2016 23° Encontro Português de Computação Gráfica e Interação (EPCGI)

# GatherMySteps

Rui Gil
Instituto Superior Técnico
Universidade de Lisboa
Av. Rovisco Pais, 1049-001 Lisboa, Portugal
Email: ruipgil@tecnico.ulisboa.pt

Daniel Gonçalves
INESC-ID
Rua Alves Redol, 9
1000-029 Lisboa, Portugal
Email: daniel.goncalves@inesc-id.pt

***Abstract*—Smartphones have become ubiquitous and a part of ourselves. Since we carry them everywhere, they are the perfect devices to record our memories: through photos, videos and through lifelogging. One specific form of lifelogging is geographic lifelogging, which consists in recording our own trajectories. Anyone with a GPS enabled device, can effortlessly record their own trajectories. Those recordings, however, often contain noise introduced by the lack of accuracy of commercial GPS devices or even regions where no recording was made, due to sensor or human error. Also, the raw recordings do not contain any explicit personally relevant information, such as the relevant places that we moved between, the purpose of our visits, or the transportation modes used in our trajectories. Furthermore, the recordings that need to be processed in a geographic lifelogging context may range from a few, to dozens, on a daily basis. Within this frame, we propose GatherMySteps. GatherMySteps is an application focused on simplifying the processing of GPS recordings — tracks, with an emphasis on usability and proactivity (suggesting the most likely corrections and annotations to the user) to make the lifelogging process as efficient as possible. It provides simplification, editing and learning capabilities so that the user can correct and validate the tracks. Then, a fast and powerful semantic editor is used to annotate the locations and transportation modes of all the trajectories in a day. We evaluated our system using usability user tests, with the help of 20 participants. We conclude that our solution is efficient to process and annotate tracks, with a small amount of clicks and time.**

***Index Terms*—GPS trajectories, Infer transportation mode, User collection, Location based systems, Life-logging, Geographic Applications, Web applications, Graphic information systems, Geographic information systems, Geospatial analysis**

## I. Introduction

The ubiquity of smartphones in our day to day life made a wide range of usage scenarios possible, that were previously difficult or impossible. Phones that we carry everywhere started sensing the world around us. Soon, they started being used to log our daily routines, our steps, our sleep habits and our trajectories. This process of collecting personally relevant information is called lifelogging.

Lifelogging, a mainly manual and burdensome process until recently, is one area that benefited from having a small computer always present and ready to capture the world around us. It is now possible for users to more easily record personally relevant information thanks to specialized mobile applications. Recording our life, however, is only one part of lifelogging. The second part of the process is to analyze the data collected to understand our past life. Its objective is to provide an effective way to reminisce about the past and elicit forgotten memories. While for certain lifelogging domains several easily usable solutions exist, geographic lifelogging is one area that still lacks a complete chain that allows users to record and analyze personally relevant geographic information. Although the recording part is solved, with a wide range of applications available to record trajectories, there are still problems with those recordings.

A recording is a list of Global Positioning System (GPS) point coordinates, where personally relevant information (such as personally relevant locations and transportation modes) are not immediately accessible. Also, recordings often contain points that are inaccurate or there may be parts of a trajectory missing. The latter can be caused by human error, as users may forget to start the recording when leaving a personally relevant location or to stop when they reach their destination. GPS recordings need to be processed so that they can be approximated to the real trajectories, eliminating errors, making changes and completions to the original tracks.

Annotating lifelogging data is not a simple process. When it comes to geographic lifelogging it is no different, there needs to be an efficient and flexible way to accommodate most use cases. The user needs to see their trajectories, at what time they left and at which time they arrived at a personally relevant location. As with any User Interface (UI), it is important to maintain simplicity and flexibility. This is even more important when lifelogging, where users commit their time to process everyday of their lives. Streamlining this process can cut the time users spend cleaning and annotating their recordings.

With this in mind we have created GatherMySteps, which is a semi-automatic, user-assisted application to process GPS tracks. Our main goal is to **provide users with interactive tools to efficiently process track data from personal GPS tracks, taking into consideration meaningful personal semantics regarding location and travel information**.

Our solution is divided into three stages: preview, adjust and annotate. The preview stage allows the user to view raw GPS tracks of a certain day. In the adjust stage, tracks have passed through the automatic processing to clean the data. The result is a better approximation to the ground truth. Those two steps allow users to make corrections as they deem fit, through a track editor. Lastly, users are prompted to annotate their personally relevant locations and transportation modes used during each trajectory. To further ease the work that the user has to do, we infer locations and transportation modes. When a day is reviewed by the user, our system learns new locations and adjusts existing ones. This is also true for transportation modes: our system learns using a classifier that supports dynamic machine learning. This way, suggestions adapt continuously to the user.

Our application was tested using usability user tests with 20 users, using the same data for all of them. This allowed us to understand if our goals of simplicity and ease of use were achieved. The results show that, as intended, our application is easy to use and understand. The metrics collected during the tests show that generally there is a low tendency to err when executing an action, and days are particularly easy to process and annotate.

The rest of the paper is organized as follows. In Section II we discuss related work of systems that display and process GPS tracks. In Section III we look at the problems regarding geographic lifelogging. In Section IV we describe the auxiliary TrackToTrip library. Section V presents the auxiliary backend module, ProcessMySteps, and an overview of the system. In Section VI we present the main module, GatherMySteps. Section VII is dedicated to the evaluation of our system. Finally, in Section VIII, we draw conclusions upon our work, and reflect about how to improve this project.

## II. State of the Art

GeoLife [1] is a system that aims to present GPS recordings in a meaningful way. The map is the main focal point of the system, having the trajectories always visible, displaying details on demand. They have also conducted an experiment [2] with 36 users, where they explored how having spatiotemporal queries, media cues, like photos improved recollection of past experiences. The results show that, even though memory deteriorated over time, having visual cues can greatly increase the ability to remember past experiences. GeoLife can also extract the transportation modes used in the GPS recordings [3]. The initial track is segmented based on changes in transportation modes, which are identified by checking for spatiotemporal pauses. They then label each transportation mode as either bike, bus, car or walk. They tested four classifiers: Decision Tree, Bayesian Net, Support Vector Machine and Conditional Random Field Classifiers. From those, the Decision Tree was the method with the most accuracy.

The MOPSI [4] system allows users to explore tracks in a map. It follows the same approach as GeoLife, presenting the map as the focal point of the system. To avoid scalability problems while displaying tracks with hundreds of thousands of points, they simplify each track and only send to the UI the region to render, which is 50% bigger than the viewport size. MOPSI is also capable of the transportation mode inference [5]. It starts by segmenting the original track into segments with similar speeds. They soft-classify each segment based on a priori probabilities, which will label them either stop, walk, run, bicycle or motor vehicle. A Hidden Markov Model (HMM) is used to exploit correlations between neighbor segments, providing a better accuracy to the inference.

To infer personally relevant locations, Umair et al. [6] uses the location history of a user, clusters them and extracts the locations of each cluster. To prevent problems with the lack of accuracy of GPS recordings, outliers are ignored.

To infer personally relevant locations, Kirmse, et al. [7] explore Leader-Based Clustering (LBC) and Mean Shift Clustering (MSC) [8] a set of tracks. LBC consists in checking if a given point already belongs to a computed cluster, while MSC finds the centroid or centroids, of a point cloud by gravitating a centroid to a region. The results show that, while MSC generally produces better clusters, its computation cost and the fact that it produces similar clusters makes LBC a better option. Work and home locations can be inferred by checking the denser clusters, and at what time they occur: home clusters have more points in the morning and at night, while work clusters have more points during the rest of the day.

To extract transportation modes from GPS tracks, Brunauer et al. [9] use Multilayers Perceptrons (MLP), Logistical Model Trees (LMT) and C4.5 classifiers, which have an accuracy of 92.24%, 92.09% and 84.48%, repectively. Using a set of 54 features. All three methods are likely to misclassify car and bus, because of their similarity.

SenseMe [10] is a system that aims to answer the questions ”Who, What, Where and When?”. To achieve it, the system uses the GPS position of the device and a C4.5 classifier to classify the context as indoor, outdoor or indoor-outdoor. From the velocities between GPS recordings, using another C4.5 classifier it can classify segments, of 10 to 60 seconds, as stationary, walking, running or in-vehicle. If the user is stationary, the location is inferred using the Locus system, Google Places API, or FourSquare Venues API. Social context recognition is done, checking the number of persisting bluetooth connections that are over two minutes. All methods of inference are accurate, with results above 90%, except the social context recognition, that has an accuracy of 87.5%.

All of these systems provide a balance between visualizing trajectories and extracting information from them. They, however, do not adapt to the personal needs or habits of the users. However, they do not adapt to the personal needs or habits of users; in fact, there is a dearth of tools that ask for user input. Instead, they compute immutable classifiers, or make hard coded decisions. These constraints are not acceptable for the lifelogging domain, where the system is expected to fit to the needs of the user and not the other way around.

## III. Geographic Lifelogging

We consider two main workflows for recording personally relevant trajectories: (1) record individual trajectories between two relevant locations, or (2) record continuously through more than one trajectory, for instance recording the entire day. Regardless of how the GPS tracks are recorded, the user then has to download tracks from the GPS device (or mobile phone) to a computer to store and organize them. In a addition to the recording workflow, users often annotate manually their tracks with personally semantic information, such as the locations and times spent at each part of the day and transportation modes used to reach those places. Those annotations go beyond what could be inferred automatically from a general-purpose location service or map, they reflect the personal semantics a location or trajectory has to the user. For instance, what in a map can appear as *9 in Liberty Avenue*; for a particular user can be "Mary's home" (and for a friend of Mary's father it could be "Jack's home"...). So, daily, a geographic lifelogger has to perform all these steps: recording, organizing and annotating a set of tracks.

The workflows are complicated by the inevitable imperfections in the processes involved. Thus, there are a handful of problems of geographic lifelogging, that we need to address. Those can be machine or human errors.

Machine errors are coupled with GPS itself: it lacks accuracy, the same stationary location often has different readings that converge to the true coordinates; and it lacks reliability, either because of the lack of GPS signal or due to the device's limitations. This means a trajectory might be missing some points. Human errors have to do with forgetfulness of when to start/stop recording, and of the actual places visited (and other personally relevant information), in the annotation stage.

The Workflow 1 (individually recording every trip) is prone to human errors, since remembering to start and finish a recording at the right time, at the right place, is a task that can be easily forgotten. Often, recordings of trajectories start after one has left the start location. Workflow 2 (a single recording for the entire day) is prone to have more GPS errors, since there will be recordings made indoors, where the GPS signal is weak, or non-existent.

## IV. TrackToTrip

As we have seen in Section III, there are problems that need to be solved to process and annotate a GPS track. While correcting a set of tracks sporadically is doable, it is also tedious. This is a problem of the lifelogging domain, as it is common to process a dozen or more tracks per day. We need to automate this process the best we can, putting the user as a reviewer. We can also learn locations, transportation modes and their most common trips to give suggestions when annotating and making corrections to tracks. With that in mind, we have developed the TrackToTrip library, which helps the user clean tracks, transforming them into trips, greatly reducing the amount of work the user has to do.

To extract more meaningful information from a raw GPS recording, we transform a **track into a trip**.

A **track** is a collection of points recorded during a trajectory. It is imprecise because of the inaccuracy of GPS; and it also may lack some recordings, either due to signal loss, or because the user forgot to start or stop the recording in the correct places.

A **trajectory** is the ground truth. It is the true, continuous, path taken by the user between two semantically relevant locations.

A **trip** is the best approximation to a trajectory. In our approach, a trip is derived from a track.

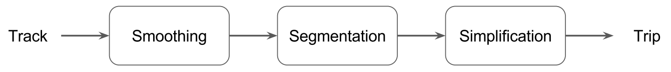


Fig. 1: Transforming a track into a trip

We have identified a set of actions that we need to take to transform a track into one or more trips. As per Figure 1, those are: *smoothing*, *segmentation* and *simplification*.

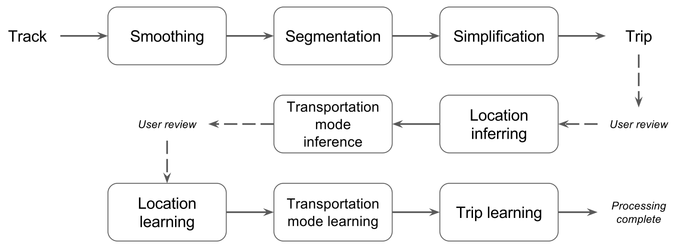


Fig. 2: TrackToTrip processing chain

As shown in Figure 2, besides transforming a track into a trip, the TrackToTrip library is also capable of extracting personally relevant information about a trip. More precisely, we can extract personally relevant locations and transportation modes. To achieve better inference, we learn the locations and transportation modes of each annotated and validated trip. Furthermore, we can learn common trips, to suggest possible completions of missing segments.

### *A. From a track to a trip*

The first step of this transformation is to smooth a track, applying a Kalman Filter [11], which removes some jitter and noise points in the recording.

The second step is to turn a track into multiple segments so that each segment represents a trajectory between two personally relevant locations. This step is particularly useful to identify places where the user has forgotten to stop the recording, as described in Workflow 1, and essential for Workflow 2. To that end we split a track if the temporal distance between two consecutive points is higher than a given threshold, defined by the user. Also, if the user is stationary at one location, there will be a denser concentration of points at that location. This means that we can split a track at each spatiotemporal cluster. We use the DBSCAN [12] algorithm to identify spatiotemporal clusters. DBSCAN was chosen because it does not need to know, in advance, the number of clusters and because it uses the distance between nearest points as a metric.

The third step, needed to transform a track into a trip, consists in removing redundant points. We simplify a track while maintaining its spatiotemporal characteristics. To guarantee those characteristics we use the Spatiotemporal Trajectory Compression Algorithm (SPT) [13], which reduces the number of points of a track without changing its spatiotemporal characteristics, such as its speed and distance accuracy.

The ordering in which smoothing, segmentation and compression are executed guarantees that we do not lose spatiotemporal information at each step. We perform the smoothing step first to produce a cleaner track, which in turn, will benefit the segmentation and simplification steps. If we simplified the track first, we would remove points that form clusters, thus making the segmentation step less accurate. The need to keep the same information is also why we use SPT, instead of Ramer–Douglas–Peucker (RDP) [14]. While the latter offers better compression ratios, it does not maintain the spatiotemporal characteristics of a track.

The result of this chain is a trip that can be validated by the user where the start and end points mark potentially relevant locations.

### B. Location inference and learning

When dealing with geographic lifelogging, one of the most important aspects is to identify personally relevant locations. To that end, and to further improve the inference of locations, we learn new locations and update existing ones.

Internally, a known location is represented by a group of points, and a centroid of those points. Given a point, a location will be inferred by retrieving the nearest location centroid and its label.

Known locations are updated by identifying the closest location centroid with the same location name (based on previous iterations). The new point will join the list of points (cluster) that describe the location and the centroid will be updated. We use *DBSCAN* to extract one cluster and its centroid. *DBSCAN* also marks points as noise (outliers), which improves the precision of the centroid according to the coordinates of the location.

### C. Transportation mode inference and learning

Before inferring transportation mode, we have to consider that a trip is often composed of more than one transportation mode. As so, the first thing we have to do, is to identify where the transportation mode changes: change points. To perform change point segmentation, we use the Pruned Exact Linear Time (PELT) algorithm. The PELT algorithm takes a data series and identifies where the properties of the data change. We look for segments that have a different variance, on normally distributed data. Specifically, we try to identify changes in variance in the acceleration differences between each consecutive pair of points. The result is a set of change points candidates.

From each segment, we extract nine features. Each corresponds to the rounded velocity spent during 10%, 20%, ..., and 90% of the duration of the segment. We use a Support Vector Machine (SVM), specifically a Stochastic Gradient Descent classifier, to label each segment either as vehicle, train, foot, or airplane. We trained the classifier with the GeoLife dataset. Using two-fold evaluation we obtain an accuracy of 84%. Purposefully, our classifier is able to learn dynamically, adapting to the user patterns after each trip has been processed.

### D. Trip learning

Besides learning locations and transportation modes, we can also learn the most common trips for a particular user. If so, we can build a graph that represents that user's mobility patterns. Furthermore, it allows the library to complete missing pieces of a trip between two points, either by identifying a likely trip from whatever information is available, or by being able to suggest the most frequently used trip between those places as a possible replacement, when no information exists.

Before learning trips, we have to be able to compare them. To compare if two trips are equal (the user followed the same route), we devised the Trajectory-Hausdorff Ratio (THR). The THR is based on the Trajectory-Hausdorff Distance (THD) [15], it has more lenient distance functions more suitable to the noisy GPS readings that we expect.

The THR compares how similar two line segments are by multiplying parallel and angular distance ratios. The angular distance ratio is computed by comparing the angles of the two segments returning a value between 0 and 1, where 0 is given by an angle difference of $180^o$; and 1 is given by an angle difference of $0^o$. The parallel distance ratio is computed by calculating the distance between one line segment start and end points, to the closest point within the other line segment. The distances are averaged and a ratio is computed against a given maximum distance threshold so that distances above the threshold yield a ratio of 0. If the two line segments are either equal, or cross one another, the resulting ratio is 1.

To obtain the similarity ratio between two two trips we have to compute the THR between each two segments that are within a threshold distance. The similarity ratio between the two trips is given by averaging all THR computations.

To learn a trip, we check whether there are any trips that are similar to the one we want to learn. If there is a similar trip (above the 0.8 threshold) we update the existing representation, to keep refining the existing representation of a particular trip.

## V. ProcessMySteps

ProcessMySteps is the backend module responsible for managing the processing of tracks, as well as the control flow between the different stages of the system. It is a lean process, exposing a lean Representational State Transfer (REST) Application Programming Interface (API) with 16 endpoints. Since our system is designed with lifelogging in mind, we opted to maintain a single state server; this eliminates concerns regarding data isolation and privacy between different users.

The API was designed to be simple, yet flexible. The current state of the system is available at `/current` endpoint, while advancing through the different processing stages is done calling `/previous` and `/next` endpoints. When those endpoints are called a representation of the server state is sent. The server state representation includes the days left to process and the current day, which is composed by a list of tracks or trips.

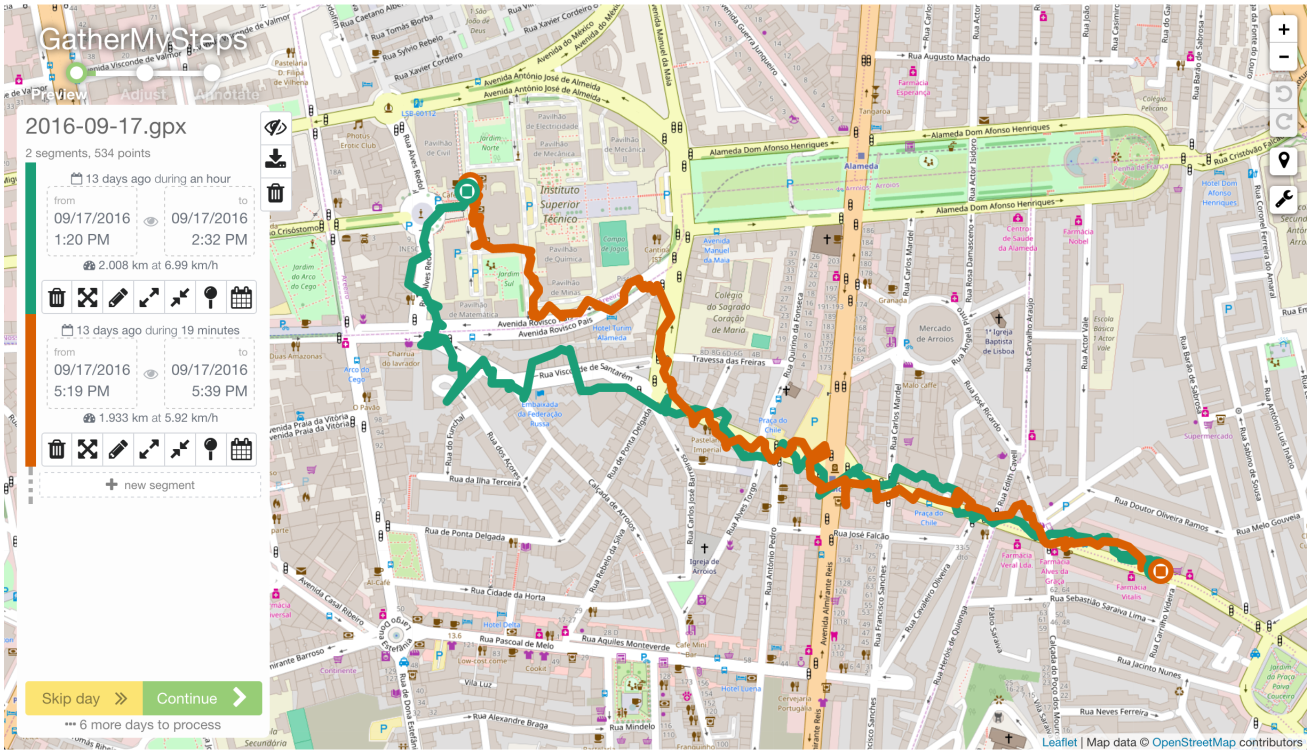


Fig. 3: Overview of GatherMySteps

The reason that our backend is this simple is because we have TrackToTrip, which transforms track into trips and infers and learns information about them. Furthermore, GatherMySteps implements the necessary actions that allows one to edit tracks inside the browser, sending the result back to the server. GatherMySteps and ProcessMySteps use the JavaScript Object Notation (JSON) format to exchange information. Figure 4, shows a high-level architecture of the system, which is organized in a client server architecture, where GatherMySteps is the client and ProcessMySteps is the server.

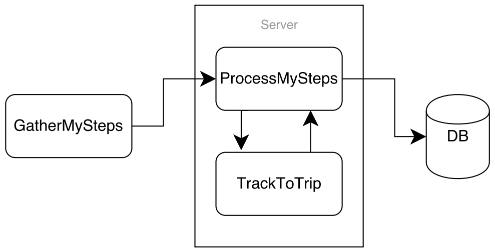


Fig. 4: Architecture of our system

When a day is processed we save the trip and its locations into a PostgreSQL database. Moreover, we create backups of the original tracks, we generate a GPS Exchange Format (GPX) file with the resulting trips and a text file of the LIFE annotations for that day. The LIFE format [1] allows users to annotate personally relevant information about their days.

[1]Created by Daniel Gonçalves, https://github.com/domiriel/LIFE, last accessed in September 28th, 2016

## VI. GatherMySteps

GatherMySteps, Figure 3, is the main component of our system. Our main premise is to have a fast UI to correct and annotate GPS tracks as easily as possible. While geographic lifelogging is an everyday effort, the processing of previously recorded tracks may happen in burst, where users process tracks weeks or months old. This, allied to our innate inability to recall precisely past life events makes correcting and annotating GPS tracks a hard process.

To help the user, we use the automatic processing and allow the user to see the tracks of each day. After the first contact, we ask the user to annotate personally relevant locations and transportation modes. Moreover, processing is done day-by-day; grouping different tracks into a single day helps recalling a past day.

We divided the processing phase into three steps: *preview*, *adjust* and *annotate*. The preview step allows one to view and edit the raw GPS track. While edits at this stage may be useful, it serves as a way to compare with the result of the automatic processing phase. On the adjust step, every track has been transformed into one or more trips. By using this step, the user has the power to correct any incorrect decisions made by the TrackToTrip library, by editing any trip being processed. The final step is solely dedicated to the annotation of locations and transportation modes based on each trip, to that end, we have developed a semantic editor.

### A. *Track editor*

As we have discussed, the track editor is the main component that powers the editing of tracks, in the preview and annotate steps.

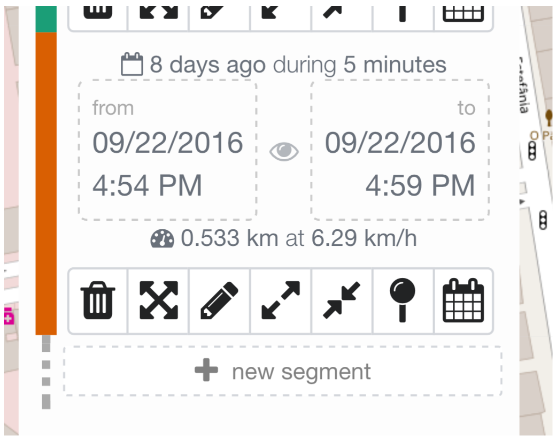


Fig. 5: Representation of a track/trip

The focal point of the track editor is the map, where tracks and trips are shown. This allows the user to explore and to identify locations. Every track segment is identified by a color, from a set of eight different colors, generated using ColorBrewer2.0 [2]. In addition to the map, each track and trip is represented in the left pane, as shown in Figure 5. It is from there that the user can correct tracks and trips to better represent their trajectories. Users can use the following functions to edit their tracks and trips:

- *Edit points*: add, remove, move, or edit the time of any track or trip point, individually;
- *Split*: split a track or trip into two;
- *Join*: join two trips into one;
- *Inspect point*: inspect position, time, ordering, velocity and distance of a point.

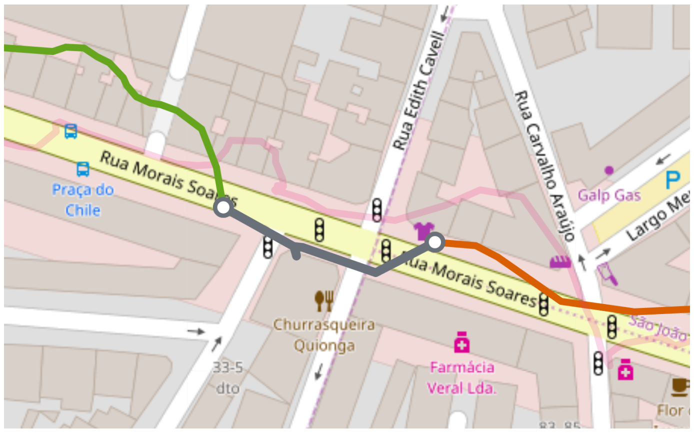


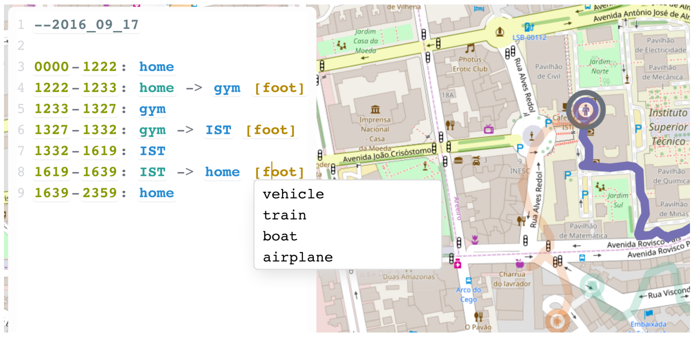
Fig. 6: Suggestions to join two tracks

From those, the *join* action is the only one that calls the server, asking for suggestions for a possible path to join, as shown in Figure 6. The alternatives are displayed in different colors, with different line width. Thicker line segments represent more common trips.

To help users identify tracks and trips more easily, we have also implemented buttons to center the start and end points of a track or trip on a map, and actions to toggle visibility and to zoom to a specific track or trip.

### B. *Semantic editor*

The semantic editor, Figure 7, is used to annotate locations and transportation modes using the LIFE format.



```
--2016_09_17

0000-1222: home
1222-1233: home -> gym [foot]
1233-1327: gym
1327-1332: gym -> IST [foot]
1332-1619: IST
1619-1639: IST -> home [foot]
1639-2359: home
```

Fig. 7: Semantic editor, for LIFE annotations

It works, seamlessly, in two ways: as a drop-down form, where users click and select suggestions, and it also works as a text editor. This allows users more experienced with the system, the flexibility of text editing, while having help from our suggestion engine.

The editor works like an Integrated Development Environment (IDE) for the LIFE format and the current day (and respective tracks) that are being processed. We create an Abstract Syntax Tree (AST) of the text and associate the existing tracks with parts of the text. This allows us to decorate LIFE elements, such as time, spans, trips and locations, and makes possible highlighting point and a segment when the mouse hovers above those decorated elements. Initial suggestions, computed in server, provide users with a starting point, from which they can change. Furthermore, to help the user, we show suggestions for locations and transportation modes, Figure 7. This is what allows the text editor to work like a drop-down-like HTML form.

## VII. Evaluation

Our objective was to create a system to simplify the processing of GPS tracks by having a simple, yet powerful, UI that allows to edit and annotate tracks. To evaluate how user-friendly and how effective our application is, we have devised usability user tests.

### A. *Methodology*

We have selected 20 users unfamiliar with GatherMySteps to test our application. Every user was submitted to the same procedure and used the same data during the whole procedure. The tests were executed in a controlled environment, without distractions, during a maximum period of 45 minutes. Every

[2]http://colorbrewer2.org, last accessed on September 28th, 2016

user used the same computer and pointing device (a generic mouse). The tests began with a preparation period, where users started by reading a handout containing a brief description of the system, the tasks to execute and a map marked with the semantic locations present in those tasks (necessary since the users were not using their own data and, thus, would otherwise have no reason to assign particular meanings to any of the locations referred to in the dataset). We did a four minute demonstration of our system, where we showed the major functionalities of GatherMySteps. Furthermore, to avoid unnecessary frustration during the test, caused by the initial contact with the system, they were given four minutes to explore the application. During this period they could ask further questions about the system. After the adaptation period users started executing the tasks. Each and every user executed the tasks in an unique, randomized order.

The dataset consisted of six days worth of tracks, recorded by us, with a total of eight files, which contained 16 trips. When recording, we set up our devices to collect GPS recordings as often as possible; the result is a median sampling rate of one point captured every five seconds, with an average of 567.6 points per file.

The users being tested had a total of eight tasks to perform. Tasks 1 to 5 were focused on actions, designed to evaluate how visible and easy an action was. An example of such is: “September 22, 2016: Split a track near the grocery store.”

Tasks 6 to 8 required the users to process a day’s worth of tracks. To do that, they were given a scenario where they had to make adjustments when needed and annotate a track, such as: “September 19, 2016: I left home, at 1:24pm, to go to the gym. I finished training at 2:35pm and went back home. At 2:37pm I was on my way to Starbucks, by bus.”

After completing all tasks, users had to fill a questionnaire where they answered questions about themselves and about our solution.

## B. Results & Discussion

Users that tested our system averaged 22.5 years old, 10% of them never use the GPS, 50% of them use the GPS once a week, 20% of them one to three days a week, 5% of them four to six days a week and the rest 15% use the GPS everyday. For those who use GPS, accessing maps and localization services are the main usage scenarios. From the 20 users, two had already done geographic lifelogging before.

| Task | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Minimum | 2 | 2 | 2 | 2 | 2 | 7 | 7 | 10 |
| Median | 6 | 4 | 3 | 2 | 2 | 10 | 10.5 | 11.5 |

TABLE I: Minimum amount of clicks required and the median clicks registered for each track

Table I describes the number of minimum clicks necessary to complete each task, determined before the start of the tests, and the median of clicks registered during the experiments.

Figure 8 plots the clicks necessary to complete each task, using Tukey box plots. With this data, we can see that tasks

| Task | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| First Quartile | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Median | 1 | 1 | 0.5 | 0 | 1 | 1 | 1 | 2 |
| Third Quartile | 2 | 2 | 1 | 1 | 2 | 1.25 | 3 | 3.25 |

TABLE II: Errors for each task

more focused in single actions (tasks 1 to 5), using the track editor, require a low amount of clicks. This is highlighted by the number of clicks that tasks 2 to 5 registered. Task 1, which requires the user to find a point, takes more clicks than the other action focused tasks. Since it is exploratory, the time spent per click in task is lower than in others, because users perform successive clicks until they find the answer. The same is true for the more complex tasks. More complex tasks, such as 6, 7, and 8, require users to explore, make small adjustments and annotate a day using the semantic editor.

Using Table II, we can see that there was not a significant amount of error introduced by the semantic editor, as the median or errors is similar to the simpler tasks. Also, the time per click is more similar to other complex tasks, which can be seen in Figure 9.

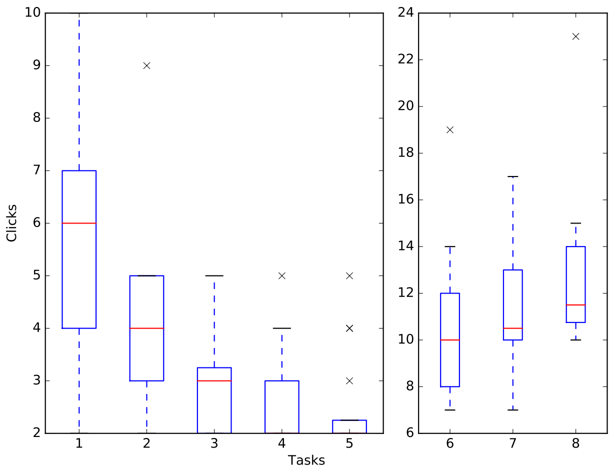


Fig. 8: Clicks to complete each task

As we can see in Figure 9, complex tasks have a similar time per click ratio, which also are comparable to focused tasks. This is because of the efficiency of the semantic editor. Often, the annotations were correct, or accessible through suggestions that require a two clicks: one to open the suggestions, the other to open suggestions. Users that choose to type the locations, when annotating, while they did less clicks, often spent more time than those who used suggestions.

When asked if the application was easy to understand, the average score was 4.1, in a scale of zero to five, where zero is very hard and five is very easy to understand. Regarding usability, our solution had an average score of 4.15, in a scale of zero to five, where zero is very hard and five is very easy to use. Also, since lifelogging is a continuous process, we asked the users to rate how easy GatherMySteps was to use, again, in a scale of zero to five, where zero is very hard and five

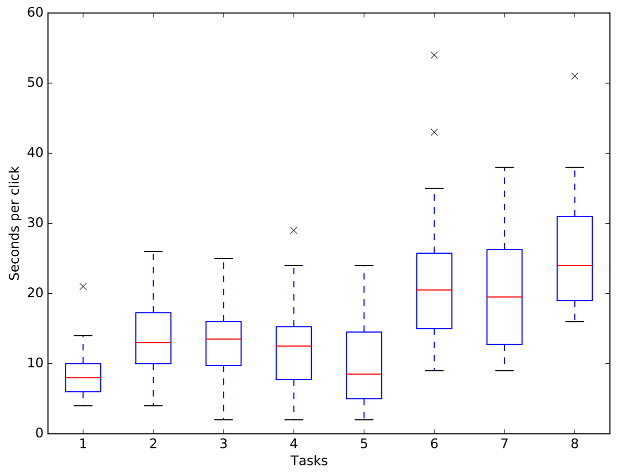


Fig. 9: Time spent per click at each task

is very easy to learn. We obtained an average score of 4.45, which goes in line with what we observed during the tests — last tasks were done more quickly, with less clicks, and errors.

During and after the tests, they also made some remarks about the system, praising the ease of use, especially for the semantic annotator. They, however, pointed to some areas that could be improved. Specifically, the ability to suggest locations that were not learned, but were used before, in the current annotation. Some users also found the buttons to split (in task 3) and join (in task 4) tracks not clear enough. This is supported by Table II, where the third quantile of both tasks have one error. They knew that one was the opposite of the other, but were not sure of the function of each button.

Overall, the results show that our application is simple and easy to use. The results also show that the semantic editor improves the speed with which they annotate tracks, specially using the suggestions.

## VIII. Conclusion & Future Work

In this paper, we presented GatherMySteps — a system that aims to simplify and improve the processing of GPS recordings, in the geographic lifelogging domain. The system achieves this by using a simple, yet power UI that streamlines the processing of personal GPS tracks, giving the user a reviewer role. This is possible because of an automatic processing phase, that cleans and identifies trips between two personally relevant places. Furthermore, the system helps users annotate trips, the start and end locations and the transportation modes. The system is also able to learn trips, locations, and transportation modes, further refining and adapting to the needs of a specific user. We evaluated our system with usability user tests, with the help of 20 participants. The results show that our system is easy to use, understand and learn having scored, respectively 4.15, 4.1, and 4.45, in a scale of 0 to 5.

We now plan to improve our system based on feedback we have received from the users, making some of the functionalities that were more difficult to use during the test more visible and clear to the user.

We believe that it is still possible to further improve the automatic processing of tracks; machine learning methods that take into account personal patterns could be used to extract more meaningful and personal information. The UI could also be improved, by adding the ability to apply automatic processing methods, such as smoothing, to certain parts of a track or trip. Systems such as GatherMySteps can greatly benefit the life of lifeloggers and further allow solutions to analyze personal information and annotations.

## Acknowledgements

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013.

## References

[1] Y. Zheng, L. Wang, R. Zhang, X. Xie, and W.-Y. Ma, "GeoLife: Managing and Understanding Your Past Life over Maps," in *The Ninth International Conference on Mobile Data Management (mdm 2008)*. IEEE, apr 2008, pp. 211–212.

[2] Y. Zheng, X. Xie, R. Zhang, and W.-Y. Ma, "Searching Your Life on Web Maps," jan 2008.

[3] Y. Zheng, L. Liu, L. Wang, and X. Xie, "Learning transportation mode from raw gps data for geographic applications on the web," in *Proceeding of the 17th international conference on World Wide Web - WWW '08*. New York, New York, USA: ACM Press, apr 2008, p. 247.

[4] K. Waga, A. Tabarcea, R. Mariescu-Istodor, and P. Fränti, "Real Time Access to Multiple GPS Tracks." in *WEBIST*, K.-H. Krempels and A. Stocker, Eds. SciTePress, 2013, pp. 293–299.

[5] K. Waga, A. Tabarcea, M. Chen, and P. Fränti, "Detecting Movement Type by Route Segmentation and Classification," *Conference on Collaborative Computing: Networking, Applications and Worksharing*, pp. 508–513, 2012.

[6] M. Umair, W. S. Kim, B. C. Choi, and S. Y. Jung, "Discovering personal places from location traces," in *16th International Conference on Advanced Communication Technology*. Global IT Research Institute (GIRI), feb 2014, pp. 709–713.

[7] A. Kirmse, T. Udeshi, P. Bellver, and J. Shuma, "Extracting patterns from location history," *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '11*, p. 397, 2011.

[8] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE transactions on pattern analysis and machine intelligence*, vol. 17, no. 8, pp. 790–799, 1995.

[9] R. Brunauer, M. Hufnagl, K. Rehrl, and A. Wagner, "Motion pattern analysis enabling accurate travel mode detection from GPS data only," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE, oct 2013, pp. 404–411.

[10] P. Bhargava, N. Gramsky, and A. Agrawala, "SenseMe: A System for Continuous, On-Device, and Multi-dimensional Context and Activity Recognition," in *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. ICST, dec 2014, pp. 40–49.

[11] G. C. Goodwin and K. S. Sin, *Adaptive filtering prediction and control*. Courier Corporation, 2014.

[12] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[13] N. Meratnia and A. Rolf, "Spatiotemporal compression techniques for moving point objects," in *International Conference on Extending Database Technology*. Springer, 2004, pp. 765–782.

[14] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.

[15] J.-G. Lee, J. Han, and K.-Y. Whang, "Trajectory clustering: a partition-and-group framework," in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. ACM, 2007, pp. 593–604.