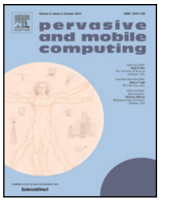
Pervasive and Mobile Computing 118 (2026) 102196

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc



A navigation framework for bicycle riders based on environmental and contextual factors☆,☆☆

Federico Montori [a,b,*], Rocco Pastore [a], Luca Sciullo [a,b], Luciano Bononi [a], Luca Bedogni [c]

[a] University of Bologna, Italy
[b] Advanced Research Center for Electronic Systems, Bologna, Italy
[c] University of Modena and Reggio Emilia, Italy

ARTICLE INFO

Keywords:
IoT
Mobile crowdsensing
Navigation system
Path planning

ABSTRACT

Urban scenarios present various concerns to micromobility users such as cyclists, which are an important part of the mobility infrastructure. As smarter cities increasingly focus on enhancing citizen well-being and infrastructure efficiency, navigation systems have primarily been designed for car drivers, with a focus on traffic conditions, while disregarding metrics that are important for micromobility users. As a matter of fact, they tend to privilege other aspects of their journey that enhance its comfort, rather than the mere path length. To address this gap, this paper presents a holistic architectural framework for micromobility-oriented navigation systems, incorporating nominal, data-driven and crowdsensed metrics in the loop. The paper presents the definition of metrics and a real world case study for each category of metrics, providing a real implementation. We demonstrate the effectiveness of our approach in a controlled environment before implementing and deploying the complete system in a real-world city, where we provide a detailed analysis of the results.

1. Introduction

In recent years, the rapid pace of urbanization has presented significant challenges for city planners and administrators worldwide. The increasing demands on resources, infrastructure, and services in urban areas call for innovative approaches to improve efficiency, sustainability, and overall quality of life for residents. Within this context, the Internet of Things (IoT) is now a well established key enabler for developing smarter cities [1]. By leveraging continuous feedback from sensors and systems, IoT facilitates the creation of an interconnected ecosystem that gathers and analyzes vast data volumes to optimize operations and support informed decision-making. The extensive use of sensor data has proven instrumental in prominent applications for smarter cities, including environmental monitoring [2], as well as mobility management [3] and traffic optimization [4]. Moreover, personal mobile devices such as smartphones have become so widespread that they are now an integral part of the IoT-enabled smart city fabric, by enabling pervasive and user-centric applications in smart cities.

The usage of smartphones in the everyday life aims to facilitate the life of users by offering data-powered services, often relying on the on-board sensors and the inferred individual context. For instance, Mobile navigation systems assist users in navigating the

☆ This article is part of a Special issue entitled: 'SmartComp 2024' published in Pervasive and Mobile Computing.
☆☆ A preliminary version of this paper has appeared in *IEEE SMARTCOMP 2024* (Montori et al., 2024).
* Corresponding author at: University of Bologna, Italy.
E-mail address: federico.montori2@unibo.it (F. Montori).

https://doi.org/10.1016/j.pmcj.2026.102196
Received 24 January 2025; Received in revised form 25 August 2025; Accepted 12 February 2026
Available online 16 February 2026

urban and extra-urban road network to get quickly to the destination, while taking into account additional real-time data such as traffic jams or road work [5]. However, current commercial navigation systems are mostly designed for automotive scenarios, while little attention is dedicated to other means of mobility, such as pedestrians or bicycle riders. It is clear that, for car drivers, the main goal is to reach their destination as quickly as possible. On the other hand, it might be very different for other users, who may privilege comfort, interest or fatigue. Bicycle riders may, for example, prefer to bike through road segments with less car traffic, a good road surface quality, served by a bike lane, with a good view, and so on so forth. Including such ancillary parameters would improve the navigation experience for such kinds of users, however the main challenges still reside in the different data sources that need to be integrated, while some of the data may not be readily available through an external API.

When data about certain phenomena of common interest is not available, the paradigm of Mobile Crowdsensing (MCS) offers an alternative to the deployment of a dedicated sensor network for monitoring applications [6]. MCS utilizes data generated by sensors embedded in the smartphones of a large group of participants to capture and describe phenomena at large. The geolocation data from these smartphones – typically collected via accelerometers, gyroscopes, magnetometers, microphones, and cameras – provides highly detailed insights with minimal cost. MCS finds great usage in many application fields, among which road navigation systems, where mobile data such as GPS and inertial sensor data is constantly gathered to estimate the amount of road traffic in certain zones [7]. This data is subsequently integrated into automotive navigation systems, enhancing their intelligence by recommending alternative routes that bypass traffic congestion, rather than solely focusing on the shortest path.

In this paper, we propose a complete architectural pipeline that features a road navigation system for micromobility users, such as bicycle riders. The system takes into account a number of metrics that concur into defining the user preferences when navigating a road network. This builds on the assumption that a micromobility user does not only take into account the route length, but a lot of other factors that make the journey more preferable, in case there is a valid alternative. Our proposal includes a backend system where the urban road network is represented as a weighted directed graph. The system, upon receiving the routing request of a user, performs a shortest path algorithm on the graph, modifying the weights of the edges according to a number of metrics. We define metrics mathematically and architecturally, and categorize them into three classes: nominal metrics, data-driven metrics and crowdsensed metrics. The system also features a mobile application that acts as the frontend client for users, as well as a data gatherer for crowdsensed metrics. The latter are powered by sensor data gathered by the smartphones and fed into our system, which in turn extracts the metrics from the raw data.

In summary, this paper tackles the following scientific challenges and proposes the following contributions:

1. **Existing solutions are customized on a few use cases and fail to provide a complete holistic solution**: We propose an entire architectural framework that outputs a mobile routing system based on graph navigation for micromobility users such as bicycle riders. The framework considers multiple metrics for recommending the best route and allows to add custom metrics by defining guidelines on how navigation metrics must be defined.
2. **Navigation metrics are customized leveraging the available data**: We define three classes of metrics from heterogeneous data and implement a real example for each one of them, highlighting the need for a diverse set of indicators.
3. **Crowdsensed proposal focus on a subset of the challenges of classic MCS systems:** For crowdsensed metrics, we provide a complete use case that features a full MCS pipeline, from data gathering on the smartphone, to data processing combining supervised and unsupervised classification techniques to compute the metric values at runtime.
4. **Solutions are often tested on small scale dataset and do not consider challenges related to their practical implementation**: We implement and deploy the system, including the mobile application, and run it in a public environment to gather sensor data and perform a number of navigation tests that prove its efficiency and how it behaves in a real city.

The rest of the paper is structured as follows: Section 2 outlines the related works in the area; Section 3 illustrates the system model and the architecture; Section 4 describes more in detail, respectively, nominal, data-driven and crowdsensed metrics, providing a case study for each of them; Section 5 describes the implementation of the frontend; Section 6 presents the results acquired with the deployment of the system, and finally Section 7 concludes the paper.

2. Related works

Data obtained from sensors and mobile devices has been used to monitor a plethora of different parameters of interest in cities and wider areas to impact navigation and recommend personalized paths. This data can be of any type, ranging from environmental data to avoid polluted areas, to proposals that optimize traveling through certain zones such as parks [8]. Other works focus instead on optimizing multiple objectives at the same time and finding paths on roads that travel through a set of defined points by optimizing more than one parameter [9]. Data obtained from campaigns can also be used to optimize the placement of Points of Interests in a city, like [10] does with bike sharing services.

There is also great interest for MCS data such as the monitoring of road surface quality, in which the idea is to use vehicles that travel along road segments to monitor their features [11]. The methodology is to use inertial sensors in vehicles or mobile devices and analyze those to detect potholes, and rough surfaces, similar to what authors in [12] do. They use dedicated MEMS accelerometers, leveraged to identify potholes on roads, cracks on the tarmac as well as different road types, which can be detected through the analysis of the inertial sensors, such as dirt, rough and similar. They validate their work using a real-world dataset, although they do not propose their work as a continuous monitoring, but rather on a one-shot recognition.

A similar approach is also proposed in [13], where the authors use smartphones with an application to label roads with the perceived quality, derived from the number of potholes and how many times the driver has to brake. As the authors state, potholes

and braking events are two of the main concerns for drivers on roads, which translate to poor road quality. A similar contribution can also be found on [14], where the authors leverage smartphone accelerometers and collect a large quantity of data to classify tarmac conditions in different scenarios. They collected their dataset in four different Brazilian cities using smartphones and driving a car, thus highlighting the validity of the study in diverse conditions. However, their work is more focused on determining the road type rather than the road condition or the road quality.

In [15], authors utilized a smartphone application on multiple devices to monitor road pavement conditions using sensors. Accelerometers detected a set of anomalies, while the GPS tracked the vehicle location and distress positions. Test vehicles traveled at speeds between 25 km/h to 40 km/h, with three accelerometers installed to explore gauge measurement accuracy. Their analysis detects indeed potholes but does not provide consistent performance in identifying their location. Also [16] considered a similar use case, and they identify vertical vibrations as one of the main parameters to monitor to determine the road quality using smartphones, likewise [17] where the authors also consider the frequency and wavelet domains to extract features from the inertial sensor data. Statistical data is also the proposal of [18], still using smartphone inertial data. A slightly different approach is the one proposed in [19], which proposes a deep neural network using video data and braking friction, used together to identify road quality. This has been extended also to use images at night in [20], significantly reducing the false negatives and positives ratio. A similar approach is also presented in [21], studied on Indian highways, where they introduce a method to automatically evaluate potholes, cracks, and patches. Their proposal works, but it may struggle if other objects are seen in the video.

Most of the works we have discussed leverage cars to perform the monitoring. Using bicycles, which is the focus of this work, would require to validate those studies again. One of the first works can be seen in [22], however the class labeling was mostly empirical. More recently in [23] the authors leverage ultrasonic sensors to detect small obstacles on the road, while a significant step forward is presented in [24], as the authors use inertial data and the GPS to determine the International Roughness Index (IRI) of the road. While the IRI has been designed for cars, it has recently proposed an extension also for bicycles [25]. Another proposal comes from [26], where the authors show the added benefits of having multiple sensors on the bicycle. However, this solution is not viable for MCS, as there are too many sensors which is impractical for users. Finally, we mention [27], in which, again, the road condition is monitored through inertial sensors, however, it is done via users who have to purposely pass through these areas to detect them, a scenario which is generally avoided by cyclists.

A preliminary version of the platform proposed in this paper was published in [28], where we designed an MCS system to assess the road quality for cyclists as a discriminant for their navigation. With respect to the state of the art, our solution does not need previously collected data, instead it leverages MCS to perform a "cold start" phase in which the metric is evaluated only once enough data is collected by the users.

Most of the works focusing on these challenges are vertical on the use case and present isolated contributions, and determine the detection and labeling of road conditions; this drawback also affects our past work [28], which only focuses on road quality. However, they do not consider the architecture needed to realize such a vision, which however is a key point to address. This paper progresses in that direction, by providing a reference architecture to integrate all the above contributions into a wider, holistic approach. This is also considered at the data level, in which we leverage data from multiple sources to homogenize and integrate them together. Moreover, we show the end to end software components needed to realize such a vision, and we evaluate our contribution on a variety of different scenarios.

3. System model

This section presents the architecture of the entire system, designed to provide bicycle riders with a smartphone-based route planner. The planner considers not only the fastest route but also various other qualitative metrics, enabling riders to choose alternative routes that, while longer in distance, prioritize comfort and safety. A metric can be, for example, the presence of noise pollution, a good view or the presence of traffic. In this section, these metrics are presented abstractly, with no reference to our implementation, stressing the fact that the system is not designed to work with specific metrics, but can be adapted to any user-defined metric that satisfies simple mathematical properties. Metrics are defined in Section 3.1, while how they are used in the path calculation is outlined in Section 3.2.

The proposed system consists of several components working together to achieve its ultimate goal. Fig. 1 illustrates the complete architecture, detailing the logical components from data acquisition to the final route planning. Let us begin by describing the individual components:

- The central component is the **App Server (AS)** where most of the data processing takes place. The AS stores the road graph of the city, where nodes are intersections and edges are road segments, and gets queried for routes by frontend clients. Each query contains a departure point and an arrival point, together with the preferences of the user against different metrics. The AS then replies to the query with the best route that balances the preferences of the user with the actual length of the path.
- Data about each and every road segment is stored onto a **Data Repository (DR)**, remotely accessible by every actor in the system through an API. The data in the DR is used by the AS to populate the road graph with weights according to the calculated metrics. The DR is here presented as a logically separated module, however it may or may not be physically separated from the AS.
- Complying with the separation of concerns philosophy, navigation metrics are calculated by external actors, called **Metric Calculators (MC)**. They are committed to gather the necessary data and inject the metric value for each road segment in the DR. Data may or may not be extracted from the DR, in fact, in certain cases, an MC may rely on external data sources. The AS is in fact metric-agnostic, i.e., it does not know how a metric is calculated and what it semantically means. This way, there is no backward-dependency.

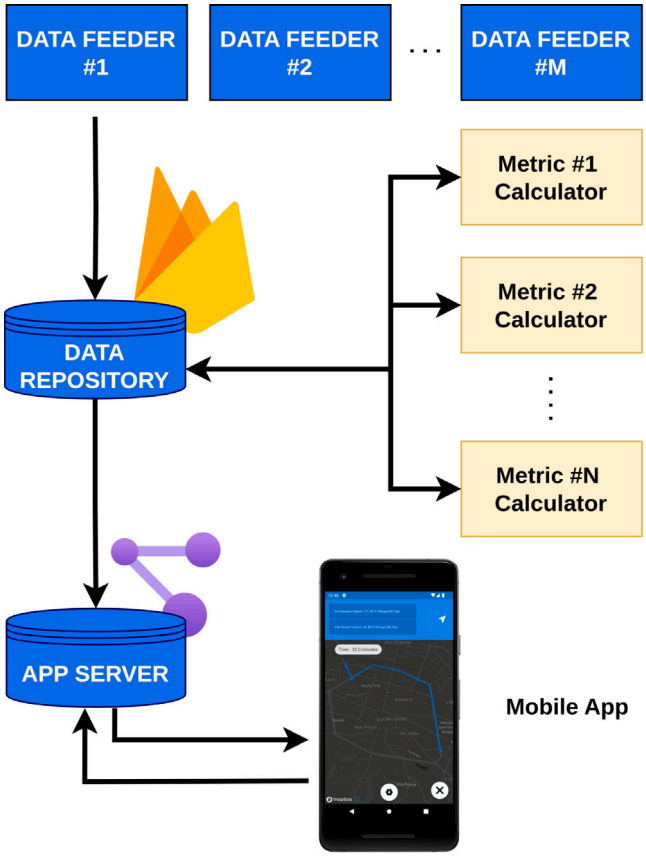


Fig. 1. High-level architecture of the system.

- The final user interacts with the system through a **Mobile App (MA)**. The MA is the client frontend and has the function of a route planner, with which the final user can get directions to a destination, by querying directly the AS. The suggested result will account both for the route length and the metrics preferences that the user sets on the MA itself.
- Data in the DR is fed by one or more **Data Feeder (DF)**. DFs are external actors that inject data in the DR, used for calculating certain data-driven metrics that necessitate constant data updates over time. One example may be the noise pollution, which consistently varies over time and, in absence of an external service that retains the necessary data, needs a dedicated DF. A DF can be different on top of the type of data and the update frequency needed. For example, in case the metric is based on data that is available online (e.g. the weather), then the DF may be a simple external component that scrapes the weather service and injects data in the DR. If the metric is crowdsensed, i.e. collected by the same end users of our platform, then the MA itself may additionally function as a DF.

3.1. Metrics

The proposed system has the goal of outputting a suggested route for the user according to his or her preferences. Because a cyclist may have different discriminant than a car driver, who is likely interested in taking the shortest path, we need here to take into account different metrics and combine them in the route planning algorithm. In this section we define the **navigation metrics**[1] that may affect the choices of a user in taking a slightly longer path.

A metric is a feature that numerically characterizes a semantic attribute of a road segment. Metrics can be nominal or dependent on data, and we assume that the road network of a closed scenario (e.g. a city) is defined on top of a certain number of these metrics. We logically distinguish metrics in the following sub categories:

- **Nominal metrics:** represent attributes of a road segment that do not change over time and are universally known. Nominal metrics can easily be extracted from an external service and do not require monitoring. Examples of these metrics are: the

[1] In this paper, we use the generic term “metric” to refer to what we define as navigation metric here.

road type (residential street, cycle path, motorway, etc.), the presence or absence of a cycle lane, the presence or absence of a landscape, the uphill/downhill incline, and many others. Some of these may change over time, but the change must be exceptional and due to extreme events such as intense road work that change the topology of the road segment itself. Nominal metrics represent attributes of a subject that remain constant over time and can be universally recognized. Once collected, these metrics can be easily retrieved from external services without the need for constant monitoring. Examples can be the type of road (residential street, cycle path, motorway, etc.), the presence of a cycle lane or a landscape, or the uphill/downhill incline. Changes to these attributes are rare but may occur in exceptional cases, such as major roadworks that alter the topology of the road segment. In this case, since the process of re-mapping the road may require a long time, wrong data provided by the external service could affect the right functionalities of our solution. Nevertheless, it is clear that the effects of such a rare situation could impact the final path for very few cases and/or very few distances.

- **Data-driven metrics:** represent attributes that depend on external data, retrievable from external services. These are expected to vary over time, therefore require a constant monitoring. Examples of these metrics are environmental conditions such as the temperature and the air quality, which can be retrieved from local weather stations. Traffic conditions fall within this category as well. In these case, the MC that takes care of a data-driven metric must periodically poll the external service (or subscribe for data updates) and consistently report the changes on the DR with the desired rate. In Section 3.1 we introduced data-driven metrics as attributes that rely on external data sources and are expected to change over time, requiring continuous monitoring. Examples include environmental conditions such as temperature and air quality, obtained from local weather stations, as well as traffic conditions. This type of data significantly impacts the overall quality of a cyclist's journey.
- **MCS metrics:** represent attributes that still depend on data, but for which public data is not available. In these cases, the user themselves act as data gatherers using a number of mobile sensors. With respect to the architecture in Fig. 1, users physically carry around a DF, which may be embedded in the same client MA, or be a separate device. In any case, the DF physically collects raw sensor data on the move, and stores it onto the DR. The dedicated MC then periodically outputs the value of the metric concerning the road segments by elaborating such raw data. Examples of MCS metrics may be the noise pollution, for which mobile phones can gather time-dependent data through the smartphone's microphone. Another example is the road surface quality, which is estimated via the inertial sensors embedded in the smartphones.
  Crowdsensed Navigation Metrics leverage the well-known Mobile Crowdsensing (MCS) paradigm to gather data on top of which computing the metrics. In other words, such data is not available from an external source, but it has to be actively gathered within the scope of the framework itself. With respect to Fig. 1, an MC that concerns one of these metrics, must rely on data that is injected into the DR by a dedicated DF. We can technically state that *a DF is necessary for each crowdsensed metric* or that *a crowdsensed metric relies on a single DF*. In most cases, we envision the DF to be a module of the MA, so that the framework final users are the ones who also opportunistically populate the DR with the data they need.
  An example of a crowdsensed metric could be the noise pollution, which is an environmental parameter for which there hardly is open data about. In our past work [29] we designed a crowdsensing framework that not only gathers data about noise pollution using the microphone embedded in smartphones, but also performs an inference for areas that are uncovered by data, or where data is too old.

In this paper we will provide an in-depth example of how we integrated a metric for each of these classes in our system, respectively in Sections 4.1, 4.2, and 4.3.

3.2. Graph construction

The AS refers to an enclosed geographical area (e.g., a city) and stores locally its road network. The road network is represented through a weighted graph $G = (V, E)$, where edges $e_i \in E$ represent uninterrupted road segments and vertices $v_j \in V$ represent road intersections. Each vertex v_j is represented by a tuple $v_j = \langle id, lat, lon \rangle$, where lat and lon are, respectively, the latitude and the longitude of the vertex. The AS defines a number of metrics $m_u \in M$, which means that there will be exactly $|M|$ MCs, one for each metric. Each edge e_i is then defined as a tuple $e_i = \langle id, src_i, dest_i, len_i, w_{i,1}, w_{i,2}, \dots, w_{i,|M|} \rangle$, where $src_i \in V$ and $dest_i \in V$ are, respectively the source and destination vertices of the edge, $len \in \mathbb{N}$ is the length in meters of the road segment, and $w_{i,u} \in [0, \omega]$ is the value of the ith edge concerning the uth metric.

The values of metrics are mapped in a fixed interval $[0, \omega]$ so that it is possible to combine and compare them. The owner of the system may statically set ω at system setup, to universally give more or less weight to the metrics as opposed to the length (which is unconstrained). Some of the metrics may not natively be expressed in numerical form, in such case the system designer must somehow translate the categorical values in numbers so that a value close to 0 represents a category mostly desired by cyclists, a value close to ω the opposite.

At runtime, a user may, through the MA, query the AS for a route. The query is represented by a tuple of parameters $Q = \langle src, dest, \alpha_0, \alpha_1, \alpha_2, \dots, \alpha_{|M|} \rangle$, where src and $dest$ are the source and the destination of the route, and $\alpha_1, \alpha_2, \dots, \alpha_{|M|}$ are the preference values that the user attributed to the various metrics, while α_0 is the preference value that the user attributed to the path length. Preference values are defined such that $\forall z . \alpha_z \in [0, 1]$ and $\sum_{z=0}^{|M|} \alpha_z = 1$. Consequently, the AS replaces the weight of each edge with the following convex combination:

$$W(e_i) = \alpha_0 len_i + \alpha_1 w_{i,1} + \alpha_2 w_{i,2} + \cdots + \alpha_{|M|} w_{i,|M|}$$

Subsequently, the AS performs a shortest path algorithm, such as Dijkstra or A*, on the updated graph. Note that the route in output is not necessarily the shortest, as the various metrics affect the result according to the preference values. As a corner case, if $\alpha_0 = 1$ and all other preference values are set to 0, then the algorithm returns the *actual* shortest path.

When the user requires a route plan to the AS through the MA, the AS checks if it contains in memory both the source and destination vertices that correspond to the precise coordinates inserted by the user. In case there is no exact match, the AS returns the closest vertices respectively to the source and destination and uses them for the shortest path computation.

4. Navigation metrics implementation

In this section we provide an implementation of three metrics, one per type as defined in Section 3.1. These implementations are just examples that can be replaced on top of the use case.

4.1. Nominal navigation metrics

IN ourt implementation, we collected nominal navigation metrics from OpenStreetMap (OSM),[2] focusing on the road type, that belongs to one of the following: “primary”,“tertiary”, “path”, “footway”, “living street”, “service”, “residential”, “pedestrian”, and “cycleway”. We decided to focus on such nominal metrics since our navigation framework is targeted for cyclists, hence the road type can play an important role when choosing the path, but we highlight that this represents an example and additional or different nominal metrics could be easily integrated in our solution. We assign a score to each street, as detailed in Section 3.2, to reflect cyclists’ preferences based on road type. According to the OSM documentation, a “primary” road refers to a national road, while a “tertiary” road is a busy non-national road, often marked with dashed white lines in the center. A “path” represents a generic route used by pedestrians, whereas a “footway” specifically maps minor pathways primarily or exclusively for pedestrian use. The term “living street” denotes roads where pedestrians have priority over vehicles, as defined by traffic rules. “Service” typically describes auxiliary or access roads, while “residential” applies to roads serving residential areas. The “pedestrian” designation marks roads or areas intended mainly or exclusively for pedestrians, and “cycleway” is used for dedicated cycling paths. Examples of such road types are depicted at https://wiki.openstreetmap.org/wiki/Key:highway.

From a cyclist’s perspective, these road types vary significantly in terms of safety and accessibility. For example, “primary” roads are more dangerous compared to “tertiary” roads, while certain types, such as “pedestrian”, may be off-limits. To account for these factors, the scoring system evaluates and incorporates these characteristics for each road type. In order to collect this information, we built a custom application written in `Node.js` designed to manage the entire process of downloading, analyzing, and processing OSM data. We used `Nest.js` framework for its modular and scalable architecture, and we integrated libraries such as `osmtogeojson`[3] for converting data into geoJSON format and `osm-read`[4] for parsing raw OSM XML data efficiently.

4.2. Data-driven navigation metrics

In our implementation, we incorporate air quality as a factor when calculating different routes. Consequently, some paths may offer better air quality but might come at the cost of increased travel distance. Similarly to how road type is used as a nominal navigation metric, air quality is just an example of the usage of data-driven metrics; our system is designed to be flexible, allowing for the seamless integration of additional metrics to enhance the navigation experience. The air quality values are obtained through the APIs provided by the *OpenData* section of the Bologna municipality’s website.[5] These APIs are accessed using a dedicated module integrated into the `Node.js` application introduced in Section 4.1, which manages data acquisition and preprocessing. The retrieved data pertains to the air quality for the current year, as measured by three monitoring stations located within the Bologna municipality.

Among the various air pollutants available, PM10 (particulate matter with a diameter of 10 μm or less) was selected as a representative metric due to its significant impact on health and its widespread use in environmental studies. The data originates from ARPA Emilia Romagna,[6] the regional environmental protection agency responsible for collecting and publishing environmental data.

The three monitoring stations, shown in Fig. 2, are strategically located in different areas of the city to provide comprehensive coverage of air quality variations. Specifically, they are situated in the following locations: (i) Giardini Margherita, a major public park in the city, (ii) Via Chiarini, a urban residential area, (iii) Porta San Felice, a location near a historic city gate with significant traffic influence.

[2] https://www.openstreetmap.org/
[3] https://github.com/tyrasd/osmtogeojson
[4] https://github.com/marook/osm-read
[5] https://opendata.comune.bologna.it/explore/dataset/centraline-qualita-aria/custom/?disjunctive.agente_atm
[6] https://www.arpae.it/it

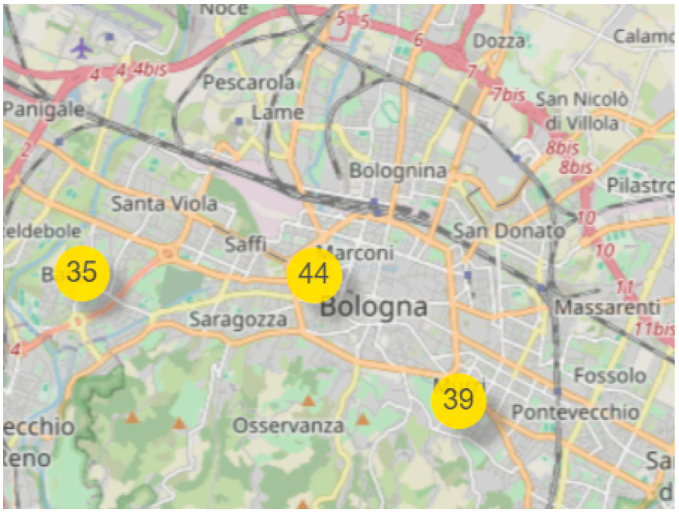


Fig. 2. Position of the monitoring stations in the city of Bologna, displaying the current value of PM10. ©OpenStreetMap.

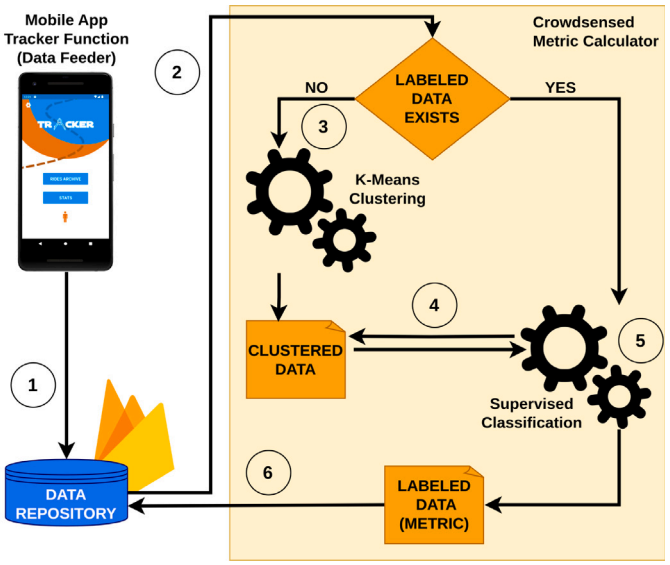


Fig. 3. Architecture of the road surface quality crowdsensed metric.

4.3. Crowdsensed navigation metrics

In our implementation, we developed an example crowdsensed metric that aims to assess the surface quality of the road segments. Logically, cyclists are less keen to take paths where the surface quality is poor, which typically means bumpy, dirt and not well-maintained roads. There are also cases of roads that are intentionally built of cobblestone in city centers, because of ornamental or historical reasons, however they are not well suited for road bicycles. Upon these premises, we implemented a metric calculator that is built on top of a DF which gathers vibration data (accelerometer and gyroscope) and injects it into the DR. The MC then extracts the raw data and classifies it into road quality classes, which determine the value attributed to the metric.

Fig. 3 illustrates that the execution flow can be logically separated into two primary phases: data gathering and classification, followed then by route calculation. The effectiveness of the second component depends on the success of the first. This section is divided into several parts: in Section 4.3.1 we present the entire workflow that concerns data gathering, initial cold start elaboration for tuning the ML algorithm, and data classification. In our implementation we performed an initial experiment in a controlled environment to understand whether our method was correct. We first gathered data from a small town (the data gathering process is explained in Section 4.3.2, then we conducted a series of experiments to find a significant number of classes for our algorithm, explained in Section 4.3.3.

4.3.1. Workflow

Below, we outline the logical steps that guide the system toward its ultimate goal. The numbers in the following list correspond to those depicted in the figure:

1. The MA records, using its "tracker" function (which acts as a DF in this case), the accelerometer and gyroscope data in the background, only when the user is actually biking (using the Activity Recognition framework offered by Android). Data is locally converted into chunks and geolocalized, then it is sent to the DR.
2. Fresh data obtained from users is unlabeled. As road quality can be subjective and really hard to unambiguously assess by humans, users should not be able to label their data, instead, we rely on a two-step ML approach, which makes up most of the MC in the yellow box in Fig. 1. Either way, the procedure starts by checking whether labeled data is not present at all in the DR; in other words, if this is a "cold start" and no classification has ever taken place before.
3. In there is no labeled data available in the DR, we find ourselves in a "cold start" phase. A cold start only takes place once and finishes when enough data has been gathered and labeled. To address this cold start phase, our approach employs an unsupervised learning algorithm on the unlabeled data. Specifically, we use the *K-means* algorithm, allowing us to control the number of clusters generated. We expect that data points within the same cluster correspond to the same road quality class (this assumption is validated in Section 4.3.3). It is crucial to carefully manage this cold start process, as insufficiently expressive data (*i.e.*, data that does not adequately represent different scenarios) could negatively impact subsequent classifications. *Therefore, this step should only be executed once a significant portion of the city has been mapped with raw data and the data exhibits sufficient heterogeneity*.
4. After clustering a sufficient number of data points into road quality classes, we use these labels to train a supervised machine learning model, such as Random Forest (RF) or Support Vector Machines (SVM). This trained model will then be utilized to classify any new unlabeled data points in the future. This phase finalizes the cold start and brings the system to a steady state. This process takes place on the DR, therefore new users may take advantage of an already trained model without the need for further configuration.
5. If labeled data is available, *i.e.*, if the cold start process has already been completed once, the supervised learning model is used to classify new data points. Optionally, these classified data points can be used to further train the model, provided the confidence of the model is higher than a predefined threshold.
6. Regardless of the chosen approach, the labeled data is converted to metric values and stored back in the DR.

4.3.2. Data gathering with the data feeder

The data gathering process occurs in two distinct phases within our system. Initially, there is a preliminary data gathering phase to populate the repository with sufficient data points, enabling the clustering algorithm to produce meaningful results. Subsequently, the data gathering process continues beyond the cold start, operating concurrently with the system in its steady state.

In both phases, the process is managed by the MA through its *tracker function*, which serves as the DF. This function can automatically determine whether the user is walking, driving a vehicle, or riding a bicycle. Notably, data gathering is only activated during the latter activity. Additional implementation details are provided in Section 5.

In our data gathering process, the user secures the phone in a holder mounted on the bicycle's handlebar. Before using the application, the user is required to input some basic information, such as their name and the smartphone model. This data may prove useful later for evaluating differences between smartphone models or conducting studies to identify riders and detect outliers that might compromise the data collection process.

While riding, the DF opportunistically records the following sensor data:

- **Accelerometer** (all three axes).
- **Gyroscope** (all three axes).
- **Magnetometer** (all three axes).
- **Linear Accelerometer** (all three axes): a software-based sensor derived by combining accelerometer and magnetometer readings to eliminate the gravitational component from the accelerometer signal.

Each sensor reading is accompanied by its corresponding GPS location and a timestamp.

With this information, once the ride is completed, the DF extracts data points from the raw data. Initially, all data records containing null values are removed. Then, the entire ride is divided into time windows of 3 s duration to calculate the features. This duration is widely accepted in the literature for similar tasks [30].

For all sensor readings within a time window, their magnitude is first calculated. Subsequently, for each sensor type, the following features are computed: mean, variance, standard deviation, minimum, maximum, range, skewness, and kurtosis. As a result, each data point in the dataset corresponds to a 3 s time window, which is represented by a single GPS point (the centroid of all sensor readings) and comprises 32 features (8 features per sensor type).

During the preliminary data gathering phase, before clustering occurs, we ensure that enough data is collected to represent a significant portion of the city, covering all possible road types. Once this is achieved, the resulting dataset is pre-processed by scaling the feature values of all data points using a standard scaler. This same scaler, without any modifications, will then be used and applied to all subsequent data gathering activities following the clustering process.

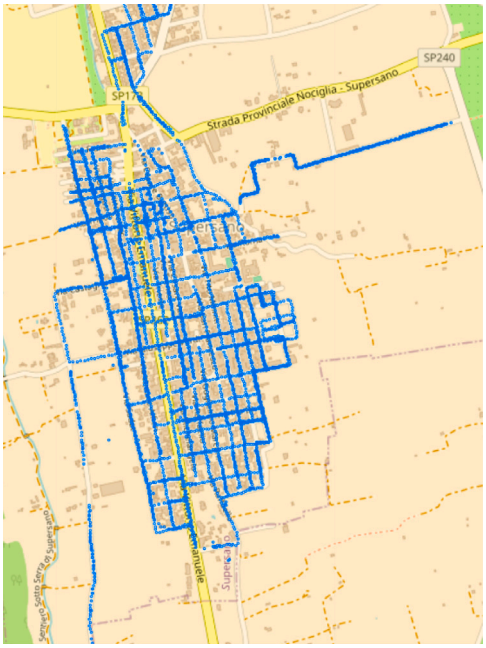


Fig. 4. The map of the town of Supersano, in Apulia, showing all data points gathered during our controlled tests.

4.3.3. Finding the classes through clustering

It is crucial to emphasize that data points collected by users are not labeled by the users themselves. In fact, attributing a "quality class" to the road surface is challenging for a human observer because (i) it is highly subjective, and (ii) the threshold that distinguishes two classes can be difficult to perceive. For this reason, we rely on a clustering algorithm to generate road quality classes based on the gathered data. However, there is no guarantee that running a clustering algorithm without any background information will accurately reflect road quality, as it may be influenced by other criteria.

To address these uncertainties, we conducted a controlled test over a small area that includes various types of road surfaces. Specifically, we carried out a data gathering process in Supersano, a small town in southern Italy, where we were able to cover the entire area. Fig. 4 shows a map of the region along with all the collected data points. The resulting dataset consists of 6500 data records, which we manually annotated. During the data collection, we also ensured that the smartphone recorded a video of the entire ride, allowing us to manually label the data afterward.[7] Given the reasons mentioned, we do not consider this manual labeling as a ground truth, but rather as a way to assess whether the clustering results align with our expectations. Specifically, our manual labeling distinguishes only *two* classes: good and bad road pavement quality, as we believe these categories are more easily understandable from a human perspective. It is important to recall that the controlled test reported in this section was performed by us to prove that our data-driven method is significant for assessing road quality, i.e. that performing a K-means clustering effectively produces clusters related to different road quality. In a real world scenario, this controlled test does not need to be performed as it is a metric-specific proof. In fact, it is not present in the workflow in Fig. 3.

Binary validation. As an initial step to evaluate the effectiveness of our manual annotation, we conduct a supervised validation experiment on the Supersano dataset. This experiment follows a standard supervised learning approach, where we split the dataset into training (70%) and test (30%) sets, and apply a basic classification algorithm to classify the test set examples into the two annotated classes. Specifically, we tested Random Forest (RF), Stochastic Gradient Descent (SGD), and Support Vector Machines (SVM) on our dataset, with all three models achieving accuracy scores between 0.97 and 0.99. Given that the two classes are fairly balanced in this dataset, we consider this as preliminary evidence that the collected features are reasonably indicative of road quality.

Next, we performed a second validation step by applying an unsupervised clustering algorithm, specifically K-means, to the Supersano dataset. We instructed K-means to identify two clusters and compared the results with our manual annotation. The outcome, shown in Fig. 5, presents the distribution of data points from each of the two annotated classes within each cluster. The results reveal that each cluster contains more than 90% of data points from a single annotated class. This suggests that a simple K-means clustering can effectively reflect road surface quality, rather than being influenced by other factors.

[7] The full details regarding the data collection methodology, the dataset's characteristics, and its usage in the experiments are comprehensively documented in the referenced thesis [31].

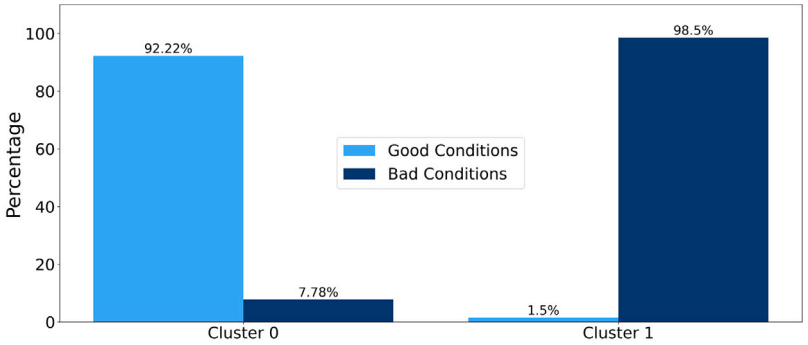


Fig. 5. Results of K-means with two clusters. The figure shows the distribution of each cluster over the pre-annotated dataset.

Finding the number of clusters. After confirming that running K-means on a sufficiently comprehensive dataset reflects road surface quality, the next step is determining a meaningful number of clusters. These clusters should be numerous enough to introduce variety, while ensuring that the differences between clusters are statistically significant.

To achieve this, we first performed a *One-Way ANOVA F-Test* to evaluate (i) the significance and discriminative power of each feature in relation to our classification goal, and (ii) the number of clusters that can be identified while maintaining statistically observable discrimination across a sufficient number of features.

The F-Test is applied to the entire dataset for each feature using the following formula:

$$F = \frac{\sum_{i=1}^{K} n_i(\overline{Y}_i - \overline{Y})^2/(K-1)}{\sum_{i=1}^{K}\sum_{j=1}^{n_i}(Y_{ij} - \overline{Y}_i)^2/(N-K)} \tag{1}$$

Eq. (1) computes an F-value F for each feature. The numerator represents the "between-group variability" and measures how much the feature values differ across the entire dataset, normalized by the number of clusters. In this context, K is the number of clusters, and n_i denotes the number of examples in the ith cluster. $\overline{Y}_i$ is the mean value of the feature in the ith cluster, while $\overline{Y}$ is the overall mean of the feature across the dataset. The denominator represents the "within-group variability", where Y_{ij} is the feature value of the jth data point in the ith cluster, and N is the total sample size.

For each feature, this statistical test yields a high F-value if the feature values vary consistently across the dataset but not within individual clusters, indicating that the feature is significant for classification. Conversely, a low F-value suggests that the feature is likely insignificant.

We conducted the F-test over the entire dataset, repeating the experiment for a range of cluster numbers from 2 to 10. The results are presented in Fig. 6, where the features are shown on the x-axis and their corresponding F-values for different cluster counts are displayed on the y-axis. The features are ranked by their average F-value, and split into two figures for display purposes. As anticipated, we observe that as the number of clusters increases, the F-score tends to decrease, eventually stabilizing at a plateau for more than 5 clusters.

We decided to perform additional validation experiments for 2, 3, and 4 clusters. Specifically, we used the results from K-means to re-label our Supersano dataset with 2, 3, and 4 classes and conducted another supervised test, similar to the one described in . In this process, we first applied K-means to label the entire dataset, then split it into training and test sets. Afterward, we ran K-means again on the training set and re-labeled it by matching the clusters to the previous labeling. This step ensures that the test set does not influence the classification process. For all three clustering configurations, the experiment produced accuracy values ranging from 0.96 to 0.99, which further reinforced our confidence in this approach.

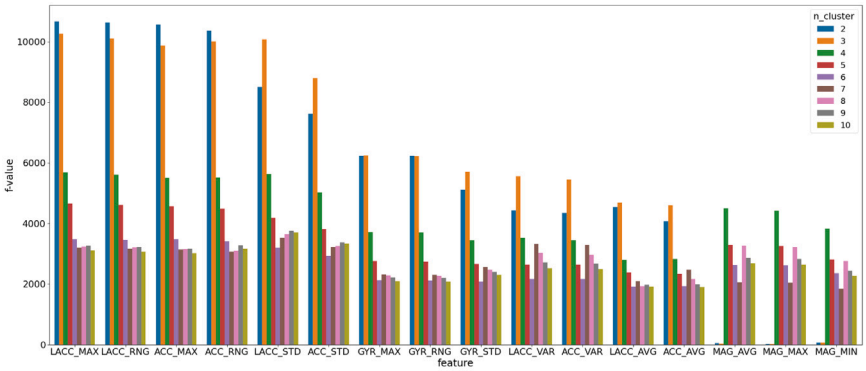
We then decided to stick to three classes for all experiments in our evaluation (Section 6), to balance the confidence of the unsupervised classification with a meaningful human perception. Samples of the three classes in the real world are shown in Fig. 7.

5. Frontend implementation and deployment

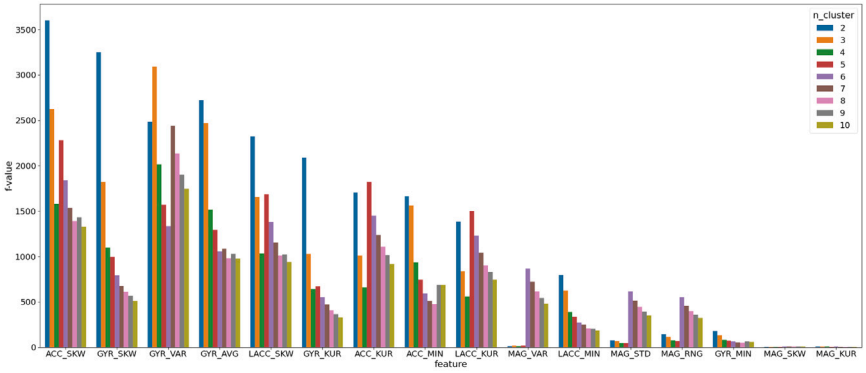
In order to test the usability of the system in the real world, we implemented and tested the end user application as a native Android app. The application we developed serves two main purposes: first, there is the tracking view, in which the application collects data while the user is on a bicycle (this serves the purpose of a DF), while the other is related to the mobile navigation client, through which the user can input starting and ending position, as well as the various values of α_j, and the application shows the ideal road taking into account the road condition (this serves the purpose of a MA).

5.1. Tracking view

The tracking view collects data about the accelerometer, the gyroscope, and the magnetometer, along with the position of the user and the timestamp. The data collection is performed only when the user is riding a bicycle, which is possible thanks to the



(a) 16 features with the highest F-value.



(b) 16 features with the lowest F-value.

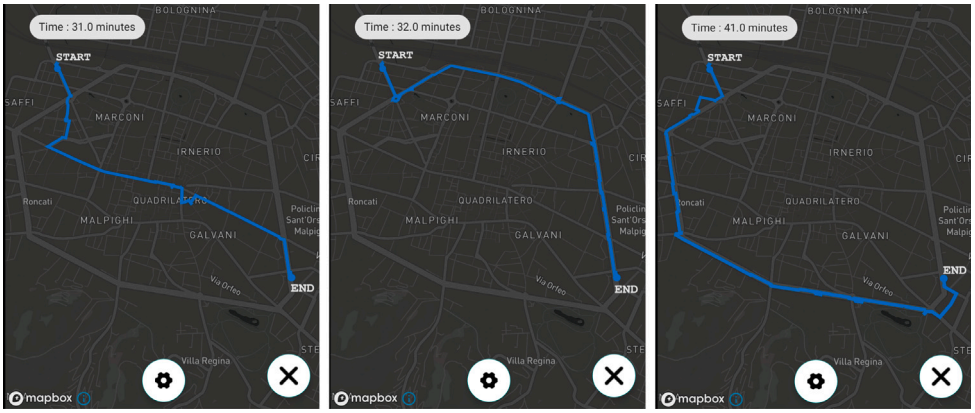
Fig. 6. ANOVA F-values for each number of clusters in K-means of all 32 features with the highest F-value.

(a) Sample of cluster 1 (b) Sample of cluster 2 (c) Sample of cluster 3

Fig. 7. Samples of road quality outputted by K-means with 3 clusters, respectively accounting for good, fair, and bad quality.

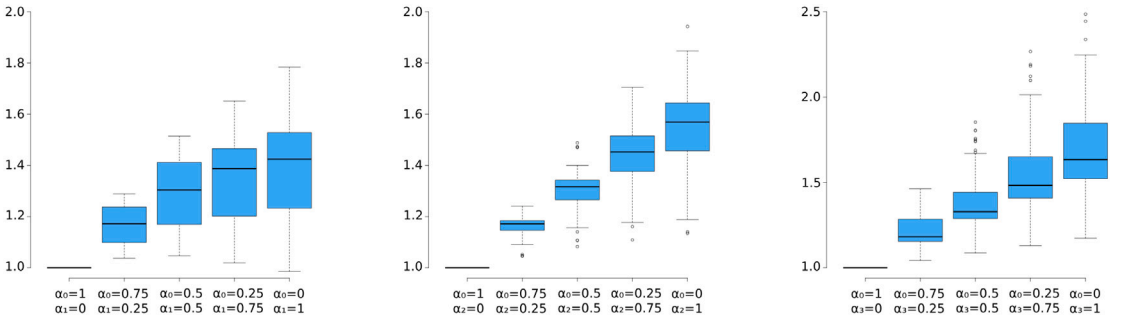
Android Activity Recognition API.[8] To strengthen the reliability of the dataset, the application can also record video footage, to allow later manual labeling of the road segments.

[8] https://developer.android.com/develop/sensors-and-location/location/transitions



(a) $\alpha = 0$ (b) $\alpha = 0.5$ (c) $\alpha = 1$

Fig. 8. Different routes proposed by the Navigation System from the same starting point and to the same destination.



(a) Road Type (α_0 vs. α_1) (b) Air Quality (α_0 vs. α_2) (c) Surface Quality (α_0 vs. α_2)

Fig. 9. The route length proposed by the system L_{fin} expressed in ratio over the shortest path length L_{in}, for the threes single metrics, aggregated for different preference values.

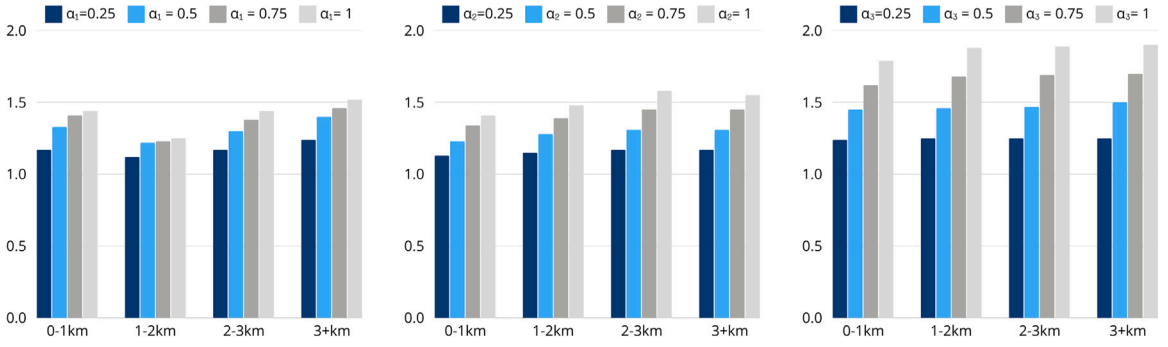
5.2. Navigation view

In the navigation view, users can leverage the already collected data to input starting and ending points and retrieve the optimized path to travel. The view is built with the Mapbox framework with its Navigation SDK[9] for mobile applications and also provides a setting customization view in which the user can select her/his preferred α_j value for any considered metric.

Upon requesting a new route, the data is handled from the MA to the AS, which obtains the information and runs the routing algorithm. Given that start and destination points may not be placed on a node, the system picks the closest nodes to the two points instead. The AS returns the sequence of graph vertices with their coordinates, upon which the MA runs the Mapbox Direction APIs[10] to get a feasible path on the local map that goes through all the points provided.

In Fig. 8 we show an example of the navigation view, in which the starting and ending points are the same. However, we vary the user preferences (in this case, the parameter concerning the road surface quality), and the three proposed paths change significantly. In the first example, the system prefers to optimize the total travel time, hence it provides the fastest path without considering the road conditions. In downtown Bologna, the road pavement is very rough, with many streets having basalt or cobblestone floors, hence not an ideal solution for cyclists. The second case it balances the parameters, and provides an intermediate solution, while the latter example optimizes the road condition, although providing a considerably longer path. The code of the app, together with the AS is released open source.[11] A version of the AS including the calculation of all metrics is also released open source.[12]

[9] https://docs.mapbox.com/android/navigation/guides/
[10] https://docs.mapbox.com/api/navigation/directions/
[11] https://github.com/stradivarius/RoadSurfaceBasedRouting4Cyclists
[12] https://github.com/stradivarius/MetricCalculator4Cyclists_AS



(a) Road Type (b) Air Quality (c) Surface Quality

Fig. 10. The route length proposed by the system L_{fin} expressed in ratio over the shortest path length L_{in} for different values of L_{in}.

6. Navigation results

In this section, we report the results of an implementation of our system, which was evaluated in a selected urban area. In our implementation, we implemented one metric for each category, in particular:

- As a nominal metric, we used the **road type**, as described in Section 4.1. We hereafter name the metric value $w_{i,1}$ for the ith edge.
- As a data-driven metric, we used the **air quality**, as described in Section 4.2. We hereafter name the metric value $w_{i,2}$ for the ith edge.
- As a crowdsensed metric, we used the **surface quality**, as described in Section 4.3 and in [28]. We hereafter name the metric value $w_{i,3}$ for the ith edge.

The weighted graph is constructed using *osm4routing*,[13] a tool that converts a PBF (Protocolbuffer Binary Format)[14] map into two *csv* files: one containing the edges and the other containing the vertices of the graph. For this study, we utilized the city center of Bologna, an area spanning approximately 18km^2. We then used Firebase as the DR and implemented the AS and the MA.

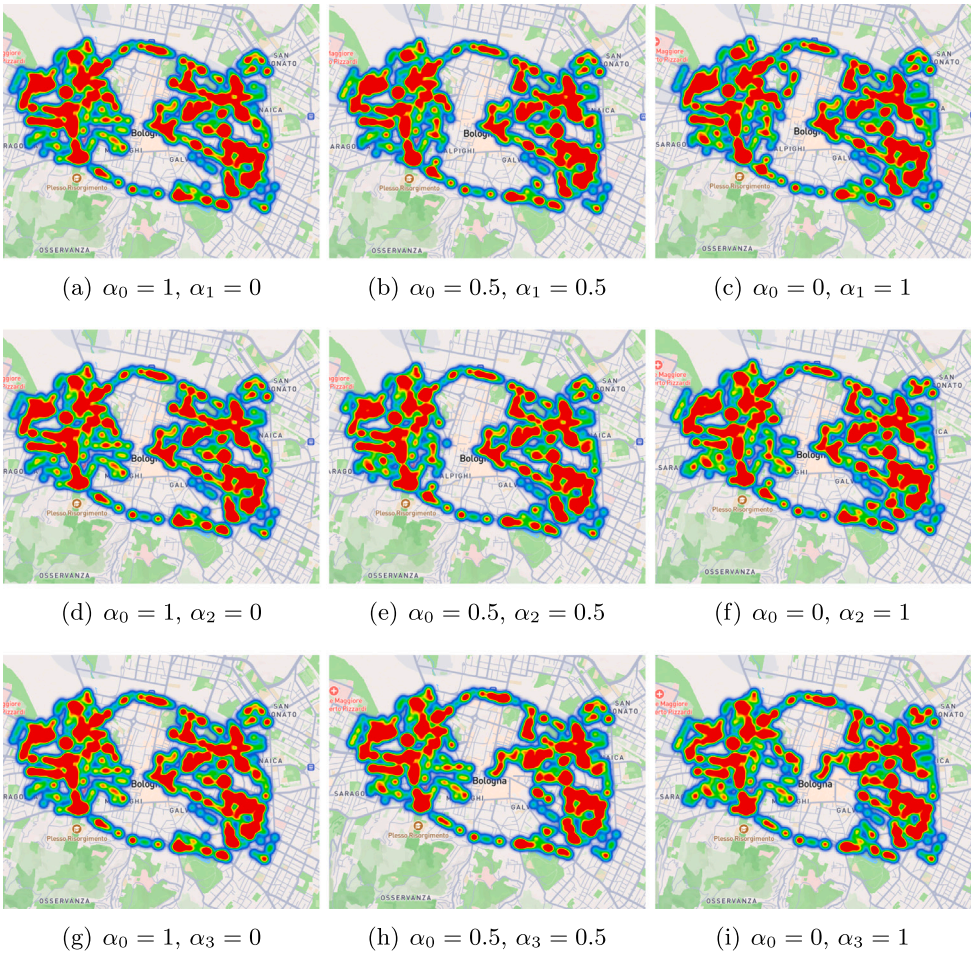
The metrics have been numerically estimated considering $\omega = 100$. Below we report how we normalized each metric in the interval $[0, 100]$.

Road type. We sorted the road types according to an educated guess of what a cyclist would prefer when traveling. In particular we used the following metric values: for "path" $w_{i,1} = 90$, for "primary" $w_{i,1} = 80$, for "footway" $w_{i,1} = 70$, for "living street" $w_{i,1} = 60$, for "service" $w_{i,1} = 50$, for "residential" $w_{i,1} = 40$, for "pedestrian" $w_{i,1} = 30$, for "tertiary" $w_{i,1} = 20$, and for "cycleway" $w_{i,1} = 10$.

Air quality. We estimated the air quality by extracting the PM10 index from the weather stations of the city of Bologna reported in Section 4.2. PM10 is about the amount of airborne particulate matter, a high value can potentially damage the health of a cyclist. Each vertex in the graph is associated with the last measurement extracted from the closest weather station, then the metric value associated to each edge is the average of the two vertex values.

Surface quality. After the controlled test in Supersano, we deployed the DF over the city of Bologna. We ran a cold start by collecting 2000 data points over the city center, however, because of the size of the city, it was impractical to cover it all. We then ran K-means over the collected data points and trained a Support Vector Machines (SVM) algorithm that can be used to infer the type of floor for every new point collected through the MA during the navigation of a path in the city. We manually labeled every cluster, as K-means cannot rank the clusters quality-wise, distinguishing between three classes of floor quality: (1) a good floor – like asphalt with no potholes –, (2) a fair floor – like basalt – or (3) a bad floor – like asphalt potholes or a cobblestone floor. As per Section 4.3, we classified each measured point using the SVM model previously trained. We assign a fixed value for each of these measurements as follows: measurements resulting in good condition (class 1) have a value fixed to 1, while the value for each other class c is calculated as the value of the previous class $c-1$ plus $avg_{len} \cdot \frac{count_{c-1}}{count_c}$, where avg_{len} is the average length of edges and $count_c$ is the number of data points belonging to class c. For instance, in Bologna, with $avg_{len} = 100$ m, if measurements of class 2 occur 1 every 2.7 measurements of class 1 and every 0.5 measurements of class 3, then the value for measurements in class 2 is 270 and 455 for measurements in class 3. We then perform an average over all measurements on an edge to come up with the final metric value for such edge, constraining it between 0 and 100. We experimentally found that this delivers appreciable results even when the dataset is unbalanced.

[13] https://github.com/Tristramg/osm4routing
[14] https://wiki.openstreetmap.org/wiki/PBF_Format



(a) $\alpha_0 = 1$, $\alpha_1 = 0$ (b) $\alpha_0 = 0.5$, $\alpha_1 = 0.5$ (c) $\alpha_0 = 0$, $\alpha_1 = 1$

(d) $\alpha_0 = 1$, $\alpha_2 = 0$ (e) $\alpha_0 = 0.5$, $\alpha_2 = 0.5$ (f) $\alpha_0 = 0$, $\alpha_2 = 1$

(g) $\alpha_0 = 1$, $\alpha_3 = 0$ (h) $\alpha_0 = 0.5$, $\alpha_3 = 0.5$ (i) $\alpha_0 = 0$, $\alpha_3 = 1$

Fig. 11. Heatmap of the most frequently chosen points by the navigation system in the map, by varying the user preference values.

In this Section, we carried out a number of experiments with the goal of evaluating the effectiveness of the navigation system according to the variation of the user preferences. For this purpose, we analyzed 200 different paths by randomly generating the source–destination couples, picking the source from the west of the city and the destination from the east or vice versa, to generate paths long enough for generating appreciable differences. We vary the metric values alternately, so that we consider only one metric at a time that competes solely with the road length, to observe more in detail the impact of such metric on the output. Specifically, we vary α_1, α_2, and α_3 such that $\alpha_j \in \{0, 0.25, 0.50, 0.75, 1\}$, and we define as L_{in} the length of the shortest path between source and destination (*i.e.,* for $\alpha_0 = 1$). The tests are meant to verify how much the navigation system proposes longer paths for a greater value of α_j, with $j > 0$ and vice versa. We define the length of the path computed by our algorithm as L_{fin}.

In Fig. 9 we show how much the navigation system extends the original shortest path length L_{in} and for different values of α_j, with $j > 0$. The three box plots show, on the x-axis, the two values of α_0 and, respectively, α_1, α_2, and α_3. When studying a single metric, we set the other metrics preference values to 0. For example, Fig. 9(a) studies the values of α_0 and α_1, therefore α_2 and α_3 are constantly set to 0. On the y-axis we have the ratio between the final calculated distance L_{fin} and L_{in}, which is 1 for $\alpha_0 = 1$. Results show, as expected, that the smaller is α_0 and the longest is L_{fin}. Both the median and the height of the boxes show an increasing trend, regardless of the considered metric. We can appreciate how the surface quality tends to display higher differences compared to the other two metrics, which are pretty similar to each other. Another important consideration is where the median sits within the box; if it is not perfectly in the middle, it indicates some asymmetry in the data. Finally, the air quality metric shows a more stable trend, because the variability of the metric is itself more homogeneous, as it happens area-wise rather than street-wise and abrupt changes are less likely to observe.

We then tested the navigation system with original paths grouped into four ranges to see if certain trends affect more longer paths. In particular we grouped them as follows: with $L_{in} \in [0, 1]$ kilometers, with $L_{in} \in (1, 2]$ kilometers, with $L_{in} \in (2, 3]$ kilometers, and with $L_{in} > 3$ kilometers. Results are shown in the bar charts in Fig. 10, divided into three plots, one per metric. We observe that, for the air quality and the road surface, the behavior is quite constant, with a small physiological increase in the difference between L_{in} and L_{fin} as L_{in} grows. It is instead noticeable how, for $L_{in} \in (1, 2]$, different values of α_1 have little impact on the

outcome, compared to other path lengths. This may be due to the shape of the road network, for which paths of such a length are more likely to be served by cycleways and be also the shortest.

Finally, we show through Fig. 11 the heatmap of the points that are more frequently chosen by the navigation system by varying the user preference parameter. These figures give us an empirical evaluation on the effectiveness of the system against the topology of the city. For instance, it is noticeable how the western central part of the city is generally avoided when $\alpha_0 = 1$, while, when α_1 is higher, then riders are more likely to choose to travel through some central streets such as "via dei Mille" and "via Galliera", in which there is a dedicated lane for bicycles. The opposite happens for α_3: in Fig. 11(g), $\alpha_0 = 1$, and the navigation system only considers the shortest paths, which elect to pass through the eastern part of the center. For instance, this is evident for "via Zamboni", a central street where the floor is basalt and bicycles often perceive it as bumpy. Fig. 11(h), instead, where $\alpha_0 = \alpha_3 = 0.5$, shows that the navigation system tries to avoid the central streets, given the higher weight of the street quality requested by the user. The differences with Fig. 11(i), with $\alpha_3 = 1$ are quite limited, but it is still appreciable how the navigation system proposes more frequently some specific areas where the floor is considered good, like the ring road zone indicated in the map, which is very well served for bicycles. Regarding α_2, as opposed to the other two, there are no clear differences, because it refers to an area-wise metric, rather than a street-wise one, therefore, we would expect more horizontal and coarse-grained changes. For instance, we can observe, as α_2 increases, a general preference for the southern part of the city, which is expected as it is closer to the hills, and the air pollution is likely to be lower.

7. Conclusions

In this paper, we introduced a comprehensive architectural pipeline for a navigation system designed for cyclists and other micromobility users, with a focus on ancillary metrics that detach from the mere path length. Unlike other solutions in the literature, our system addresses every stage of the process, from data collection and classification to navigation and route planning and provides a holistic approach that allows for the inclusion of different metrics other than the ones presented. Furthermore, any state-of-the-art MCS approaches either concentrate on a single aspect or rely on pre-labeled road data, which is often impractical. Our MCS metric system operates without prior knowledge of the environment, autonomously identifying metric categories, in our case related to road surface quality, using them to classify additional data points, and constructing a robust and consistent dataset. We proved the efficacy of our system in a real urban deployment with real data, hoping that it will foster more research serving micromobility users.

CRediT authorship contribution statement

Federico Montori: Project administration, Methodology, Formal analysis, Conceptualization. **Rocco Pastore:** Software. **Luca Sciullo:** Writing – review & editing, Data curation. **Luciano Bononi:** Resources, Funding acquisition. **Luca Bedogni:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Luciano Bononi reports financial support was provided by Bologna City Council. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is funded by the "Progetto Casa delle Tecnologie Emergenti" - Comune di Bologna - PSC MISE 2014–2020, and by CN-MOST, Spoke 7.

Data availability

No data was used for the research described in the article.

References

[1] F. Montori, L. Bedogni, L. Bononi, A collaborative internet of things architecture for smart cities and environmental monitoring, IEEE Internet Things J. 5 (2) (2017) 592–605.
[2] S.L. Ullo, G.R. Sinha, Advances in smart environment monitoring systems using IoT and sensors, Sensors 20 (11) (2020) 3113.
[3] N. Dahiya, S. Dalal, V. Jaglan, Mobility management in green IoT, in: Green Internet of Things for Smart Cities, CRC Press, 2021, pp. 125–134.
[4] A.A. Ouallane, A. Bahnasse, A. Bakali, M. Talea, Overview of road traffic management solutions based on IoT and AI, Procedia Comput. Sci. 198 (2022) 518–523.
[5] J. Wahlström, I. Skog, P. Händel, Smartphone-based vehicle telematics: A ten-year anniversary, IEEE Trans. Intell. Transp. Syst. 18 (10) (2017) 2802–2825.
[6] A. Capponi, C. Fiandrino, B. Kantarci, L. Foschini, D. Kliazovich, P. Bouvry, A survey on mobile crowdsensing systems: Challenges, solutions, and opportunities, IEEE Commun. Surv. Tutor. 21 (3) (2019) 2419–2465.

[7] X. Wan, H. Ghazzai, Y. Massoud, Mobile crowdsourcing for intelligent transportation systems: Real-time navigation in urban areas, IEEE Access 7 (2019) 136995–137009.
[8] R.A. Purba, N. Riccardo, L. Bedogni, Smart path planner: Enhancing personalized navigation and environmental awareness, in: 2024 IEEE/ACM Symposium on Edge Computing, SEC, 2024, pp. 370–375, http://dx.doi.org/10.1109/SEC62691.2024.00039.
[9] Y. Yao, Z. Peng, B. Xiao, Parallel hyper-heuristic algorithm for multi-objective route planning in a smart city, IEEE Trans. Veh. Technol. 67 (11) (2018) 10307–10318, http://dx.doi.org/10.1109/TVT.2018.2868942.
[10] F. Chiariotti, C. Pielli, A. Zanella, M. Zorzi, A dynamic approach to rebalancing bike-sharing systems, Sensors 18 (2) (2018) http://dx.doi.org/10.3390/s18020512, URL https://www.mdpi.com/1424-8220/18/2/512.
[11] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, H. 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, 2008, pp. 29–39.
[12] T. Wickramarathne, V. Garg, P. Bauer, On the use of 3-d accelerometers for road quality assessment, in: 2018 IEEE 87th Vehicular Technology Conference (VTC Spring), IEEE, 2018, pp. 1–5.
[13] R. Bhoraskar, N. Vankadhara, B. Raman, P. Kulkarni, Wolverine: Traffic and road condition estimation using smartphone sensors, in: 2012 Fourth International Conference on Communication Systems and Networks, COMSNETS 2012, IEEE, 2012, pp. 1–6.
[14] V.M. Souza, Asphalt pavement classification using smartphone accelerometer and complexity invariant distance, Eng. Appl. Artif. Intell. 74 (2018) 198–211.
[15] V. Astarita, M.V. Caruso, G. Danieli, D.C. Festa, V.P. Giofrè, T. Iuele, R. Vaiana, A mobile application for road surface quality control: UnIquALroad, Procedia-Social Behav. Sci. 54 (2012) 1135–1144.
[16] F. Menant, J.-M. Martin, D. Meignen, D. Bétaille, M. Ortiz, Using probe vehicles for pavement monitoring: experimental results from tests performed on a road network, Transp. Res. Procedia 14 (2016) 3013–3020.
[17] F. Seraj, B.J. Van Der Zwaag, A. Dilo, T. Luarasi, P. Havinga, Roads: A road pavement monitoring system for anomaly detection using smart phones, in: International Workshop on Modeling Social Media, Springer, 2014, pp. 128–146.
[18] C. Wu, Z. Wang, S. Hu, J. Lepine, X. Na, D. Ainalis, M. Stettler, An automated machine-learning approach for road pothole detection using smartphone sensor data, Sensors 20 (19) (2020) 5564.
[19] E. Šabanovič, V. Žuraulis, O. Prentkovskis, V. Skrickij, Identification of road-surface type using deep neural networks for friction coefficient estimation, Sensors 20 (3) (2020) 612.
[20] H. Bekku, M. Minami, T. Kawasaki, J. Nakazawa, Detecting potholes from dashboard camera images using ensemble of classification mechanisms, in: 2023 IEEE International Conference on Smart Computing, SMARTCOMP, IEEE, 2023, pp. 108–115.
[21] L. Huidrom, L.K. Das, S. Sud, Method for automated assessment of potholes, cracks and patches from road surface video clips, Procedia-Social Behav. Sci. 104 (2013) 312–321.
[22] M. Hoffmann, M. Mock, M. May, Road-quality classification and bump detection with bicycle-mounted smartphones., in: UDM@ IJCAI, 2013, p. 39.
[23] Y. Taniguchi, K. Nishii, H. Hisamatsu, Evaluation of a bicycle-mounted ultrasonic distance sensor for monitoring road surface condition, in: 2015 7th International Conference on Computational Intelligence, Communication Systems and Networks, IEEE, 2015, pp. 31–34.
[24] K. Zang, J. Shen, H. Huang, M. Wan, J. Shi, Assessing and mapping of road surface roughness based on GPS and accelerometer sensors on bicycle-mounted smartphones, Sensors 18 (3) (2018) 914.
[25] K. Tomiyama, K. Takahasi, K. Tachibana, T. Akeda, T. Hagiwara, A concept of surface roughness index for cycle path: Bicycle ride index (BRI), in: P. Pereira, J. Pais (Eds.), Proceedings of the 10th International Conference on Maintenance and Rehabilitation of Pavements, Springer Nature Switzerland, Cham, 2024, pp. 67–76.
[26] M. Springer, C. Ament, A mobile and modular low-cost sensor system for road surface recognition using a bicycle, in: 2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, MFI, IEEE, 2020, pp. 360–366.
[27] B.D. Setiawan, V.V. Kryssanov, U. Serdült, A. Loshchilov, W.F. Mahmudy, H. Nurwasito, Monitoring road surface conditions with cyclist's smartphone sensors, in: CEUR Workshop Proceedings, (2627) CEUR-WS, 2020, pp. 76–82.
[28] F. Montori, R. Pastore, L. Sciullo, L. Bononi, L. Bedogni, An MCS navigation system based on road surface quality for bicycle riders, in: 2024 IEEE International Conference on Smart Computing, SMARTCOMP, IEEE, 2024, pp. 125–132.
[29] J. Rimediotti, F. Montori, L. Sciullo, L. Bononi, Inferring the urban noise pollution with sparse data through crowdsensing, in: 2024 IEEE International Conference on Pervasive Computing and Communications Workshops and Other Affiliated Events (PerCom Workshops), IEEE, 2024, pp. 643–648.
[30] A. Ferrari, D. Micucci, M. Mobilio, P. Napoletano, Trends in human activity recognition using smartphones, J. Reliab. Intell. Environ. 7 (3) (2021) 189–213.
[31] E. Visconti, Rilevamento della qualità del manto stradale: un approccio sperimentale tramite Machine Learning (Bachelor thesis), URL https://amslaurea.unibo.it/id/eprint/26086/.