, and Hao Yan. Embodiment in conversational interfaces: Rea. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 520–527. ACM, 1999.

[26] Muhammad Benny Chaniago and Apri Junaidi. Student presence using RFID and Telegram Messenger application. 8th Widyatama International Seminar on Sustainability (WISS 2016), Widyatama University and IEEE, 2016.

[27] Andre Charland and Brian Leroux. Mobile application development: Web vs. native. *Commun. ACM*, 54(5):49–53, May 2011. ISSN 0001-0782. doi: 10.1145/1941487.1941504. URL `http://doi.acm.org/10.1145/1941487.1941504`.

[28] Daniel Chen, Anne Driemel, Leonidas J Guibas, Andy Nguyen, and Carola Wenk. Approximate map matching with respect to the fréchet distance. In *Proceedings of the meeting on algorithm engineering & expermiments*, pages 75–83. Society for Industrial and Applied Mathematics, 2011.

[29] Yohan Chon, Nicholas D Lane, Fan Li, Hojung Cha, and Feng Zhao. Automatically characterizing places with opportunistic crowdsensing using smartphones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 481–490. ACM, 2012.

[30] Yu-Kai Chou. Octalysis: Complete gamification framework. *Yu-Kai Chou & Gamification*, 2013.

[31] Sangeet Paul Choudary, Marshall W Van Alstyne, and Geoffrey G Parker. Platform revolution: How networked markets are transforming the economy–and how to make them work for you. 2016.

[32] Samantha Clarke, Sylvester Arnab, Ian Dunwell, and Katherine Brown. Pr: Epare: A game-based approach to relationship guidance for adolescents. *Procedia Computer Science*, 15:38–44, 2012.

[33] Samantha Clarke, Sylvester Arnab, Mark Lewis, Luca Morini, Saverio Delpriori, Alessandro Bogliolo, and Cuno Lorenz Klopfenstein. A gamified approach to facilitate a user-engagement strategy for public-led, collective awareness of sustainable travel habits. In *European Conference on Games Based Learning (ECGBL) 2017*, 2017.

[34] European Commision. Eu transport in figures. statistical pocketbook. URL `http://ec.europa.eu/transport/facts-fundings/statistics/doc/2014/pocketbook2014.pdf`, 2014.

[35] comScore. The U.S. mobile app report. URL `https://www.comscore.com/Insights/Presentations-and-Whitepapers/2014/The-US-Mobile-App-Report`, 2014. Accessed: 2017-15-09.

[36] VW Consulting. mhealth for development: the opportunity of mobile technology for healthcare in the developing world. washington, dc and berkshire, uk: Un foundation-vodafone foundation partnership. 2009, 2014.

[37] Marco Conti and Mohan Kumar. Opportunities in opportunistic computing. *Computer*, 43(1), 2010.

[38] M Cox, P Harris, and BR-L Siebert. Evaluation of measurement uncertainty based on the propagation of distributions using monte carlo simulation. *Measurement Techniques*, 46(9):824–833, 2003.

[39] MG Cox. Propagation of distributions by a monte carlo method, with an application to ratio models. *The European Physical Journal-Special Topics*, 172(1):153–162, 2009.

[40] Andrew Crooks, Arie Croitoru, Anthony Stefanidis, and Jacek Radzikowski. # earthquake: Twitter as a distributed sensor system. *Transactions in GIS*, 17(1):124–147, 2013.

[41] Dana Cuff, Mark Hansen, and Jerry Kang. Urban sensing: out of the woods. *Communications of the ACM*, 51(3):24–33, 2008.

[42] Vickie Curtis. Motivation to participate in an online citizen science game: A study of foldit. *Science Communication*, 37(6):723–746, 2015.

[43] Emilia Danowska-Florczyk and Piotr Mostowski. Gamification as a new direction in teaching polish as a foreign language. *ICT for Language Learning*, 2012.

[44] Tathagata Das, Prashanth Mohan, Venkata N Padmanabhan, Ramachandran Ramjee, and Asankhaya Sharma. Prism: platform for remote sensing using smartphones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 63–76. ACM, 2010.

[45] Saverio Delpriori, Valerio Freschi, Emanuele Lattanzi, and Alessandro Bogliolo. Efficient algorithms for accuracy improvement in mobile crowdsensing vehicular applications. *UBICOMM 2015*, page 158, 2015.

[46] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. From game design elements to gamefulness: defining gamification. In *Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments*, pages 9–15. ACM, 2011.

[47] Transport DETR (Department Of Environment and The Regions). A new deal for transport: Better for everyone. the government's white paper on the future of transport. Technical report, London: Department of the Environment, Transport and the Regions, 1998.

[48] Bradley Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics*, pages 569–593. Springer, 1992.

[49] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[50] Jakob Eriksson, Lewis Girod, Bret Hull, Ryan Newton, Samuel Madden, and Hari Balakrishnan. The pothole patrol: using a mobile sensor network for road surface monitoring. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pages 29–39. ACM, 2008.

[51] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of cryptology*, 1 (2):77–94, 1988.

[52] Neil M Ferguson, Derek AT Cummings, Christophe Fraser, James C Cajka, Philip C Cooley, and Donald S Burke. Strategies for mitigating an influenza pandemic. *Nature*, 442(7101):448, 2006.

[53] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79, 2010.

[54] Luciano Floridi. Open data, data protection, and group privacy. *Philosophy & Technology*, 27(1): 1–3, 2014.

[55] FA Fonte, Juan Carlos, Burguillo Rial, and Martín Llamas Nistal. Tq-bot: an aiml-based tutor and evaluator bot. *Journal of Universal Computer Science*, 15(7):1486–1495, 2009.

[56] Richard TT Forman. *Road ecology: science and solutions*. Island Press, 2003.

[57] Valerio Freschi, Saverio Delpriori, Lorenz Cuno Klopfenstein, Emanuele Lattanzi, Gioele Luchetti, and Alessandro Bogliolo. Geospatial data aggregation and reduction in vehicular sensing applications: The case of road surface monitoring. In *Connected Vehicles and Expo (ICCVE), 2014 International Conference on*, pages 711–716. IEEE, 2014.

[58] Valerio Freschi, Saverio Delpriori, Emanuele Lattanzi, and Alessandro Bogliolo. Bootstrap based uncertainty propagation for data quality estimation in crowdsensing systems. *IEEE Access*, 5:1146–1155, 2017.

[59] Elia Gabarron, Thomas Schopf, J Artur Serrano, Luis Fernández Luque, and Enrique Dorronzoro Zubiete. Gamification strategy on prevention of stds for youth. In *Medinfo2013: The 14th World Congress on Medical and Health Informatics (2013),*. IMIA, 2013.

[60] Raghu K Ganti, Fan Ye, and Hui Lei. Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine*, 49(11), 2011.

[61] Anargyros Garyfalos and Kevin C Almeroth. Coupons: A multilevel incentive scheme for information dissemination in mobile networks. *IEEE Transactions on Mobile Computing*, 7(6):792–804, 2008.

[62] Fosca Giannotti, Lorenzo Gabrielli, Dino Pedreschi, and Salvatore Rinzivillo. Understanding human mobility with big data. In *Solving Large Scale Learning Tasks. Challenges and Algorithms*, pages 208–220. Springer International Publishing, 2016.

[63] Geoff Goehle. Gamification and web-based homework. *Primus*, 23(3):234–246, 2013.

[64] Chong Yang Goh, Justin Dauwels, Nikola Mitrovic, Muhammad Tayyab Asif, Ali Oran, and Patrick Jaillet. Online map-matching based on hidden markov model for real-time traffic sensing applications. In *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pages 776–781. IEEE, 2012.

[65] Phil Goodwin. Transformation of transport policy in great britain. *Transportation Research Part A: Policy and Practice*, 33(7):655–669, 1999.

[66] Aldo Gordillo, Daniel Gallego, Enrique Barra, and Juan Quemada. The city as a learning gamified platform. In *Frontiers in Education Conference, 2013 IEEE*, pages 372–378. IEEE, 2013.

[67] Bettina Graf, Maike Krüger, Felix Müller, Alexander Ruhland, and Andrea Zech. Nombot: simplify food tracking. In *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia*, pages 360–363. ACM, 2015.

[68] Nathan Green, Jan Kruger, Chirag Faldu, and Robert St. Amant. A reduced qwerty keyboard for mobile text entry. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '04, pages 1429–1432, New York, NY, USA, 2004. ACM. ISBN 1-58113-703-6. doi: 10.1145/985921.986082. URL `http://doi.acm.org/10.1145/985921.986082`.

[69] Bin Guo, Daqing Zhang, Zhu Wang, Zhiwen Yu, and Xingshe Zhou. Opportunistic iot: Exploring the harmonious interaction between human and the internet of things. *Journal of Network and Computer Applications*, 36(6):1531–1539, 2013.

[70] Bin Guo, Daqing Zhang, Zhiwen Yu, Yunji Liang, Zhu Wang, and Xingshe Zhou. From the internet of things to embedded intelligence. *World Wide Web*, 16(4):399–420, 2013.

[71] Bin Guo, Zhiwen Yu, Xingshe Zhou, and Daqing Zhang. From participatory sensing to mobile crowd sensing. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on*, pages 593–598. IEEE, 2014.

[72] Bin Guo, Zhu Wang, Zhiwen Yu, Yu Wang, Neil Y. Yen, Runhe Huang, and Xingshe Zhou. Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm. *ACM Comput. Surv.*, 48(1):7:1–7:31, August 2015. ISSN 0360-0300. doi: 10.1145/2794400. URL `http://doi.acm.org/10.1145/2794400`.

[73] Bin Guo, Chao Chen, Daqing Zhang, Zhiwen Yu, and Alvin Chin. Mobile crowd sensing and computing: when participatory sensing meets participatory social media. *IEEE Communications Magazine*, 54(2):131–137, 2016.

[74] Charles T Hemphill, John J Godfrey, and George R Doddington. The ATIS spoken language systems pilot corpus. In *Proceedings of the DARPA speech and natural language workshop*, pages 96–101, 1990.

[75] Dan Hermes. Mobile development using xamarin. In *Xamarin Mobile Application Development*, pages 1–8. Springer, 2015.

[76] Tim C Hesterberg. What teachers should know about the bootstrap: Resampling in the undergraduate statistics curriculum. *The American Statistician*, 69(4):371–386, 2015.

[77] Takamasa Higuchi, Hirozumi Yamaguchi, Teruo Higashino, and Mineo Takai. A neighbor collaboration mechanism for mobile crowd sensing in opportunistic networks. In *Communications (ICC), 2014 IEEE International Conference on*, pages 42–47. IEEE, 2014.

[78] Jeff Howe. Crowdsourcing: A definition. *Crowdsourcing: Tracking the rise of the amateur*, 2006.

[79] Jeff Howe. The rise of crowdsourcing. *Wired magazine*, 14(6):1–4, 2006.

[80] Shaohan Hu, Lu Su, Hengchang Liu, Hongyan Wang, and Tarek F Abdelzaher. Smartroad: Smartphone-based crowd sensing for traffic regulator detection and identification. *ACM Transactions on Sensor Networks (TOSN)*, 11(4):55, 2015.

[81] Xiping Hu, Xitong Li, Edith Ngai, Victor Leung, and Philippe Kruchten. Multidimensional context-aware social network architecture for mobile crowdsensing. *IEEE Communications Magazine*, 52(6):78–87, 2014.

[82] Kuan Lun Huang, Salil S Kanhere, and Wen Hu. Are you contributing trustworthy data?: the case for a reputation system in participatory sensing. In *Proceedings of the 13th ACM international conference on Modeling, analysis, and simulation of wireless and mobile systems*, pages 14–22. ACM, 2010.

[83] Kuan Lun Huang, Salil S Kanhere, and Wen Hu. On the need for a reputation system in mobile phone based sensing. *Ad Hoc Networks*, 12:130–149, 2014.

[84] Shenlong Huangfu, Bin Guo, Zhiwen Yu, and Dongsheng Li. Using the model of markets with intermediaries as an incentive scheme for opportunistic social networks. In *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*, pages 142–149. IEEE, 2013.

[85] Adrian Iftene and Jean Vanderdonckt. MOOCBuddy: a chatbot for personalized learning with MOOCs. In *RoCHI–International Conference on Human-Computer Interaction*, page 91, 2016.

[86] Luis G Jaimes, Idalides J Vergara-Laurens, and Andrew Raij. A survey of incentive techniques for mobile crowd sensing. *IEEE Internet of Things Journal*, 2(5):370–380, 2015.

[87] Philippe Jeanrenaud, Greg Cockroft, and Allard VanderHeidjen. A multimodal, multilingual telephone application: the wildfire electronic assistant. In *EUROSPEECH*, 1999.

[88] Haiming Jin, Lu Su, Bolin Ding, Klara Nahrstedt, and Nikita Borisov. Enabling privacy-preserving incentives for mobile crowd sensing systems. In *Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on*, pages 344–353. IEEE, 2016.

[89] Peter Jones and Karen Lucas. The social consequences of transport decision-making: clarifying concepts, synthesising knowledge and assessing implications. *Journal of transport geography*, 21:4–16, 2012.

[90] Dmytro Karamshuk, Anastasios Noulas, Salvatore Scellato, Vincenzo Nicosia, and Cecilia Mascolo. Geo-spotting: mining online location-based services for optimal retail store placement. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 793–801. ACM, 2013.

[91] Robert E Kass, Uri T Eden, and Emery N Brown. Propagation of uncertainty and the bootstrap. In *Analysis of Neural Data*, pages 221–246. Springer, 2014.

[92] Henry Kautz, Bart Selman, Michael Coen, Steven Ketchpel, and Chris Ramming. An experiment in the design of software agents. In *AAAI*, pages 438–443, 1994.

[93] Felix Keis and Kevin Wiesner. Participatory sensing utilized by an advanced meteorological nowcasting system. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014 IEEE Ninth International Conference on*, pages 1–6. IEEE, 2014.

[94] Ian R Kerr. Bots, babes and the californication of commerce. *University of Ottawa Law and Technology Journal*, 1(1-2):20004, 2004.

[95] Wazir Zada Khan, Yang Xiang, Mohammed Y Aalsalem, and Quratulain Arshad. Mobile phone sensing systems: A survey. *IEEE Communications Surveys & Tutorials*, 15(1):402–427, 2013.

[96] Lorenz Cuno Klopfenstein and Alessandro Bogliolo. The quiz-master bot: a persistent augmented quiz delivered through online messaging. In *INTED2017 Proceedings*, 11th International Technology, Education and Development Conference, pages 9806–9811. IATED, 6-8 March, 2017 2017. ISBN 978-84-617-8491-2. doi: 10.21125/inted.2017.2328. URL `http://dx.doi.org/10.21125/inted.2017.2328`.

[97] Lorenz Cuno Klopfenstein, Saverio Delpriori, Silvia Malatini, and Alessandro Bogliolo. The rise of bots: A survey of conversational interfaces, patterns, and paradigms. In *Proceedings of the 2017 Conference on Designing Interactive Systems*, pages 555–565. ACM, 2017.

[98] Stefan Kopp, Lars Gesellensetter, Nicole C Krämer, and Ipke Wachsmuth. A conversational agent as museum guide–design and evaluation of a real-world application. In *International Workshop on Intelligent Virtual Agents*, pages 329–343. Springer, 2005.

[99] David M Kreps. Intrinsic motivation and extrinsic incentives. *The American Economic Review*, 87 (2):359–364, 1997.

[100] Anthony Kulesa, Martin Krzywinski, Paul Blainey, and Naomi Altman. Points of significance: sampling distributions and the bootstrap. *Nature Methods*, 12(6):477–478, 2015.

[101] Rajiv Kumar, Abhijit Mukherjee, and VP Singh. Community sensor network for monitoring road roughness using smartphones. *Journal of Computing in Civil Engineering*, 31(3):04016059, 2016.

[102] Nicholas D Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T Campbell. A survey of mobile phone sensing. *IEEE Communications magazine*, 48(9), 2010.

[103] Doug Laney. 3d data management: Controlling data volume, velocity and variety. *META Group Research Note*, 6:70, 2001.

[104] Soojeong Lee, Chee-Hyun Park, and Joon-Hyuk Chang. Improved gaussian mixture regression based on pseudo feature generation using bootstrap in blood pressure estimation. *IEEE Transactions on Industrial Informatics*, 12(6):2269–2280, 2016.

[105] Pierre Lévy. *Collective intelligence*. Plenum/Harper Collins New York, 1997.

[106] Cen Li, Zhijiang Dong, Roland H Untch, and Michael Chasteen. Engaging computer science students through gamification in an online social network based collaborative learning environment. *International Journal of Information and Education Technology*, 3(1):72, 2013.

[107] Fang Liu, Chun Wang, Andres Pico, Danfeng Yao, and Gang Wang. Measuring the insecurity of mobile deep links of android. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, 2017.

[108] Liang Liu, Assaf Biderman, and Carlo Ratti. Urban mobility landscape: Real time monitoring of urban mobility patterns. In *Proceedings of the 11th International Conference on Computers in Urban Planning and Urban Management*, pages 1–16, 2009.

[109] Yuan Liu and Chunyan Miao. A survey of incentives and mechanism design for human computation systems. *arXiv preprint arXiv:1602.03277*, 2016.

[110] Malamati Louta, Konstantina Mpanti, George Karetsos, and Thomas Lagkas. Mobile crowd sensing architectural frameworks: A comprehensive survey. In *Information, Intelligence, Systems & Applications (IISA), 2016 7th International Conference on*, pages 1–7. IEEE, 2016.

[111] Thomas Ludwig, Tim Siebigteroth, and Volkmar Pipek. Crowdmonitor: Monitoring physical and digital activities of citizens during emergencies. In *International Conference on Social Informatics*, pages 421–428. Springer, 2014.

[112] Thomas Ludwig, Christian Reuter, Tim Siebigteroth, and Volkmar Pipek. Crowdmonitor: Mobile crowd sensing for assessing physical and digital activities of citizens during emergencies. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 4083–4092. ACM, 2015.

[113] Yunior Luis, Pedro M Santos, Tiago Lourenco, Carlos Pérez-Penichet, Tânia Calcada, and Ana Aguiar. Urbansense: An urban-scale sensing platform for the internet of things. In *Smart Cities Conference (ISC2), 2016 IEEE International*, pages 1–6. IEEE, 2016.

[114] Tie Luo, Salil S Kanhere, and Hwee-Pink Tan. Sew-ing a simple endorsement web to incentivize trustworthy participatory sensing. In *Sensing, Communication, and Networking (SECON), 2014 Eleventh Annual IEEE International Conference on*, pages 636–644. IEEE, 2014.

[115] Tie Luo, Salil S Kanhere, Jianwei Huang, Sajal K Das, and Fan Wu. Sustainable incentives for mobile crowdsensing: Auctions, lotteries, and trust and reputation systems. *IEEE Communications Magazine*, 55(3):68–74, 2017.

[116] I Scott MacKenzie, Shawn X Zhang, and R William Soukoreff. Text entry using soft keyboards. *Behaviour & Information technology*, 18(4):235–244, 1999.

[117] RP Mahapatra, Naresh Sharma, Aakash Trivedi, and Chitransh Aman. Adding interactive interface to e-government systems using AIML based chatterbots. In *Software Engineering (CONSEG), 2012 CSI Sixth International Conference on*, pages 1–6. IEEE, 2012.

[118] Nicolas Maisonneuve, Matthias Stevens, and Bartek Ochab. Participatory noise pollution monitoring using mobile phones. *Information Polity*, 15(1, 2):51–71, 2010.

[119] Thomas W Malone, Robert Laubacher, and Chrysanthos Dellarocas. Harnessing crowds: Mapping the genome of collective intelligence. 2009.

[120] Sumudu Hasala Marakkalage, Billy Pik Lik Lau, Viswanath Sanjana Kadaba, Thirunavukarasu Balasubramaniam, Chau Yuen, Belinda Yuen, and Richi Nayak. Identifying points of interest for elderly in singapore through mobile crowdsensing. 2017.

[121] Oleksiy Mazhelis. Using recursive bayesian estimation for matching gps measurements to imperfect road network data. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 1492–1497. IEEE, 2010.

[122] Jane McGonigal. *Reality is broken: Why games make us better and how they can change the world.* Penguin, 2011.

[123] Michael McTear, Zoraida Callejas, and David Griol. *The Conversational Interface*. Springer, 2016.

[124] Rens Meijkamp. Changing consumer behaviour through eco-efficient services: an empirical study of car sharing in the netherlands. *Business Strategy and the Environment*, 7(4):234–244, 1998.

[125] Giovanni Merlino, Stamatis Arkoulis, Salvatore Distefano, Chrysa Papagianni, Antonio Puliafito, and Symeon Papavassiliou. Mobile crowdsensing as a service: a platform for applications on top of sensing clouds. *Future Generation Computer Systems*, 56:623–639, 2016.

[126] Fernando A Mikic, Juan C Burguillo, Martín Llamas, Daniel A Rodríguez, and Eduardo Rodríguez. CHARLIE: An AIML-based chatterbot which works as an interface among INES and humans. In *EAEEIE Annual Conference, 2009*, pages 1–6. IEEE, 2009.

[127] Emiliano Miluzzo, Cory T Cornelius, Ashwin Ramaswamy, Tanzeem Choudhury, Zhigang Liu, and Andrew T Campbell. Darwin phones: the evolution of sensing and inference on mobile phones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 5–20. ACM, 2010.

[128] Laurent Moccozet, Camille Tardy, Wanda Opprecht, and Michel Léonard. Gamification-based assessment of group work. In *Interactive Collaborative Learning (ICL), 2013 International Conference on*, pages 171–179. IEEE, 2013.

[129] Prashanth Mohan, Venkata N Padmanabhan, and Ramachandran Ramjee. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 323–336. ACM, 2008.

[130] San Murugesan. Mobile apps in africa. *IT Professional*, 15(5):8–11, 2013.

[131] Paul Newson and John Krumm. Hidden markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 336–343. ACM, 2009.

[132] Dong-Anh Nguyen, Tarek Abdelzaher, Steven Borbash, Xuan-Hong Dang, Raghu Ganti, Ambuj Singh, and Mudhakar Srivatsa. On critical event observability using social networks: A disaster monitoring perspective. In *Military Communications Conference (MILCOM), 2014 IEEE*, pages 1633–1638. IEEE, 2014.

[133] Phuong Nguyen and Klara Nahrstedt. Context-aware crowd-sensing in opportunistic mobile social networks. In *Mobile Ad Hoc and Sensor Systems (MASS), 2015 IEEE 12th International Conference on*, pages 477–478. IEEE, 2015.

[134] Jakob Nielsen. *Designing web usability: The practice of simplicity*. New Riders Publishing, 1999.

[135] Anastasios Noulas, Salvatore Scellato, Renaud Lambiotte, Massimiliano Pontil, and Cecilia Mascolo. A tale of many cities: universal patterns in human urban mobility. *PloS one*, 7(5):e37027, 2012.

[136] Dragan Obradovic, Henning Lenz, and Markus Schupfner. Fusion of map and sensor data in a modern car navigation system. *The Journal of VLSI Signal Processing*, 45(1):111–122, 2006.

[137] Robert Ighodaro Ogie. Adopting incentive mechanisms for large-scale participation in mobile crowdsensing: from literature review to a conceptual framework. *Human-centric Computing and Information Sciences*, 6(1):24, 2016.

[138] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . `https://www.openstreetmap.org`, 2017. Accessed: 2017-10-10.

[139] Gang Pan, Guande Qi, Zhaohui Wu, Daqing Zhang, and Shijian Li. Land-use classification using taxi gps traces. *IEEE Transactions on Intelligent Transportation Systems*, 14(1):113–123, 2013.

[140] Geoffrey G Parker, Marshall W Van Alstyne, and Sangeet Paul Choudary. *Platform revolution: How networked markets are transforming the economy–and how to make them work for you*. WW Norton & Company, 2016.

[141] C4Rs partner consortium. The Crowd4Roads website, `http://www.c4rs.eu`, 2015. URL `http://www.c4rs.eu`.

[142] Jeremy Peckham. Speech understanding and dialogue over the telephone: an overview of the ESPRIT SUNDIAL project. In *HLT*, 1991.

[143] Jeremy Pitt, Aikaterini Bourazeri, Andrzej Nowak, Magda Roszczynska-Kurasinska, Agnieszka Rychwalska, Inmaculada Rodriguez Santiago, Maite Lopez Sanchez, Monica Florea, and Mihai Sanduleac. Transforming big data into collective awareness. *Computer*, 46(6):40–45, 2013.

[144] Georgios Portokalidis, Philip Homburg, Kostas Anagnostakis, and Herbert Bos. Paranoid android: versatile protection for smartphones. In *Proceedings of the 26th Annual Computer Security Applications Conference*, pages 347–356. ACM, 2010.

[145] Layla Pournajaf, Li Xiong, Daniel A Garcia-Ulloa, and Vaidy Sunderam. A survey on privacy in mobile crowd sensing task management. *Tech. Rep. TR-2014-002*, 2014.

[146] Layla Pournajaf, Daniel A Garcia-Ulloa, Li Xiong, and Vaidy Sunderam. Participant privacy in mobile crowd sensing task management: A survey of methods and challenges. *ACM SIGMOD Record*, 44(4):23–34, 2016.

[147] Evangelos Pournaras, Izabela Moise, and Dirk Helbing. Privacy-preserving ubiquitous social mining via modular and compositional virtual sensors. In *Advanced Information Networking and Applications (AINA), 2015 IEEE 29th International Conference on*, pages 332–338. IEEE, 2015.

[148] Bratislav Predić, Zhixian Yan, Julien Eberle, Dragan Stojanovic, and Karl Aberer. Exposuresense: Integrating daily activities with air quality using mobile participatory sensing. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*, pages 303–305. IEEE, 2013.

[149] Rüdiger Pryss, Manfred Reichert, Berthold Langguth, and Winfried Schlee. Mobile crowd sensing services for tinnitus assessment, therapy, and research. In *Mobile Services (MS), 2015 IEEE International Conference on*, pages 352–359. IEEE, 2015.

[150] Cesar AV Queiroz and Surhid Gautam. *Road infrastructure and economic development: some diagnostic indicators*, volume 921. World Bank Publications, 1992.

[151] Moo-Ryong Ra, Bin Liu, Tom F La Porta, and Ramesh Govindan. Medusa: A programming framework for crowd-sensing applications. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 337–350. ACM, 2012.

[152] Rajib Kumar Rana, Chun Tung Chou, Salil S Kanhere, Nirupama Bulusu, and Wen Hu. Ear-phone: an end-to-end participatory urban noise mapping system. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 105–116. ACM, 2010.

[153] Sasank Reddy, Deborah Estrin, and Mani Srivastava. Recruitment framework for participatory sensing data collections. In *International Conference on Pervasive Computing*, pages 138–155. Springer, 2010.

[154] Torsten Reiners, Lincoln C Wood, Vanessa Chang, Christian Gütl, Jan Herrington, Hanna Teräs, and Sue Gregory. Operationalising gamification in an educational authentic environment. 2012.

[155] John Rice. *Mathematical statistics and data analysis*. Nelson Education, 2006.

[156] Leonard Richardson and Sam Ruby. *RESTful web services*. " O'Reilly Media, Inc.", 2008.

[157] Kelly Elizabeth Rouse. *Gamification in science education: The relationship of educational games to motivation and achievement*. The University of Southern Mississippi, 2013.

[158] Lukas Ruge, Bashar Altakrouri, and Andreas Schrader. Soundofthecity-continuous noise monitoring for a healthy city. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*, pages 670–675. IEEE, 2013.

[159] David Salomon. *Data compression: the complete reference*. Springer Science & Business Media, 2004.

[160] Antonella Santangelo, Agnese Augello, Antonio Gentile, Giovanni Pilato, and Salvatore Gaglio. A chat-bot based multimodal virtual guide for cultural heritage tours. In *PSC*, pages 114–120, 2006.

[161] Saket Sathe, Timos Sellis, and Karl Aberer. On crowdsensed data acquisition using multidimensional point processes. In *Data Engineering Workshops (ICDEW), 2015 31st IEEE International Conference on*, pages 124–128. IEEE, 2015.

[162] Md Shahriare Satu, Md Hasnat Parvez, et al. Review of integrated applications with AIML based chatbot. In *2015 International Conference on Computer and Information Engineering (ICCIE)*, pages 87–90. IEEE, 2015.

[163] Stephanie Seneff and Joseph Polifroni. Dialogue management in the mercury flight reservation system. In *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational Systems - Volume 3*, ANLP/NAACL-ConvSyst '00, pages 11–16, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics. doi: 10.3115/1117562.1117565. URL `http://dx.doi.org/10.3115/1117562.1117565`.

[164] Carl Shapiro and Hal R Varian. *Information rules: a strategic guide to the network economy*. Harvard Business Press, 1998.

[165] Abu Shawar, Eric Atwell, and Andrew Roberts. FAQchat as in information retrieval system. In *Human Language Technologies as a Challenge for Computer Science and Linguistics: Proceedings of the 2nd Language and Technology Conference*, pages 274–278. Poznan: Wydawnictwo Poznanskie: with co-operation of Fundacja Uniwersytetu im. A. Mickiewicza, 2005.

[166] Bayan Abu Shawar and Eric Atwell. Chatbots: are they really useful? In *LDV Forum*, volume 22, pages 29–49, 2007.

[167] Xiang Sheng, Jian Tang, Xuejie Xiao, and Guoliang Xue. Leveraging gps-less sensing scheduling for green mobile crowd sensing. *IEEE Internet of Things Journal*, 1(4):328–336, 2014.

[168] Swapneel Kalpesh Sheth, Jonathan Schaffer Bell, and Gail E Kaiser. Increasing student engagement in software engineering with gamification. *Department of Computer Science, Columbia University, New York, NY, USA. Columbia University Computer Science Technical Reports. Dpartment of Computer Science, Columbia University*, 2012.

[169] Minho Shin, Cory Cornelius, Dan Peebles, Apu Kapadia, David Kotz, and Nikos Triandopoulos. Anonysense: A system for anonymous opportunistic sensing. *Pervasive and Mobile Computing*, 7 (1):16–30, 2011.

[170] James Surowiecki. *The wisdom of crowds*. Anchor, 2005.

[171] Arnab Sylvester. Gamification as responsible experience design disrupting the new norm and fostering a hero's journey. *First Person Scholar*, 2016.

[172] John Taylor. *Introduction to error analysis, the study of uncertainties in physical measurements*. 1997.

[173] Telegram LLP. The Telegram messenger. URL `https://telegram.org`, 2017. Accessed: 2017-15-09.

[174] Telegram LLP. The Telegram Bot API. URL `https://core.telegram.org/bots`, 2017. Accessed: 2017-15-09.

[175] Arvind Thiagarajan, Lenin Ravindranath, Katrina LaCurts, Samuel Madden, Hari Balakrishnan, Sivan Toledo, and Jakob Eriksson. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 85–98. ACM, 2009.

[176] Arvind Thiagarajan, Lenin Ravindranath, Hari Balakrishnan, Samuel Madden, and Lewis Girod. Accurate, low-energy trajectory mapping for mobile devices. 2011.

[177] Alan M Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[178] Nagendra R Velaga, Mohammed A Quddus, and Abigail L Bristow. Developing an enhanced weight-based topological map-matching algorithm for intelligent transport systems. *Transportation Research Part C: Emerging Technologies*, 17(6):672–683, 2009.

[179] Richard Wallace. The elements of AIML style. *Alice AI Foundation*, 2003.

[180] Richard S Wallace. The anatomy of ALICE. In *Parsing the Turing Test*, pages 181–210. Springer, 2009.

[181] Jing Wang, Jian Tang, Dejun Yang, Erica Wang, and Guoliang Xue. Quality-aware and fine-grained incentive mechanisms for mobile crowdsensing. In *Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on*, pages 354–363. IEEE, 2016.

[182] Leye Wang, Daqing Zhang, Yasha Wang, Chao Chen, Xiao Han, and Abdallah M'hamed. Sparse mobile crowdsensing: challenges and opportunities. *IEEE Communications Magazine*, 54(7):161–167, 2016.

[183] Tricia Wang. Big data needs thick data. *Ethnography Matters*, 13, 2013.

[184] Yin Wang, Xuemei Liu, Hong Wei, George Forman, Chao Chen, and Yanmin Zhu. Crowdatlas: Self-updating maps for cloud and personal use. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pages 27–40. ACM, 2013.

[185] Diane Watson, Mark Hancock, and Regan L Mandryk. Gamifying behaviour that leads to learning. In *Proceedings of the First International Conference on gameful design, research, and applications*, pages 87–90. ACM, 2013.

[186] Joseph Weizenbaum. ELIZA - a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45, January 1966. ISSN 0001-0782. doi: 10.1145/365153.365168. URL `http://doi.acm.org/10.1145/365153.365168`.

[187] Huimin Wen, Zhongwei Hu, Jifu Guo, Liyun Zhu, and Jianping Sun. Operational analysis on beijing road network during the olympic games. *Journal of Transportation Systems Engineering and Information Technology*, 8(6):32–37, 2008.

[188] Yutian Wen, Jinyu Shi, Qi Zhang, Xiaohua Tian, Zhengyong Huang, Hui Yu, Yu Cheng, and Xuemin Shen. Quality-driven auction-based incentive mechanism for mobile crowd sensing. *IEEE Transactions on Vehicular Technology*, 64(9):4203–4214, 2015.

[189] Amy Wesolowski, Nathan Eagle, Andrew J Tatem, David L Smith, Abdisalan M Noor, Robert W Snow, and Caroline O Buckee. Quantifying the impact of human mobility on malaria. *Science*, 338 (6104):267–270, 2012.

[190] Will McKitterick. The messaging app report: How instant messaging can be monetized. Technical report, 2015.

[191] Susan P. Wyche, Sarita Yardi Schoenebeck, and Andrea Forte. "facebook is a luxury": An exploratory study of social media use in rural kenya. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 33–44, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1331-5. doi: 10.1145/2441776.2441783. URL `http://doi.acm.org/10.1145/2441776.2441783`.

[192] Liang Xiao, Jinliang Liu, Qiangda Li, and H Vincent Poor. Secure mobile crowdsensing game. In *Communications (ICC), 2015 IEEE International Conference on*, pages 7157–7162. IEEE, 2015.

[193] Yu Xiao, Pieter Simoens, Padmanabhan Pillai, Kiryong Ha, and Mahadev Satyanarayanan. Lowering the barriers to large-scale mobile crowdsensing. In *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, page 9. ACM, 2013.

[194] Liwen Xu, Xiaohong Hao, Nicholas D Lane, Xin Liu, and Thomas Moscibroda. More with less: Lowering user burden in mobile crowdsourcing through compressive sensing. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 659–670. ACM, 2015.

[195] Mengting Yan, Paul Castro, Perry Cheng, and Vatche Ishakian. Building a chatbot with serverless computing. In *Proceedings of the 1st International Workshop on Mashups of Things and APIs*, page 5. ACM, 2016.

[196] Dejun Yang, Guoliang Xue, Xi Fang, and Jian Tang. Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 173–184. ACM, 2012.

[197] Dingqi Yang, Daqing Zhang, Zhiyong Yu, and Zhiwen Yu. Fine-grained preference-aware location search leveraging crowdsourced digital footprints from lbsns. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 479–488. ACM, 2013.

[198] Yonglei Yao, Laurence T Yang, and Neal N Xiong. Anonymity-based privacy-preserving data reporting for participatory sensing. *IEEE Internet of Things Journal*, 2(5):381–390, 2015.

[199] Zhiwen Yu, Yunji Liang, Bukan Xu, Yue Yang, and Bin Guo. Towards a smart campus with mobile social networking. In *Internet of Things (iThings/CPSCom), 2011 international conference on and 4th international conference on cyber, physical and social computing*, pages 162–169. IEEE, 2011.

[200] Zhiwen Yu, Yun Feng, Huang Xu, and Xingshe Zhou. Recommending travel packages based on mobile crowdsourced data. *IEEE Communications Magazine*, 52(8):56–62, 2014.

[201] Willian Zamora, Carlos T Calafate, Juan-Carlos Cano, and Pietro Manzoni. A survey on smartphone-based crowdsensing solutions. *Mobile Information Systems*, 2016, 2016.

[202] Daqing Zhang, Leye Wang, Haoyi Xiong, and Bin Guo. 4w1h in mobile crowd sensing. *IEEE Communications Magazine*, 52(8):42–48, 2014.

[203] Xinglin Zhang, Zheng Yang, Wei Sun, Yunhao Liu, Shaohua Tang, Kai Xing, and Xufei Mao. Incentives for mobile crowd sensing: A survey. *IEEE Communications Surveys & Tutorials*, 18(1):54–67, 2016.

[204] Dong Zhao, Huadong Ma, Liang Liu, and Xiang-Yang Li. Opportunistic coverage for urban vehicular sensing. *Computer Communications*, 60:71–85, 2015.

[205] Vincent W Zheng, Yu Zheng, Xing Xie, and Qiang Yang. Towards mobile intelligence: Learning from gps history data for collaborative recommendation. *Artificial Intelligence*, 184:17–37, 2012.

[206] Yu Zheng, Hao Fu, X Xie, WY Ma, and Q Li. Geolife gps trajectory dataset-user guide, 2011.

[207] Yu Zheng, Lizhu Zhang, Zhengxin Ma, Xing Xie, and Wei-Ying Ma. Recommending friends and locations based on individual location history. *ACM Transactions on the Web (TWEB)*, 5(1):5, 2011.

[208] Yu Zheng, Furui Liu, and Hsun-Ping Hsieh. U-air: When urban air quality inference meets big data. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1436–1444. ACM, 2013.

[209] Gabe Zichermann and Christopher Cunningham. *Gamification by design: Implementing game mechanics in web and mobile apps.* " O'Reilly Media, Inc.", 2011.