SmartWheels: Detecting Urban Features for Wheelchair Users' Navigation †

Sergio Mascetti[a], Gabriele Civitarese[a], Omar El Malak[a], Claudio Bettini[a]

[a] *Università degli studi di Milano, Milan, Italy*

Abstract

People with mobility impairments have heterogeneous needs and abilities while moving in an urban environment and hence they require personalized navigation instructions. Providing these instructions requires the knowledge of urban features like curb ramps, steps or other obstacles along the way. Since these urban features are not available from maps and change in time, crowdsourcing this information from end-users is a scalable and promising solution. However, it is inconvenient for wheelchair users to input data while on the move. Hence, an automatic crowdsourcing mechanism is needed.

In this contribution we present *SmartWheels*, a solution to detect urban features by analyzing inertial sensors data produced by wheelchair movements. Activity recognition techniques are used to process the sensors data stream. *SmartWheels* is evaluated on data collected from 17 real wheelchair users navigating in a controlled environment (10 users) and in-the-wild (7 users). Experimental results show that *SmartWheels* is a viable solution to detect urban features, in particular by applying specific strategies based on the confidence assigned to predictions by the classifier.

Keywords: Mobility impairments, activity recognition, urban navigation

1. Introduction

Modern navigation systems compute the route depending on the user's current mean of transport, like car, public transportation, foot and others. However, none of the most common navigation systems offers specific support for users with limited mobility, like wheelchair users. Indeed, at the time of writing the only form of support available regards information about accessibility in public transportation and it is limited to major cities. While this is surely a useful service, it is clearly an insufficient solution for the overall mobility problem that wheelchair users are facing [1].

A major problem, that emerged during an interview with twelve wheelchair users living in Milan (Italy), is that a person moving on a wheelchair does not know in advance which obstacles she/he will face along a route. For example, even if curb ramps are commonly available at intersections, sometimes they can be missing or damaged. According to interviewed users, these problems are so

Email addresses: sergio.mascetti@unimi.it (Sergio Mascetti), gabriele.civitarese@unimi.it (Gabriele Civitarese), omar.elmalak@studenti.unimi.it (Omar El Malak), claudio.bettini@unimi.it (Claudio Bettini)

† A preliminary version of this work appeared in the Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom 2019).

frequent and their effects are so frustrating that they declare to be reluctant to move along unknown routes.

To address these issues we are currently developing the *Moving Wheels* navigation system that aims at supporting the mobility of people with disabilities by guiding them on routes that are personalized according to their needs. For example, the system will compute a route where curb ramps are available at all intersections for a user moving on an electric wheelchair (it is generally impossible to climb steps up or down with electric wheelchairs).

The problem of offering personalized navigation instructions to people with disabilities has been addressed before in the literature (see e.g., [2, 3]), and can be adapted to this application. *Moving Wheels* addresses an additional challenge: to acquire detailed information about *urban features* i.e., architectural barriers, obstacles (e.g., a step) and accessibility elements (e.g., a curb ramp). This paper focuses on this challenge and presents *SmartWheels*, a solution to automatically recognize the presence of urban features in specific locations from the users themselves: while a user moves in the environment on a wheelchair (e.g., climbs up a ramp) inertial sensors acquire data that can be used to automatically detect the urban feature (i.e., the curb ramp). *SmartWheels* is the essential module of a more complex system to automatically collect and aggregate this information so that it can be used when computing the route for other users. This is a form of data crowdsourcing that does not require user intervention.

This paper has three main contributions. First, it presents a new research problem, motivated by the *Moving Wheels* system: the detection of urban features from wheelchair movements. Second, it illustrates the technical solution to recognize urban features that includes data acquisition, labeling, features extraction, and classification with a supervised machine learning technique. Third, the paper presents an extensive experimental evaluation of the proposed technique on data acquired from 17 subjects with motion disabilities both in an outdoor controlled environment (10 subjects) and in the wild (7 subjects). Results show that it is possible to reliably recognize urban features from data collected in the controlled environment. The recognition is more challenging for the dataset collected in the wild, but we show that the automatic detection process can still be very useful in the crowdsourcing process, by applying specific strategies based on the confidence assigned to predictions by the classifier.

2. The *Moving Wheels* system

Moving Wheels is a context-aware assistive navigation system being developed by the EveryWare Lab in Milan with two main objectives: first, to provide navigation instructions to people with disabilities, guiding them along routes that are personalized according to their abilities. To compute these routes, *Moving Wheels* needs detailed information not only about the road network but also about the urban features that can prevent the user from moving along the route (e.g., steps) or, vice-versa,

can enable him/her to reach the destination (e.g., curb ramps). The acquisition of this information is the second objective of *Moving Wheels*.

The user interacts with *Moving Wheel* through a mobile client that is similar to a traditional navigation app and that guides end-users from a start position (generally their current position) to a target destination. This application has a main difference with respect to other solutions: it allows end-users to finely tune preferences concerning classes of urban features depending on their (dis)abilities. For each class, the user can specify whether the urban features in that class should be avoided or not. A third option is available as well: "avoid if possible" means that the user is able to deal with that urban feature, but this costs some effort. Consider the following example:

- `small-step-up`: avoid if possible
- `small-step-down`: no problem
- `medium-step-up`: avoid
- `medium-step-down`: avoid if possible

The above preferences capture the fact that the user is unable (or not willing) to climb up steps of medium height. Vice versa, descending a short step is not a problem for this user. Also, the user would prefer to avoid to climb down steps of medium height and to climb up short steps.

The *Moving Wheels* web service computes the route when required by the mobile app. While doing this, *Moving Wheels* will avoid all urban features marked as "avoid" and will try to balance the route length with the number of urban features marked as "avoid if possible". For example, consider two alternative routes: one is 200m long with one urban feature marked as "avoid if possible" while the other is $1.5km$ with no urban features marked as "avoid if possible". In this case, the system will automatically suggest the former route, as it is much shorter. In other cases, the system may automatically suggest a slightly longer route, if it has fewer features marked as "avoid if possible". When there is not a stark difference between two or more routes, the system asks the user to select his/her preferred route.

On the server-side, *Moving Wheels* represents the road network as a directed graph in which each edge is labeled with the urban features that the user encounters by moving along that edge, as exemplified in Figure 1.

A major challenge in *Moving Wheels* is to acquire knowledge about the relevant urban features (e.g., steps, ramps), which is needed to populate the graph. We are currently considering these sources:

- existing geo-referenced data stores, including public (e.g., traffic lights from open street map) and private ones (list of curbs ramps from the municipality);
- data annotated by human actors, such as employees, volunteers or end-users, that visit a place either physically or virtually (e.g., looking at *Google street view* images);

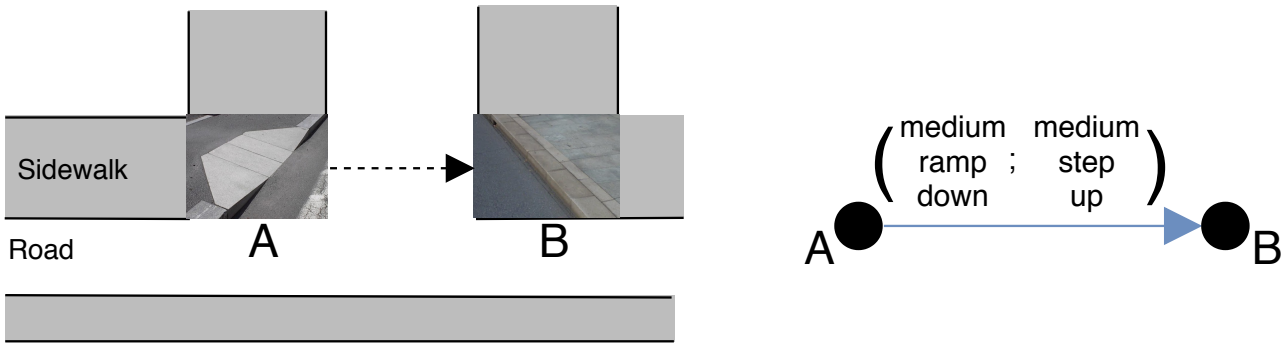


Figure 1: Road network representation

- data automatically extracted from geo-spatial image databases (e.g., *Google street view*), adopting computer-vision techniques, similarly to those proposed in [4].

Each of these solutions has advantages and limitations with respect to a number of factors, including cost (e.g., manual annotation by employees can be costly), scalability (e.g., acquiring data from different municipalities incurs into scalability issues), reliability (e.g, the technique proposed in [4] correctly identifies 93% of zebra crossings), maintenance (i.e., data need to be periodically updated) and types of urban features that can be detected (e.g., some features, like zebra crossings, are easier to detect with computer vision, while others, like the inclination of a curb ramp, are harder to detect).

This contribution focuses on crowdsourcing data from end-users (i.e., people with disabilities). This approach has many advantages: it is scalable, inexpensive, and it keeps information up to date. Since our studies revealed that these users are not really keen to manually enter data or have difficulties doing so, in this paper we show how *Moving wheels* aims at collecting data about urban features with no or limited end-user intervention. For example, pedestrian crossings can be detected from the camera (e.g., a wearable one) and acoustic traffic signals can be detected from the microphone. In this contribution, we focus on urban features that can be detected with inertial sensors mounted on a wheelchair, which include ramps and uneven roads.

3. Problem analysis

The analysis of the *Moving Wheels* system was conducted in two main phases. An informal interview was conducted in 2018 with two sets of users: those using an electric wheelchair and those using a traditional one. The interviews were aimed at better understanding the mobility problems of wheelchair users. The interviews revealed that current navigation systems are only partially useful for this target population because they do not provide crucial information, like architectural barriers, and more generally obstacles for wheelchair users. In the second phase we conducted semi-structured interviews with the participants involved in the data collection process (see Section 5). It emerges that all the subjects agree that a navigation app specifically designed for people with disabilities would encourage them to go outside more frequently and to follow new routes. However, only one person

(a) Electric wheelchair

(b) Self-propelled traditional wheelchair

(c) Electric-propelled traditional wheelchair

Figure 2: Wheelchairs: types, propulsion modes and sensors position.

ever tried one of these navigation apps reporting that "it did not work". This is due to the fact that navigation systems specifically designed for people with disabilities only cover small geographic areas and, outside of these areas, they basically provide the same functions as traditional navigation systems.

Overall, these two analysis phases motivate the need for the *Moving Wheels* system and also provided relevant domain-dependant knowledge. In the following, we report the results from the analysis that are relevant for this contribution.

3.1. Mobility

There are basically two classes of wheelchairs used for urban mobility[2]: electric and traditional ones. The latter can be propelled in three ways: (a) *self-propelled* when the user sitting on the wheelchair uses his/her arms to move the wheels, (b) *attendant-propelled*, when a caregiver pushes the wheelchair and (c) *electric-propelled* in which an electric device is attached to the wheelchair to provide motion. Figure 2 shows some examples of wheelchairs.

Note that an electric wheelchair is different from an *electric-propelled* traditional one: in the former, the motion system (motor, batteries, commands) is integrated into the wheelchair, while in the latter the electric device is external and can be attached when needed. Generally, electric wheelchairs are used by people who are not able to use a traditional wheelchair (e.g., tetraplegic people), while traditional wheelchairs are used by people who are able to use a self-propelled traditional wheelchair and that possibly attach an external electric device when needed (e.g., when they need to cover large distances).

The ability to move in an urban environment and to face obstacles strongly depends on the wheelchair type, on how it is propelled and on the user's abilities. For example, climbing up a steep ramp is generally not a problem with an electric wheelchair, while it can be hard for a self-propelled one if the user is not well trained. Vice versa, climbing up a step can be impossible with an electric

[2]A number of other models are used for indoor use (e.g., in the hospitals), sport and outdoor.

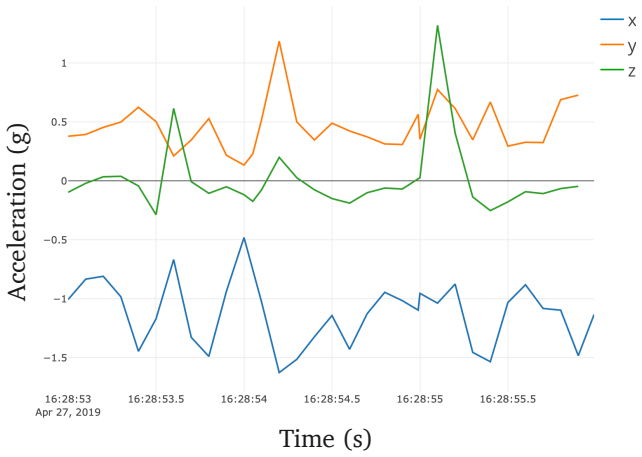


Figure 3: Sensor data acquired from a smartphone stored in a bag.

wheelchair, while it is generally easier with an electric-propelled traditional wheelchair, or with a self-propelled wheelchair if the user is well trained.

In this paper, we focus on detecting urban features from self-propelled traditional wheelchairs. We believe that the methodology and technique we propose in this paper can be easily adapted to the other cases.

3.2. Sensor data acquisition

Since smartphones include inertial sensors, they could be considered as a data source. For this reason, during the interview we asked the participants where they usually keep their smartphone while moving on the wheelchair. It emerges that there are heterogeneous habits: some people using an electric wheelchair have a holder (like the tablet in the red circle in Figure 2a), vice versa a common choice among traditional wheelchair users is to store the smartphone in a bag positioned on the rear side of the wheelchair back.

Our preliminary results show that when the smartphone is not firmly attached to the wheelchair frame (e.g., when it is stored in the bag) the collected inertial data is noisy and recognition is harder. For example, consider Figure 3 that shows accelerometer data recorded by a smartphone stored in a bag while the user is moving on a smooth surface. We can observe that, while the user is only accelerating along the frontal direction, spikes are observable on all three axes. This is due to the fact that the bag keeps swinging and the smartphone inside the bag moves and rotates in all directions.

For this reason, the technique proposed in this contribution is designed to use data from sensors whose movements reflect the user's or the wheelchair's movements. In particular, we consider three types of sensing devices: standalone inertial units (see Figure 4), smartphones and smartwatches. In Section 5 we specify how we positioned these devices during data acquisition. We believe that the experiments with standalone inertial units are significant since we expect that similar sensing capabilities may be easily integrated into next-generation wheelchairs [5], possibly enabling other kinds of applications.

Figure 4: A standalone inertial unit is slightly larger than a coin.

3.3. The urban features of interest

The main focus of the interviews was to understand the challenges that arise when moving with a wheelchair in an urban environment. The following environmental features emerged to be relevant:

- steps: their height and whether they should be climbed up or down;
- ramps: their inclination and whether they should be climbed up or down;
- pavement: whether it is flat or inclined (up or down and how much) and whether the surface is smooth, asphalt or dirt road;
- uneven roads: potholes and their height;
- movement aids: like lifts and stairlift.
- tramway tracks.

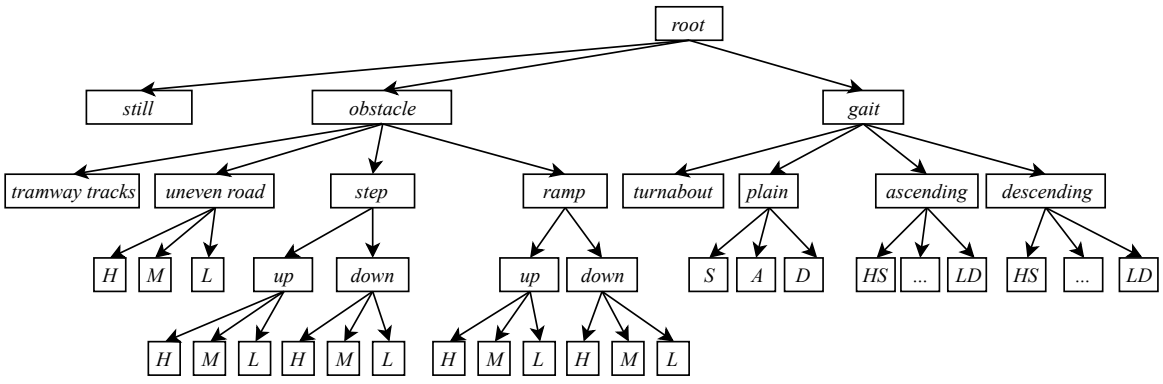


Figure 5: Labels' hierarchy as emerging from the analysis. H=*High*, M=*Medium*, L=*Low*, S=*Smooth*, A=*Asphalt*, D=*Dirt road*, LS=*Low* and *Smooth*, etc.

Based on the observations emerging from the interviews we derived the hierarchical set of labels shown in Figure 5. Each label corresponds to a user's action that discloses the presence of an urban feature. For example, `obstacle-step-up-M`[3] indicates that the user climbed up a step of medium

[3]Henceforth we omit the `action` label (i.e., the root node) when no confusion arises.

height. By knowing the user's position and direction at that time we can recognize the urban feature (the step), its characteristics (medium height) and its orientation (whether it should be climbed up or down when following the same route as the user). Two labels are exceptions as they do not disclose an urban feature: `still` and `turnabout`. The former indicates that the user is not moving, so there is no urban feature to detect. The latter instead does not disclose an exact urban feature but can be used to infer that the user cannot overcome an obstacle and hence can lead to infer a generic accessibility issues when the same behaviour is observed by several users in the same location.

In Figure 5 the first level of labels contains: `obstacle`, `gait`, `movement aid` and `still`. Obstacle represents events with a short temporal duration (intuitively between a fraction of a second and few seconds) while the other events have a longer duration. We discretize steps heights, roads unevenness as well as ramps inclination into three classes (high, medium, low).

4. Automatic detection technique

In order to recognize the urban features of interest, we use machine learning techniques, adapting to our specific domain an approach widely used for sensor-based human activity recognition. The current implementation of our method relies on batch learning: data are first acquired from wheelchair users, then manually annotated with the ground truth, and finally used to train a supervised classifier. Once the recognition model is trained, our system can detect wheelchair users' actions in real-time.

In the following, we describe the main steps of the data management process necessary for the classification task.

4.1. Data pre-processing

The user's wheelchair is equipped with several devices, placed in different positions, each acquiring data from various inertial sensors. Data acquired from these sensors is pre-processed in three main steps: data cleaning, fusion and segmentation.

A common technique for data cleaning is data smoothing, which aims at reducing the intrinsic noise of inertial sensor measurements [6]. Many techniques have been adopted in the literature (e.g., median filter). However, in our domain it emerged that data smoothing actually decreases the recognition rate. We believe that the reason is that some obstacles are crossed in a short time and they result in peaks in sensor measurements. Smoothing those peaks removes important information that is needed to correctly detect obstacles.

Data fusion consists of temporally aligning the data streams originated by each sensor. This is achieved by acquiring sensor data from a single gateway (e.g., a smartphone in our case) and timestamping the data with the gateway clock.

After data fusion, sensor data is segmented using a temporal sliding window approach. The application of this method is influenced by two parameters: window temporal length l in seconds, and

windows overlap factor o in percentage. Banos et al. [7] observe that sliding windows is the most widely employed segmentation technique due to its simplicity and efficiency but it is not always suitable for the detection of sporadic activities (like obstacles). This suggests that dynamic segmentation techniques (like the ones proposed by Zameni et al. [8]) could yield better results. However, one limitation of dynamic segmentation techniques is that they have significantly higher computational costs, while we intend to run our system in real-time on resource-constrained devices. In Section 6 we show that our technique leads to results close to those obtained with a *perfect segmentation*. Hence, in our setting we expect dynamic techniques to have a minor effect on recognition rate that does not balance the increased computational cost.

4.2. Segments labeling

The wheelchair movements (or *activities*) that we need to detect have different duration, from a fraction of a second for *obstacles* to several tens of seconds, for *gait* or *still*. Figure 6 shows an example: a *step* is performed between two *gait* activities.

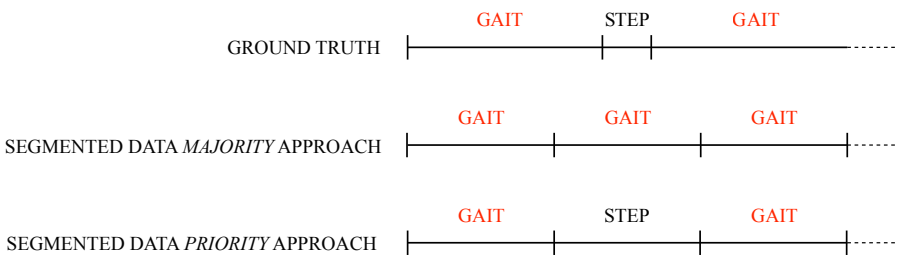


Figure 6: Labelling approaches

As usual for supervised learning approaches, we faced the problem of how to assign a ground truth label to each segment. A possibility is to use very short segments so that each one temporally overlaps a single activity. However, as we experimentally show, using very short segments results in poor classification quality. On the other hand, by using longer segments a user may perform more than one activity during a single segment, as shown in Figure 6 (see the second segment in the second and third lines). In this case, a solution is to label a segment according to the prevalent activity for that segment (the one that is performed more than any other during the segment duration). We call this the *majority* approach and an example of its application is shown in Figure 6.

The *majority* approach turned out not to be effective in our domain since obstacles are generally crossed in a very short time (e.g., half a second). Indeed, since segments have a length in the order of seconds, none of them is labeled as an obstacle (as in Figure 6). To address this issue we adopt a *priority* labeling approach. The intuition is that obstacles are particularly relevant in our domain, so we give them a higher priority when labeling a segment: if a segment overlaps with an obstacle at least for a given percentage p of the segment length, then we label the segment as *obstacle*, independently from the other labels. This is shown in Figure 6: the second segment has an overlap of 25% with a step (a type of obstacle), so, assuming $p = 25\%$, the segment is labelled with *step*. In Section 6 we

show how different values of p impact on the performance and we show that *priority* outperforms the *majority* approach.

4.3. Feature Extraction

From each segment, we extract several statistical features that are widely adopted in the literature for activity recognition from inertial sensors [9]. In particular, we use the following 46 features for each 3-axis inertial sensor:

- For each axis: *minimum*, *maximum*, *difference between maximum and minimum*, *mean*, *standard deviation*, *variance*, *median*, *root mean square*, *kurtosis*, *skewness*, *zero crossing rate*, *number of peaks* and *energy*;
- For each pair of axis: *Pearson correlation* and *coefficient cross correlation*
- For all axes of a given sensor: *magnitude*.

4.4. Data balancing

Data balancing through undersampling and oversampling is often necessary when the annotated dataset is unbalanced, and this is our case too. Indeed, especially in our dataset collected in the wild, obstacles are sparse in the urban environment, and they are usually crossed in a very short time. On the other hand, wheelchair users will likely follow a flat path for most of the time, actually avoiding obstacles when possible. Hence, it is necessary to balance the support values of urban features in the training set. For this reason, we apply a well-known technique combining oversampling and undersampling [10]. The technique is organized in three main steps.

First, the labeled feature vectors extracted from the training set are analyzed to determine which classes are considered as *minority* (i.e., poorly represented) and which ones as *majority* (i.e., with a high support value with respect to other classes).

Then, we apply the SMOTE method to generate, for each feature vector fv labeled with a *minority* class c, s_c synthetic feature vectors [11]. Choosing an optimal value of s_c for each class c is not trivial: a high value of s_c leads to a training set with too many synthetic data, which may lead to overfitting; on the other hand, a low value of s_c may not be sufficient to properly balance the dataset. Our approach consists in choosing s_c considering the support value of c in the dataset: the less the class is represented and the higher the s_c. Each synthetic feature vector is computed considering the u nearest feature vectors with respect to fv. A high u leads to low variability in synthetic feature vectors with respect to the existing ones, while a low u may lead to unrealistic synthetic feature vectors.

Finally, we downsample the *majority* classes using the Edited Nearest Neighbor (ENN) method [10]. In particular, for each feature vector fv we compute its k neighbors. If fv is labeled with a *majority* class while the neighbors are not, fv is removed from the training set. Otherwise, if fv is labeled with

a *minority* class while the k neighbors are labeled with a *majority class*, the k neighbors are removed from the training set. Again, a high value of k may lead to underfitting (i.e., removing too many samples of the majority classes), while a low value of k may not be enough to properly balance the dataset.

4.5. Urban Features Classification

In order to investigate how different classifiers impact the recognition rate, we considered different well-known macro-categories of machine learning algorithms: support-vector-based, tree-based, generative and neural networks. In particular, we experimented with SVM, Random Forest, Multinomial Naive Bayes, and Multi-layer Perceptron. With the only exception of neural networks, we chose the most representative classifier of each category, considering the activity recognition literature. We did not consider more sophisticated deep learning classifiers since they usually require a significant amount of training data, which is not our case (see Section 5). As we show in the experiments, Random Forest resulted to have the highest recognition rate.

Given that our set of labels is naturally represented as a hierarchy, we also designed and implemented a hierarchical Random Forest classifier [12]. In this approach, a separate classifier is used for each internal node of the hierarchy tree. A segment is first classified by the root classifier as belonging to one of the first level labels (for example it is labeled as *obstacle*), and then considered by a specialized classifier in order to get a label from the second level (for example as *tramway tracks*), and further descending the hierarchy until eventually being assigned a label corresponding to a leaf (for example a high ramp). We compared this classifier with a *flat* version with experimental results reported in Section 6.

5. Data collection

In order to validate our method, we acquired two labeled datasets of urban features collected by 17 participants with motor disabilities. During data acquisition, the participants moved by self-propelling their own traditional wheelchair. The first dataset (called *DS1*) has been collected in a controlled environment (i.e., an outdoor training facility for wheelchair users), while the other one (*DS2*) has been acquired in the wild, asking the users to follow an urban route in Milan (Italy). Table 1 summarizes the main datasets characteristics.

From a technical point of view, DS1 differs from DS2 also in terms of the sensors being deployed. DS1 was acquired in a previous phase of our project and only includes data from standalone inertial units while DS2 also includes data acquired by the integrated sensors of a smartphone and a smartwatch.

During the acquisition of both datasets, we noticed a high variability of ways of crossing urban features among different users. For instance, not all users were able to go up or down all steps (e.g.,

	Dataset 1 (DS1)	***Dataset 2 (DS2)***
Area type	Controlled environment (outdoor training facility closed to traffic)	In the wild (urban environment in Milan, Italy)
Number of subjects	10	7
Avg. session duration	10 min.	20 min.
Type of obstacles	Steps	Ramps, uneven roads, tramway tracks
Data sources	3 standalone inertial units	3 standalone inertial units, a smartphone and a smartwatch

Table 1: Comparison of the two datasets

going up a high step is difficult for many users). We also noticed that the speed at which wheelchair users cross urban features is highly variable, mainly based on the participant's physical condition.

Annotation was performed offline, by analyzing video recordings. In order to synchronize sensor data with video recordings, we aligned the clocks of every device with the one of the smartphone used to record the experiments and we used an application that prints the device time on each frame[4]. Actual data annotation was performed using Anvil [13], a free video annotation tool originally developed for gesture recognition.

For each individual, the data acquisition process includes the following steps: a) the participant provides the informed consent, b) standalone inertial sensors are deployed on the wheelchair[5] and are set to collect accelerometer, gyroscope and magnetometer data at 25Hz; c) the participant wears the required mobile devices (in DS2 only), d) the data collection applications are started in order to acquire sensor data, e) the user crosses a predefined route while being video recorded.

In the following, we provide more details about the collected datasets, describing our experimental setup and the acquisition protocol in the details.

5.1. Dataset acquired in a controlled environment (DS1)

In the initial phase of our research, we conducted our experiments at *Spazio Vita*, a Non-Profit Organization (NPO) based in Milan, Italy, that supports people with motor disabilities. This NPO owns an outdoor *training area* which includes common urban obstacles, like steps, ascents, etc. The training area is closed to traffic, so that wheelchair users can practice moving in a simulated urban environment without hazards. Overall, 10 wheelchair users volunteered for data collection in this environment. As shown in Figure 7, wheelchairs were equipped with 3 standalone inertial units attached in different positions on the wheelchair: front-left, rear-right and rear-center.

The route consisted in going on a dirt road, going on asphalt, being still, doing turnabout, going up and down on inclined roads with different slopes (high, medium and low), and going up and down on steps with different heights (high, medium and low).

[4] `http://www.timestampcamera.com/`

[5] We used MbientLab's *MetaMotionR*: `https://mbientlab.com/product/metamotionr/`

Figure 7: Positions of the inertial sensors on the wheelchair

Not all urban features identified in Section 3.3 were available in the environment where we conducted the experiments. In particular, the only available obstacles were the steps, while there are no bumps or potholes. There were indeed some ramps, but they were about 8 meters long, so we do not classify them as obstacles, which should take a short time (e.g., a curb ramp is an obstacle) to cross, but instead we classify them as *gait-ascending* or *gait-descending*.

In Figure 8, we show the hierarchy of urban features that we actually collected.

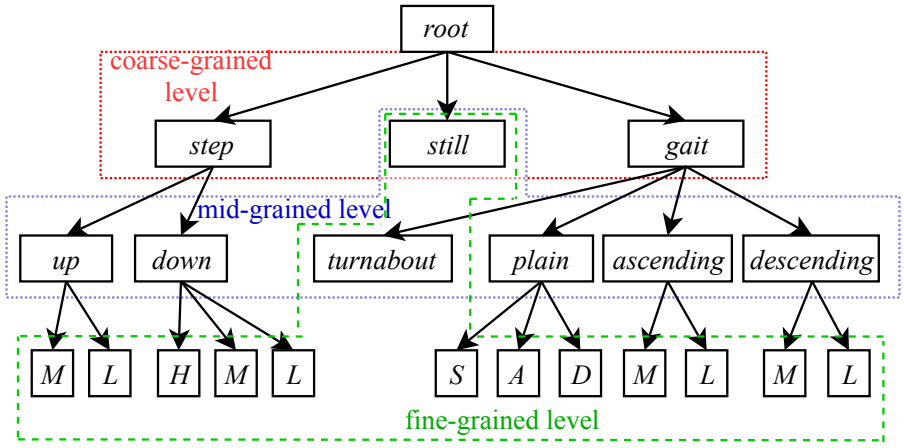


Figure 8: Hierarchy of labels in DS1. H=*High*, M=*Medium*, L=*Low*, S=*Smooth*, A=*Asphalt-like*, D=*Dirt road*.

Table 2 shows some details of the collected data. From this table, it emerges that the dataset is unbalanced. This is due to the fact that many users were not able to cross specific urban features (e.g. high/medium steps) and those who were actually able could not repeat the exercise several times, as these activities are physically demanding. Another reason for the unbalance is that the time required to cross an obstacle like a step is often very short (e.g., half a second) compared to gait, for example.

5.2. Dataset acquired in-the-wild (DS2)

For the acquisition of DS2 we used the same standalone inertial units as in DS1, attached to the wheelchair in the same positions. Additionally, we also collected inertial data from a smartphone and

Urban Feature	#instances	#seconds
Step down high	9	8s
Step down medium	18	14s
Step up medium	14	15s
Step up low	34	27s
Step down low	43	31s
Gait plain on dirt road	16	218s
Gait descendent medium slope	48	230s
Gait ascendant medium slope	43	248s
Gait descendent low slope	54	252s
Turnabout	119	295s
Gait ascendant low slope	53	304s
Gait plain indoor	27	362s
Still	63	628s
Gait plain on asphalt-like	368	2821s

Table 2: Urban features occurrences and duration (DS1).

a smartwatch. We asked the participants to wear the smartwatch on the wrist (see the blue circle in Figure 2b). We aimed at positioning the smartphone where its inertial sensors could actually capture the wheelchair movements without too much noise (e.g., a bag on the wheelchair back is not a good solution, as we observed in Section 3). We first attempted to use a smartphone holder (e.g., like the one shown in Figure 2a), but it was difficult to attach it on some wheelchair models. So we opted for a "leg-band", an arm-band (typically used to hold the smartphone during jogging) adapted for attaching it the user's leg (see the yellow circle in Figure 2b). This allowed us to maintain a fixed position of the device, reducing the noise. The smartwatch and smartphone both run custom Android applications to collect data from built-in inertial sensors.

For the collection of data in DS2 each participant was asked to move along a route of approximately 800m. The route was selected to be as short as possible (hence limiting the participants' effort), but including several different urban features like uneven roads, ramps and road crossings with tramway tracks. Figure 9 shows some examples of the urban features that wheelchair users crossed during DS2 data acquisition.The users also crossed long ramps (at the entrance of the NPO, like the one in Figure 9c) which we labeled as *gait ascendent* or *gait descendent* depending on the direction crossed by the user.

(a) Uneven road: a pothole

(b) A ramp

(c) Gait ascending/descending

Figure 9: Examples of urban features in DS2

Despite our effort to collect a large number of different urban features, it was impossible to collect all those reported in Figure 5 because some of them were not available in the area (e.g., movements aids). Also, during the data collection process, we identified a problem with the steps: most of those available in the area were too high and only a few participants were able to cross them upward or downward. For these reasons, DS2 does not include all urban features we presented in Section 3.3. A subset of the urban features collected in DS2 is shown under the *obstacle* node in Figure 10.

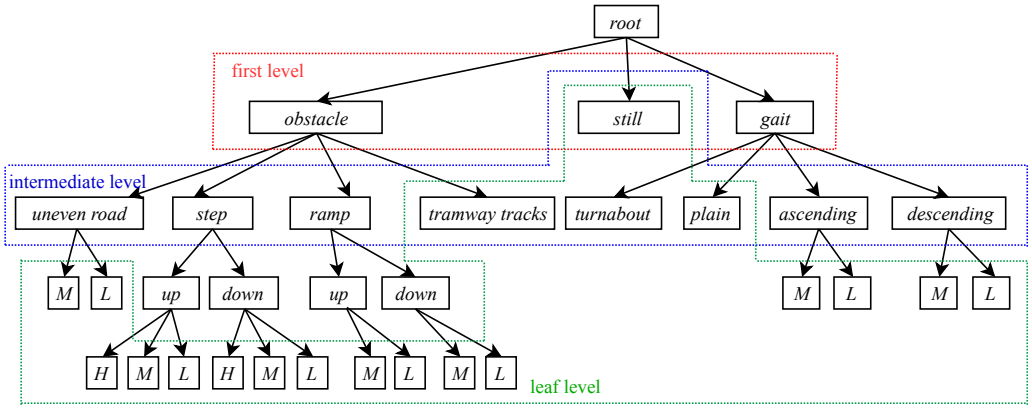


Figure 10: DS2: Hierarchy of labels collected during experiments. M=*Medium*, L=*Low*

Table 3 shows the amount of data collected for each considered urban feature. DS2 is even more unbalanced with respect to DS1. This is due to the fact that obstacles are sparse in the urban environment, and they are crossed by users in a very short time.

Urban Feature	**#instances**	**#seconds**
Tramway tracks	158	112s
Uneven road medium	68	136s
Gait descendent medium slope	18	228s
Turnabout	89	270s
Gait ascendant medium slope	18	279s
Ramp down	115	316s
Uneven road low	248	330s
Ramp up	125	455s
Still	74	946s
Gait plain	727	8706s

Table 3: Urban features occurrences and duration (DS2).

6. Experimental evaluation

In this section, we describe the methodology that we adopted to evaluate *SmartWheel*, and we show the results on the datasets described in Section 5.

6.1. Evaluation methodology

For each segment, the classifier provides a probability distribution among the leaf labels in the hierarchy (see Figures 8 and 10). We consider the label with the highest probability as the one predicted by the classifier.

In order to evaluate the classifier, for each segment we compare the label with the highest probability with the ground-truth, hence marking the segment as true-positive (TP), true-negative (TN), false-positive (FP) or false-negative (FN) for each possible label. For example, if a segment is actually *ramp-up* but is predicted as *ramp-down*, it will result in a FN for *ramp-up* and FP for *ramp-down*. Finally, given the numbers of TP, TN, FP, FN we compute for each label the standard metrics of precision, recall and F1-score.

In order to reliably estimate the accuracy of our approach, we adopt a leave-one-subject-out cross-validation method: given a dataset acquired by n participants, at each fold we use $n-1$ sessions (one for each participant) to train our model, using the remaining one to test it. We then compute the average metrics (precision, recall, F1) among the folds.

Since we modeled urban features using a hierarchical structure, we are interested in investigating the quality of our classifiers at different levels of the hierarchy. Indeed, while it would be desirable to accurately detect urban features at the finest granularity (e.g., distinguish a high, medium and low ramp down), we are also interested in the recognition rate for coarser-grained urban features, like, for example, whether an obstacle is present or not, or whether a ramp has been climbed up or down. For this reason, we identify three groups of nodes in our hierarchy as shown in Figures 8 and 10: *coarse-grained*, *mid-grained* and *fine-grained*.

6.2. Results on the dataset acquired in a controlled environment (DS1)

In the following, we report the main results we achieved with DS1. We tested various classifiers and several parameters trying to identify those yielding the best results. The configuration that gave the best overall results for DS1 is the following:

- a flat Random Forest classifier;
- all available sensor data (3 devices, each with accelerometer, gyroscope and magnetometer);
- a window size of $l = 2$sec and overlap $o = 50\%$;
- a *priority* approach to segments labelling with $p = 20\%$;
- no undersampling/oversampling

The results obtained using the above parameters are reported in Table 4. Overall, the classifier is reliable at the first level. At finer granularities (intermediate and leaf levels) there are large differences among the various activities. For example, *Gait-plain-asphalt-like* and *Gait-descending-low* have F1

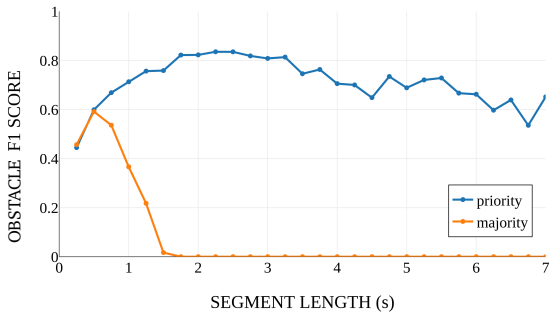
scores of 0.806 and 0.290, respectively. In particular, at fine-grained level, the classifier is often not effective in distinguishing two sibling labels. Consider for example *step-down*: while this label is recognized with high precision and recall, its child *step-down-medium* is not; this is due to the fact that in about 50% of the cases *step-down-medium* is actually classified as *step-down-low*.

Granularity	Class	Precision	Recall	F1 score
Coarse-grained	*Obstacle*	0.893	0.814	0.851
	Gait	0.978	0.986	0.981
	Still	0.935	0.912	0.923
Mid-grained	*Step-up*	0.847	0.727	0.783
	Step-down	0.892	0.847	0.869
	Plain-gait	0.807	0.945	0.871
	Gait-ascending	0.833	0.568	0.675
	Gait-descending	0.737	0.308	0.434
	Gait-turnabout	0.806	0.669	0.731
Fine-grained	*Step-up-medium*	0.688	0.379	0.489
	Step-up-low	0.725	0.806	0.763
	Step-down-high	0.737	0.737	0.737
	Step-down-medium	0.500	0.371	0.426
	Step-down-low	0.647	0.663	0.655
	Gait-plain-smooth	0.621	0.254	0.360
	Gait-plain-asphalt-like	0.709	0.935	0.806
	Gait-plain-dirt-road	0.625	0.320	0.423
	Gait-ascending-medium	0.803	0.677	0.735
	Gait-ascending-low	0.750	0.408	0.529
	Gait-descending-medium	0.590	0.385	0.466
	Gait-descending-low	0.931	0.172	0.290

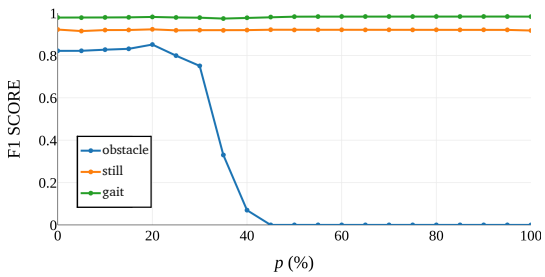
Table 4: Results with best configuration (DS1).

While tuning the parameters with experiments conducted with DS1 four interesting results emerged. First, segment length and the labelling approach strongly impacts on the results. Figure 11a shows that, by using the majority approach, the F1 score for obstacles rapidly decreases when the segment length is longer than 0.5s. Instead, using the priority approach, the classifier performs better when the segments have a length between 1.75 and 3.25 seconds. The second interesting result is that parameter p (see Section 4) strongly affects the obstacle detection rate (see Figure 11b) and best results are obtained with values between 0% and 25%. Third, the classifier provides best results when data from all standalone inertial units is used, but using a single unit only marginally affects the results (see Figure 11c). Finally, the comparison among various classifier shows that flat random forest provides the best result, in term of average F1-score (see Figure 11d). Given the hierarchical structure of our labels, we expected a hierarchical classifier to outperform the others, but actually hierarchical Random Forest resulted to have almost the same performance (but slightly worse) than the flat version. The same holds for Multinomial Naive Bayes. Two classifiers provide clearly worse results: support vector machines and multi-layer perceptron. We believe that this is due to the relatively small training set.

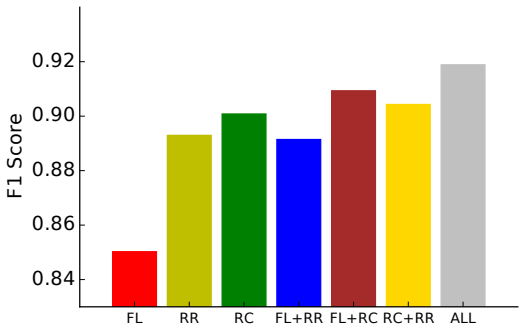
The results also showed that applying data balancing methods on this dataset was counterproductive. We believe that this may be due to the very small number of samples of the minority class (i.e.,



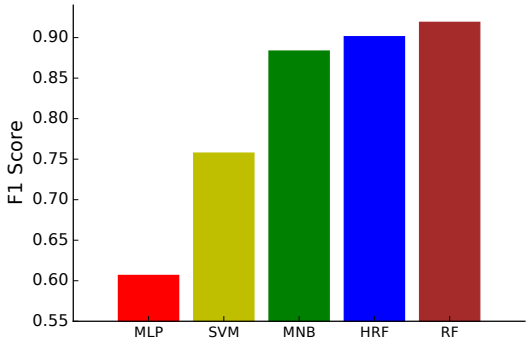
(a) *Majority* vs *priority* segment labeling varying l.



(b) Impact of parameter p.



(c) Position (FL=front-left, RR=rear-right, RC=rear-center).



(d) Classifiers (RF=flat random forest, HRF=hierarchical random forest, MNB=multinomial naive bayes, MLP=multi-layer perceptron, SVM=support vector machine).

Figure 11: Effects of parameters and alternative configurations (DS1).

the step). Hence, oversampling this class leads to unreliable synthetic feature vectors which degraded the recognition rate.

6.3. Results on the dataset acquired in the wild (DS2) considering standalone inertial units only

In order to fairly compare the results obtained on DS1 and DS2, we first analyze the results obtained on DS2 by considering data from the standalone inertial units only. The configuration that gave the best overall results for DS2 is the following:

- a flat Random Forest classifier;
- 3 standalone inertial units;
- a window size of $l = 3$sec and overlap $o = 50\%$;
- a *priority* approach to segments labeling with $p = 20\%$
- oversampling with SMOTE parameter $u = 5$ and undersampling with ENN parameter $k = 3$

Table 5 shows the results obtained with the above parameters. Overall, we observe a reduced recognition rate with respect to the results previously shown for *DS1*. There are three possible motivations for this: first, *DS2* is obtained by a lower number of participants. Second, the recognition task in the wild is intrinsically harder because there is higher variability in the collected data. For

example, crossing the tramway track at different locations can generate substantially different inertial movements. Third, we observed that participants develop skills in limiting the impact of obstacles on the wheelchair movements. For example, when asked to move over an uneven road they select the route that minimizes their effort hence avoiding as much as possible the obstacle itself (this was not possible in the controlled environment). The result is that obstacles are harder to distinguish among themselves and with other actions (e.g., *gait-plain*).

Granularity	Class	Precision	Recall	F1 score
Coarse-grained	*Obstacle*	0.540	0.717	0.616
	Gait	0.905	0.825	0.863
	Still	0.827	0.848	0.837
Mid-grained	*Plain gait*	0.866	0.778	0.820
	Gait ascending	0.829	0.578	0.681
	Gait descending	0.641	0.497	0.560
	Turnabout	0.526	0.746	0.616
	Tramway tracks	0.451	0.728	0.557
	Uneven road	0.456	0.668	0.542
	Ramp	0.430	0.501	0.463
Fine-grained	*Ramp down*	0.307	0.394	0.345
	Ramp up	0.422	0.453	0.437
	Uneven road low	0.305	0.538	0.389
	Uneven road medium	0.227	0.142	0.175

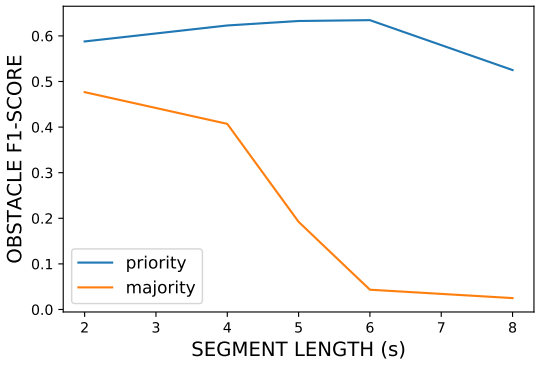
Table 5: Results with the best configuration (DS2)

Most parameters have an impact on the classifier performance similar to what observed for DS1. For example, Figure 12a shows that the priority approach outperforms the majority one, and that the best results are achieved with segments of length between 2 and 6 seconds, while for longer segments the classifier is less reliable in recognizing the obstacles. Also, the comparison among different classifier provided the same result as with DS1: the classifier showing better performance (in terms of F1-score) is flat Random Forest (see Figure 12c).

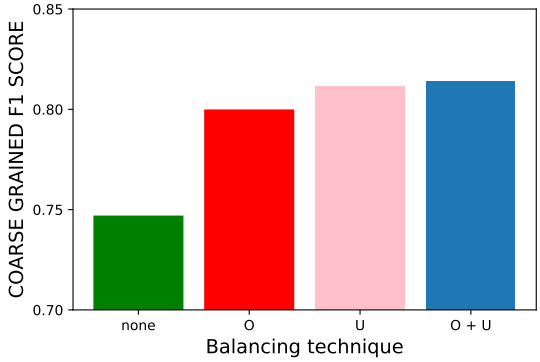
There is one major difference between the configuration yielding the best results with DS1 and DS2: the use of data balancing techniques. Figure 12b shows the impact of the data balancing techniques using DS2. Using either oversampling and undersampling techniques improve on average by 8% and 13%, respectively. Using both techniques the improvement is even larger, with an average F1 gain of 15%.

6.4. Impact of data collected from smartphone and smartwatch

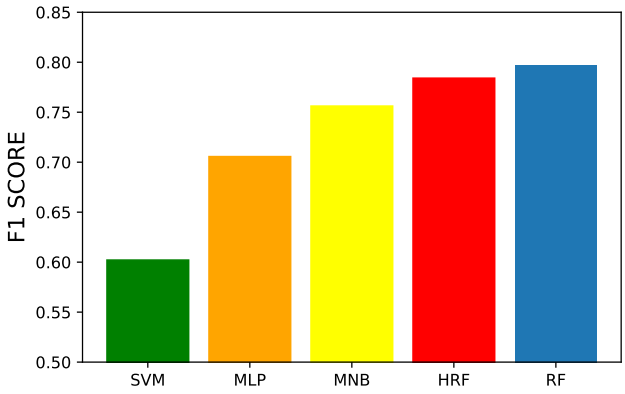
As described in Section 5, DS2 also contains inertial data collected from a smartphone and a smartwatch. We conducted a set of experiments to investigate the recognition accuracy of our classifier when data is collected by sensors on these devices. Figure 13 shows F1 score at the coarse-grained level for different combinations of device. We can observe that using the smartphone only, the accuracy is only marginally affected.



(a) *Majority* vs *priority* segment labeling varying l.



(b) Balancing techniques (O=oversampling, U=undersampling).



(c) Classifier (RF=flat random forest, HRF=hierarchical random forest, MNB=multinomial naive bayes, MLP=multi-layer perceptron, SVM=support vector machine).

Figure 12: Effects of parameters (DS2).

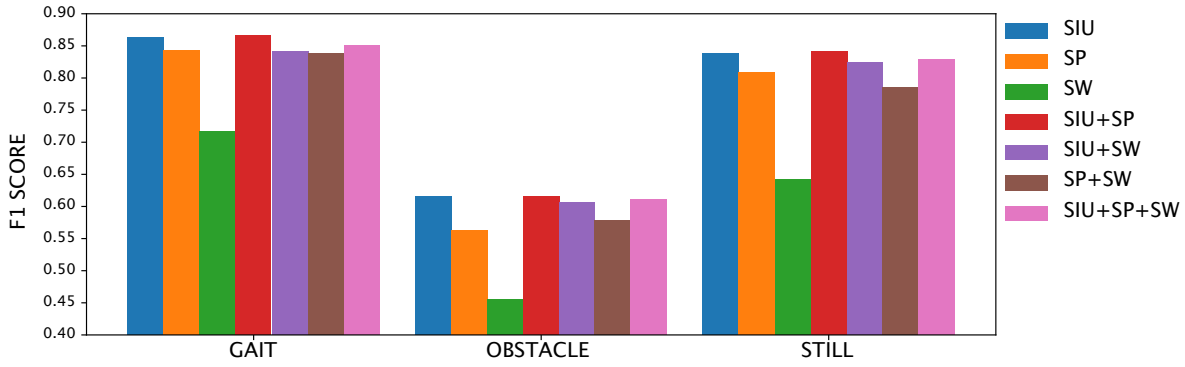


Figure 13: Coarse-grained level average F1 score using different device combinations (SIU = standalone inertial units, SP = smartphone, SW = smartwatch).

This is in line with what we observed in Section 6.2: collecting inertial data from a single sensor, only marginally decreases the classifier reliability. This also confirms that placing the smartphone on the user's leg is a practical solution, as inertial sensors positioned here actually capture the wheelchair movements. Indeed, Table 6 shows that the recognition rate obtained by using only the smartphone is just slightly worse than the one reached by considering only standalone inertial sensors (see Table 5). The average loss of F1 at the coarse-grained level is 4%, while at the mid-grained level is 10%.

By contrast, much lower F1 scores are obtained from the smartwatch. Our intuition is that, being positioned on the wrist, the smartwatch does not reliably capture the wheelchair movements. To support this intuition, during the experiments we also observed that many users did not only move their arms and hands to control the wheelchair, but also to adjust their position on the wheelchair, to cover their mouth while coughing, etc... Clearly, all these movements add noise to the collected inertial data.

The noisy data acquired by the smartwatch also impact when this device is used in addition with other devices: when the smartwatch is paired with the standalone inertial units, the smartphone or both, it does not improve the classifier reliability and, instead, for some urban features, it actually decreases the value of F1 score.

Granularity	**Class**	**Precision**	**Recall**	**F1 score**
Coarse-grained	*Obstacle*	0.491	0.660	0.563
	Gait	0.886	0.803	0.842
	Still	0.792	0.826	0.809
Mid-grained	*Plain gait*	0.838	0.749	0.791
	Gait ascending	0.602	0.302	0.403
	Gait descending	0.563	0.503	0.531
	Turnabout	0.503	0.723	0.593
	Tramway tracks	0.312	0.636	0.418
	Uneven road	0.395	0.410	0.402
	Ramp	0.356	0.512	0.421
Fine-grained	*Ramp down*	0.146	0.245	0.182
	Ramp up	0.363	0.456	0.406
	Uneven road low	0.272	0.305	0.287
	Uneven road medium	0.242	0.194	0.215

Table 6: Results obtained using the smartphone (DS2).

6.5. Evaluation of the segmentation strategy

In our approach, we use a fixed-size sliding window segmentation approach. While this approach is common in the activity recognition literature, it is questionable whether it is suitable for urban feature detection. Indeed, obstacles have a very short duration with respect to other activities like *gait plain*. Hence, dynamic segmentation techniques, like the one proposed in [8], could be more appropriate to tackle this problem. However, such techniques are significantly more expensive from the computational point of view, so a trade-off is required between computational performance and classification reliability.

In order to investigate this problem, we define as the *perfect segmentation* the one generating a single segment for each instance of urban feature/action in the ground-truth. This is depicted in Figure 14 where *perfect segmentation* is compared with the sliding windows approach. Intuitively, this segmentation makes sure that the sensor data corresponding to the crossing of an urban feature is all and only captured by a single segment. A dynamic segmentation strategy has a similar goal. Clearly

perfect segmentation cannot be implemented in a real system since it requires the knowledge of the ground truth.

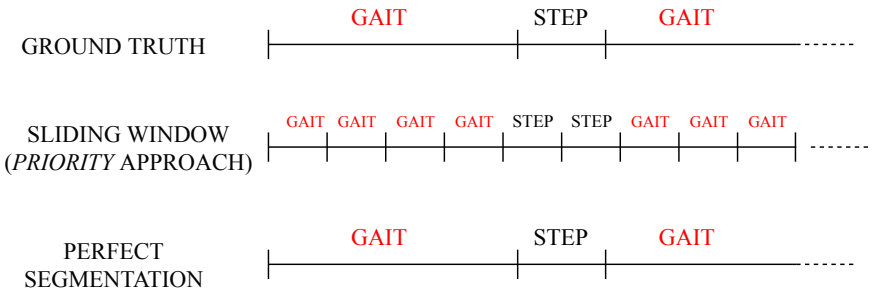


Figure 14: The *perfect segmentation* compared with sliding windows.

Table 7 compares the results of our segmentation and labeling approach with the ones obtained by using *perfect segmentation*. As expected, the overall F1-score of *perfect segmentation* outperforms our method. Indeed, *perfect segmentation* can better isolate the movements related to obstacles to let the classifier better discriminate urban features. This is especially true considering the mid-grained level, where urban features like *tramway-tracks* and *uneven road* exhibit a significantly increased recognition rate. However, from the results it also emerges that the gain in prediction quality obtained by *perfect segmentation* is not as high as expected. Indeed, our method reaches F1 scores which are only slightly worse. This means that, even using *perfect segmentation*, the problem of classifying urban features in the wild is still challenging. Note that, in order to obtain the results above, we added to each feature vector generated from *perfect segmentation* an additional feature that represents the duration of its corresponding segment.

Granularity	**Avg. F1** *Sliding windows*	**Avg. F1** *Perfect Segmentation*
Coarse-Grained	0.772	0.792
Mid-Grained	0.606	0.664
Fine-Grained	0.334	0.343

Table 7: Comparing classifier recognition rates with sliding windows and *perfect segmentation* (DS2)

6.6. System evaluation with k-fold cross-validation

The results presented above were obtained with leave-one-subject-out cross-validation. The rationale for this approach is that we wanted to assess the model capability to classify data for users not included in the training set. However, during data acquisition, we observed that wheelchair users have very different and personal ways of crossing urban features. Hence, motivated by the well-known effectiveness of using personalized activity recognition models [14], in this section we evaluate *SmartWheels* with k-fold cross-validation so that the training and test sets include data from the same users. This evaluation is representative of the hypothetical setting in which *SmartWheels*, in the first phase of deployment, requires the user to manually provide some labels (e.g., through active learning).

In the following, we report the results of k-fold cross-validation on dataset *DS2*. We set the number of folds $k = 7$ (as the number of users in $DS2$), so that the size of the training set is approximately the same as for the leave-one-subject-out cross-validation. We performed a grid search to find the best hyper-parameters using the smartphone as the sensing device. The configuration that gave the best results is the following:

- a flat Random Forest classifier;
- a windows size of $l = 8sec$ and overlap $o = 75\%$;
- a *priority* approach to segments labeling with $p = 0\%$;
- oversampling with SMOTE parameter $u = 3$ and undersampling with ENN parameter $k = 2$.

The results are shown in Table 8. With this form of evaluation *SmartWheels* achieves better results (in terms of precision, recall and F1 score) for all labels in all granularity levels (compare Table 8 with Table 6).

Granularity	**Class**	**Precision**	**Recall**	**F1 score**
Coarse-grained	*Obstacle*	0.80	0.883	0.84
	Gait	0.92	0.856	0.886
	Still	0.88	0.916	0.90
Mid-grained	*Plain gait*	0.916	0.792	0.85
	Gait ascending	0.807	0.93	0.863
	Gait descending	0.797	0.904	0.85
	Turnabout	0.645	0.882	0.745
	Tramway tracks	0.745	0.741	0.743
	Uneven road	0.74	0.90	0.81
	Ramp	0.76	0.80	0.78
Fine-grained	*Ramp down*	0.65	0.704	0.677
	Ramp up	0.722	0.734	0.73
	Uneven road low	0.654	0.856	0.741
	Uneven road medium	0.787	0.733	0.76

Table 8: Results obtained using the smartphone with k-fold cross validation (DS2).

Looking closely at the results, we can appreciate that the recognition rate of obstacles at the coarse-grained level is improved by 28%. Most importantly, the overall results at the fine-grained level increased, on average, by 43% and at the mid-grained level by 30%.

Hence, we expect that our system could potentially benefit by adopting semi-supervised learning techniques to automatically update the recognition model (initialized with labeled data from other users) with examples from the same user that is using the system.

6.7. Entropy-based prediction selection

The results shown so far in this section consider, for each segment, the classifier prediction, defined as the label with the highest probability in the probability distribution returned by the classifier for that segment. This is indeed a common approach to evaluate a classifier reliability. In the *Moving*

Wheels system, these predictions need to be collected by a server and aggregated to infer the presence of an urban feature from multiple (and possibly conflicting) predictions from the same location. While this form of data aggregation is out of the scope of this paper, there is one important aspect to note: for each prediction, it is possible to compute a confidence level in terms of the probability distribution entropy. The lower the entropy is, the higher is the confidence.

There are at least two possible ways to use the entropy values while collecting and aggregating the data in *Moving Wheels*. First, we may decide to crowdsource the prediction only when the entropy value is lower than a given threshold. This can also reduce the network usage by the mobile device running the classifier and the computational overhead on the server. Second, the prediction can be crowdsourced together with the entropy value so that the server can use this information during data aggregation. For example, in the case of two contrasting predictions, the server can prefer the one with lower entropy.

An example of the correlation between entropy and the classifier performance is shown in Figure 15.

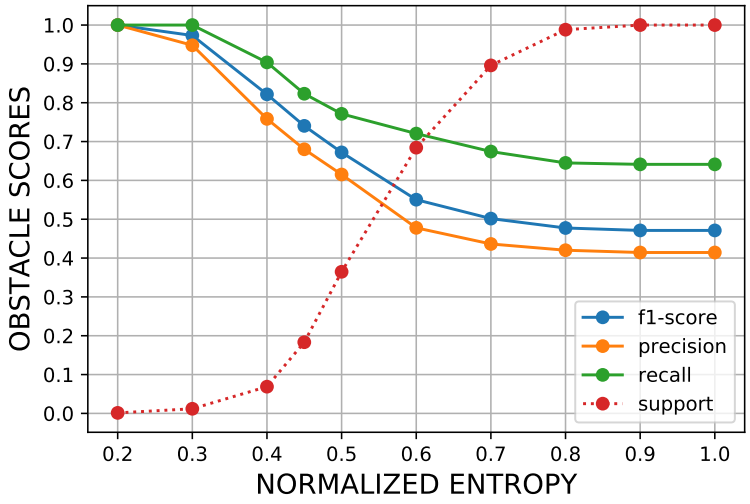


Figure 15: Obstacle recognition rate at mid-grained level varying the entropy threshold (DS2).

On the x-axis, the figure shows normalized entropy values. For each x value, the y-axis reports the average precision, recall and F1-score among the segments whose probability distribution has a normalized entropy below x. Support indicates the percentage of these samples. The figure considers the average values (precision, recall and F1-score) for obstacles at the mid-grained level. We can observe that for low normalized entropy values (e.g., 0.2) perfect recognition is achieved (precision, recall and F1-score equal 1). However, only a few segments (about 0.13%) have such a small normalized entropy value. For larger normalized entropy values the classification performance decreases and the support increases. For example, 18% of all obstacles at the mid-grained level have normalized entropy value of 0.45 or below and for these segments the F1-score is 0.74. Table 9 shows the individual results for the three obstacles at the mid-grained level when normalized entropy is lower than 0.45.

Class	Support	Precision	Recall	F1 score
Tramway tracks	29%	0.769	0.946	0.833
Uneven road	18%	0.750	0.757	0.754
Ramp	17%	0.594	0.846	0.698

Table 9: Results obtained with entropy threshold at 0.45 (DS2).

7. Related work

Several commercial solutions have been proposed to detect urban features from images (e.g., *Mapillary*[6]) or to support people with disabilities during navigation. Similarly to *Moving Wheels*, some of these services provide personalized routes. The main limitation of these systems is that they cover relatively small regions; for example, *Route4U*[7] can only provide navigation instructions in some parts of 5 cities/towns in Europe while *Kimap*[8] only covers a few small towns in Italy. This shows that the main challenge with these applications is the large scale collection of geo-referenced information, and indeed our contribution is aimed at mitigating this problem.

Considering the scientific literature, four main challenges have been addressed in the field of navigation for people with disabilities: (a) to compute the user's position with high precision [15, 16, 17, 18], (b) to compute personalized navigation instructions [2, 3], (c) to effectively convey them (e.g., to blind users) [19, 20], and (d) to detect urban features. This last challenge has been addressed with two different approaches: crowdsourcing and automatic detection techniques. With crowdsourcing, information is manually annotated by end-users or other stakeholders [21, 22, 23] as in the *sidewalk* project[9]. A well-known problem with crowdsourcing is to motivate users to contribute since it often requires explicit user action. This problem is addressed, among others, by Liu et al. [24] while designing the *WeMap* system [25] that, similarly to *Moving Wheels*, is aimed at providing accessibility information about routes and places. Other projects share a similar objective; In particular, several services allow the users to rate a Point of Interest (POI) accessibility (e.g., *aXs map* (`www.axsmap.com`) and *wheelmap* (`wheelmap.org`)). Unfortunately, based on our study, wheelchair users are rarely willing to manually insert accessibility data. As a consequence, only a small fraction of the necessary information is provided, it is often unreliable, and it easily becomes obsolete. Consequently, these services are rarely useful, according to the users we interviewed.

Automatic detection of urban features can be adopted to overcome the limitations of crowdsourcing. Computer vision techniques are effective to detect some urban features, like pedestrian crossings and traffic lights, both from images captured by the device camera [26, 27], and from satellite images [28]. Recently, deep learning has been applied to Gooogle Street View images in order to detect accessibility problems (e.g., damaged sidewalks or obstructions) [29]. The main limitation of these techniques is

[6] `www.mapillary.com`
[7] `route4u.org`
[8] `www.kimap.it`
[9] `sidewalk.umiacs.umd.edu`

that there are some features (e.g., a ramp inclination) that can be hard to detect with computer vision but that our inertial approach can indeed detect. Hence, we believe that the two approaches are complementary.

An alternative approach to automatically detect urban features is to process inertial data and, to the best of our knowledge, the only solution proposed in the literature is based on data collected from people walking in the urban environment [30], while *Moving Wheels* uses data from wheelchair users.

The machine learning methods we propose and adapt to our application are well known in human activity recognition and have been extensively studied in the literature. Supervised or semi-supervised classification techniques are usually adopted to address this problem [9]. Several works proposed to recognize human activities (walking, running, etc.) by analyzing data from inertial sensors found in commonly available mobile devices, like smartphones, smartwatches or wristbands [31, 32, 33]. However, activity recognition for wheelchair users is an application domain with its own peculiarities that has been only partially investigated. Smart cushions have been proposed to monitor lifestyle revealing activities for sedentary subjects (including wheelchair users) [34]. Inertial sensors have also been used to detect simple activities to improve GPS-based localization for both pedestrian and wheelchair users [18]. Differently from those approaches, we rely on inertial sensors to detect activities which in turn disclose detailed information about urban features.

A very closely related work presenting a system called *WheelShare* [35] appeared concurrently with the conference version of our paper. Similarly to *Moving Wheels*, the general aim of *WheelShare* is to design a navigation system suggesting accessible routes based on crowdsourcing data acquired from inertial sensors installed on wheelchairs. The work in [35] can be considered somehow complementary to ours since it is focused on the routing and crowdsourcing problems while we focus on designing and evaluating techniques to automatically detect urban features. Moreover, while urban features in [35] are limited to road surfaces, we investigate the detection problem considering also steps, obstacles, ramps, etc..

8. Conclusion

We presented *Moving Wheels*, an urban navigation system for wheelchair users and we proposed *SmartWheels*, a technical solution for automatic detecting urban features. Training and testing *SmartWheels* required the acquisition of movement data from 17 wheelchair users in a controlled environment and in the wild. Our experiments show that the proposed approach is indeed effective, in particular using data collected in the controlled environment.

While the detection problem is particularly challenging with data collected in the wild, the detection rate can still be high if the crowdsourcing process also takes into account the confidence, which can be computed for each prediction. Another factor that highly impacts the detection rate is the personalization of the recognition model. Indeed, the wheelchair users involved in our data acquisition

campaign exhibited very different and personal ways of crossing urban features. Our preliminary results using k-fold cross-validation suggest that semi-supervised learning could be particularly effective in this domain.

Another interesting insight is that the recognition rate obtained with a single smartphone positioned on the user's leg is only slightly lower than when three standalone inertial sensors are attached to the wheelchair. This suggests that the system can be used with existing technology, without the need of smart-wheelchairs.

In the future we intend to implement the whole *Moving Wheels* system, including a navigation service that computes personalized routes, and a mobile client specifically designed for people with motion impairments that also detects urban features implementing *SmartWheels*. One major challenge is to design a system that populates a geo-referenced urban feature database by integrating the data collected from *SmartWheels* with those from other data sources (e.g., urban features extracted with computer vision techniques). The problem is particularly challenging because this system have to deal with data that changes over time and that are possibly incorrect and approximate. One source of possibly incorrect data is *SmartWheels* that, while generally reliable, does not always compute the correct information (like any other ML-based classification system). Higher classification accuracy can clearly help mitigating this problem and, to achieve this, in future work we want to use the classification's confidence to enable semi-supervised urban feature detection, combining self-learning and active learning. This would enable to continuously improve and personalize the classifier over time. A second source of approximate information is the location associated to the detected urban features. Indeed, in addition to the intrinsic approximation due to the available location sensing technologies (e.g., GNSS), the data-integration system will also have to deal with the fact that the exact time when the user crosses the urban feature (e.g., gets down a ramp) is unknown. Indeed, our technique can detect a urban feature during a time window, but the user's location during that time changes. There are two possible solutions to mitigate this problem: to correlate information from two or more consecutive windows and to use small windows. The latter is indeed the reason why, when tuning the hyper-parameters, we opted for short windows, when this does not significantly affect the recognition accuracy.

9. Acknowledgment

We acknowledge Alessandro Fellegara of *Tribe Communication* for inspiring this work. We also thank the volunteers and *Spazio Vita* non-profit organization for their help in data collection.

[1] R. Velho, Transport accessibility for wheelchair users: A qualitative analysis of inclusion and health, International Journal of Transportation Science and Technology (2018).

[2] L. Beale, K. Field, D. Briggs, P. Picton, H. Matthews, Mapping for wheelchair users: Route navigation in urban spaces, The Cartographic Journal 43 (1) (2006) 68–81.

[3] T. Völkel, G. Weber, Routecheckr: personalized multicriteria routing for mobility impaired pedestrians, in: Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility, ACM, 2008, pp. 185–192.

[4] S. Mascetti, D. Ahmetovic, A. Gerino, C. Bernareggi, Zebrarecognizer: Pedestrian crossing recognition for people with visual impairment or blindness, Pattern Recognition 60 (2016) 405–419. doi:10.1016/j.patcog.2016.05.002.
URL https://doi.org/10.1016/j.patcog.2016.05.002

[5] J. Leaman, H. M. La, A comprehensive review of smart wheelchairs: past, present, and future, IEEE Transactions on Human-Machine Systems 47 (4) (2017) 486–499.

[6] A. Bulling, U. Blanke, B. Schiele, A tutorial on human activity recognition using body-worn inertial sensors, ACM Computing Surveys (CSUR) 46 (3) (2014) 33.

[7] O. Banos, J.-M. Galvez, M. Damas, H. Pomares, I. Rojas, Window size impact in human activity recognition, Sensors 14 (4) (2014) 6474–6499.

[8] M. Zameni, A. Sadri, Z. Ghafoori, M. Moshtaghi, F. D. Salim, C. Leckie, K. Ramamohanarao, Unsupervised online change point detection in high-dimensional time series, Knowledge and Information Systems (2019) 1–32.

[9] O. D. Lara, M. A. Labrador, et al., A survey on human activity recognition using wearable sensors., IEEE Communications Surveys and Tutorials 15 (3) (2013) 1192–1209.

[10] G. E. Batista, R. C. Prati, M. C. Monard, A study of the behavior of several methods for balancing machine learning training data, ACM SIGKDD explorations newsletter 6 (1) (2004) 20–29.

[11] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, Smote: synthetic minority oversampling technique, Journal of artificial intelligence research 16 (2002) 321–357.

[12] C. N. Silla, A. A. Freitas, A survey of hierarchical classification across different application domains, Data Mining and Knowledge Discovery 22 (1-2) (2011) 31–72.

[13] M. Kipp, Anvil-a generic annotation tool for multimodal dialogue, in: Seventh European Conference on Speech Communication and Technology, 2001.

[14] G. M. Weiss, J. Lockhart, The impact of personalization on smartphone-based activity recognition, in: Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence, 2012.

[15] J. Rajamäki, P. Viinikainen, J. Tuomisto, T. Sederholm, M. Säämänen, Laureapop indoor navigation service for the visually impaired in a wlan environment, in: Proceedings of the 6th WSEAS International Conference on Electronics, Hardware, Wireless and Optical Communications, World Scientific and Engineering Academy and Society (WSEAS), 2007.

[16] M. Nakajima, S. Haruyama, Indoor navigation system for visually impaired people using visible light communication and compensated geomagnetic sensing, in: Communications in China, IEEE, 2012.

[17] M. Murata, D. Ahmetovic, D. Sato, H. Takagi, K. M. Kitani, C. Asakawa, Smartphone-based indoor localization for blind navigation across building complexes, in: IEEE International Conference on Pervasive Computing and Communications (PerCom), 2018.

[18] M. Ren, H. A. Karimi, Movement pattern recognition assisted map matching for pedestrian/wheelchair navigation, The journal of navigation 65 (4) (2012) 617–633.

[19] D. Ahmetovic, U. Oh, S. Mascetti, C. Asakawa, Turn right: Analysis of rotation errors in turn-by-turn navigation for individuals with visual impairments, in: Conf. on Computers and Accessibility, ACM, 2018.

[20] G. Dubus, R. Bresin, A systematic review of mapping strategies for the sonification of physical quantities, PLoS ONE 8 (12) (2013) e82491. doi:10.1371/journal.pone.0082491.

[21] K. Hara, S. Azenkot, M. Campbell, C. L. Bennett, V. Le, S. Pannella, R. Moore, K. Minckler, R. H. Ng, J. E. Froehlich, Improving public transit accessibility for blind riders by crowdsourcing bus stop landmark locations with google street view: An extended analysis, ACM Transactions on Accessible Computing (TACCESS) 6 (2) (2015) 5.

[22] M. Saha, K. Hara, S. Behnezhad, A. Li, M. Saugstad, H. Maddali, S. Chen, J. E. Froehlich, A pilot deployment of an online tool for large-scale virtual auditing of urban accessibility, in: Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility, ACM, 2017, pp. 305–306.

[23] C. Gleason, D. Ahmetovic, S. Savage, C. Toxtli, C. Posthuma, C. Asakawa, K. M. Kitani, J. P. Bigham, Crowdsourcing the installation and maintenance of indoor localization infrastructure to support blind navigation, Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 2 (1) (2018) 9.

[24] Z. Liu, S. Shabani, N. G. Balet, M. Sokhn, F. Cretton, How to motivate participation and improve quality of crowdsourcing when building accessibility maps, in: Consumer Communications & Networking Conference (CCNC), 2018 15th IEEE Annual, IEEE, 2018, pp. 1–6.

[25] Z. Liu, N. G. Balet, M. Sokhn, E. De Gaspari, Crowdsourcing-based mobile application for wheelchair accessibility, Journal on Technology & Persons with Disabilities (2017).

[26] S. Mascetti, D. Ahmetovic, A. Gerino, C. Bernareggi, Zebrarecognizer: Pedestrian crossing recognition for people with visual impairment or blindness, Pattern Recognition (2016).

[27] S. Mascetti, D. Ahmetovic, A. Gerino, C. Bernareggi, M. Busso, A. Rizzi, Robust traffic lights detection on mobile devices for pedestrians with visual impairment, Computer Vision and Image Understanding (2016).

[28] D. Ahmetovic, R. Manduchi, J. M. Coughlan, S. Mascetti, Mind your crossings: Mining gis imagery for crosswalk localization, ACM Transactions on Accessible Computing (TACCESS) (2017).

[29] G. Weld, E. Jang, A. Li, A. Zeng, K. Heimerl, J. E. Froehlich, Deep learning for automatically detecting sidewalk accessibility problems using streetscape imagery, in: The 21st International ACM SIGACCESS Conference on Computers and Accessibility, ACM, 2019, pp. 196–209.

[30] A. Bujari, B. Licar, C. E. Palazzi, Movement pattern recognition through smartphone's accelerometer, in: Consumer communications and networking conference (CCNC), 2012 IEEE, IEEE, 2012, pp. 502–506.

[31] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, P. J. Havinga, A survey of online activity recognition using mobile phones, Sensors 15 (1) (2015) 2059–2085.

[32] G. M. Weiss, J. L. Timko, C. M. Gallagher, K. Yoneda, A. J. Schreiber, Smartwatch-based activity recognition: A machine learning approach, in: Biomedical and Health Informatics (BHI), 2016 IEEE-EMBS International Conference on, IEEE, 2016, pp. 426–429.

[33] T. Brezmes, J.-L. Gorricho, J. Cotrina, Activity recognition from accelerometer data on a mobile phone, in: International Work-Conference on Artificial Neural Networks, Springer, 2009, pp. 796–799.

[34] C. Ma, W. Li, R. Gravina, J. Cao, Q. Li, G. Fortino, Activity level assessment using a smart cushion for people with a sedentary lifestyle, Sensors 17 (10) (2017) 2269.

[35] J. Edinger, A. Hofmann, A. Wachner, C. Becker, V. Raychoudhury, C. Krupitzer, Wheelshare: Crowd-sensed surface classification for accessible routing, in: 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), IEEE, 2019, pp. 584–589.