Algorithm analysis

olivier.moitroux

February 2019

Part I
Initial plan

1 Data acquisition

1.1 Methodology

- Use a flutter background execution package to get and store gps data point onto device memory frequently in background
- Use a geofence parameter to detect the start of a trip (point within geofence won't be sent to the server nor stored). Use again a geofence parameters and a time treshold to stop the recording of a trip.
- Synchronize with the server the little trips (trajectories) in background (or not ?) as soon as the wifi is available.
- Analyse the pre-cut (by the app) trajectories of the user on the server and send him back the predictions.

1.2 Difficulties and flaws

- No (working) package of background execution working in flutter $\Longrightarrow$ need to code in both platform with unknown language for IOS.
- OS scheduler decide when to run the app code in background
  - no way to get points that are separated in a fixed time manner
  - points are gathered every 15 minutes at best
- No clear algorithm to process the data on-line ($><$ batch-mode)
- The application might not be aware enough to detect by its own the start of a trip due to imprecision, scheduler, ... $\Longrightarrow$ send everything to the server ?
- May not have enough data to extract correct semantic about clusters.

We can just maintain a table with an id per trajectory (caracterized like figure 1). Recompute the habbits once in a while (depending on user predictive behaviour). Cluster start and endpoint of each trajectories to recognize the trajectories that are the same -> discard these trajectories and replace them by the habbit with meta-data associated to it (confidence, number of trajectories to build it, std deviation of time, ...).

Part II
New project framework

2 Notions & definitions

Trajectory - gps log: set of GPS points that represent a user from point A to a point B.

Stay point [1]: locations where user stays between trajectories (detection covered in [1])

Start Point		End Point		Start Time	Duration	Length	Day	Direction
Lat	Long	Lat	Long					

Figure 1: Representation of a trajectory object

Travel sequences:

3 Data acquisition

Difficulties ?

- Thesis had garmin equipment with frequent update
- Often lot's of noise in data
  - Out of bound points, teleportation
  - Signal lost (building, narrow street, start in middle of highway)
- Lag between real start and start of trajectories
- Need to code in IOS and Androïd + integration with Flutter
- Data acquisition gonna be slow to accumulate enough data
- Can't define stay points, geofence, ... in the application because of tuning necessities !
- Curvature of the earth (1 degree $\approx$ 111km)
- Re-acquisition time after a signal loss is around 45 seconds to sometimes 5 minutes (see [5]) $\rightarrow$ time threshold to detect eventual stay points needs to be greater than 5 minutes.
- Storing data onto device may take a very large amount of space
- Lot's of IO operation if we store on disk (slow-more power drain) but take perhaps too much space on ram.

How do we get gps points in background ?

todo

No flutter package $\longrightarrow$ integration of specific-OS code (Androïd first). Use a foreground trick to be able to have control over the frequency of update of the gps point instead of having the app awaken only a few time per hour. If time available, we can reduce the battery usage (maintaining connection to satellite) by turning it off and using the wifi-signal power and gsm antennas.

How frequent should we get gps data ?

Geolife: once every 2 seconds and [5] used once every second. This is done in order to handle human walk that moves at around 3mph.

What processing does the application ?

todo

The application only controls the frequency of acquisition and the amount of data that will be sent to the server through a two parameters: space through a geofence of ... meters and time through a threshold of ... minutes. However, to tune the algorithms, we need to send everything to the server.

What libraries are we going to use ?

Methods to precise

What tuning parameters are expected for stay points ?

todo

[1] Time threshold = 20 minutes for a stay point and Geofence= 200 meters. (user stays in it for this amount of time)

What are the dataset properties required for proper analysis ?

todo

Dataset	km	users	period	city	size (Gb)	points/traj
Geolife[1]	1,292, 951	182 (students)	5 years	Beijing	4+	561

Table 1: Dataset of thesis

About Geolife:

(+) Data already pre-cut in trajectories (different .plt files) for each user when user arrived at destination.

(+) Lots of data

(-) Data already pre-cut in trajectories (different .plt files) for each user when user arrived at destination.

(-) Noisy and gaps between trajectories

(-) No data during stay (can't thus deduce importance of point of interest and differentiate a stay of 5 minutes or a longer one)

See [1] for stay point detection problem.

What is the data representation

The conversion of degree (longitude) to meters can be done by applying the technique developed page 14 of UCL thesis.

What load on the server is expected ?

todo

[...] recorded over 1Gbyte of data points every day. We expect ... of data because ...

Title	Representation
Datetime String	yyyy-mm-dd hh:mm:ss
Day of the week	int
UTC time	int
latitude	decimal
longitude	decimal
Accuracy	float (meters)
Speed	float (m/s)

Table 2: Formalism for data

How do we synchronize the acquired data to the server ?

todo

When/how frequent ?

Which formalism ?

How accurate is the gps ?

In [5], we have more or less 15 meters error but more can be expected !

Ideas or improvements

- Use of google latitude (depreciated)
- Encode data in .gpx format
- Asking for where is home and work place when sign up

4 Data analysis

4.1 Difficulties ?

- Two main techniques arise
- Small dataset to analyse
- No time to develop a visualisation tool
- No real "coude" in clustering plot
- Space scale may interfer analysis between location and sublocation (hard tuning)
- Tuning should be done in batch mode but final project should be in online mode for performance
- No online algorithm found up to now
- No feature involving the day of the week found (faux)
- The ugly theorem: there is no best set of features, must give some input to algo and will influence result (see [5]).

4.2 High-level workflow of a user-habit detection [UCL + [1]]

Based on using a ΔT and a geofence to detect stay points. We analyse two times the raw data. Once to detect locations=points of interest = label and a second time to detect trajectories. Then we keep trajectories linking two enough visited points of interest. These trajectories are subdivided according to their feature (mainly length). Then if a set of trajectories out of this subdivision is sufficiently large, it can be converted into an habit.

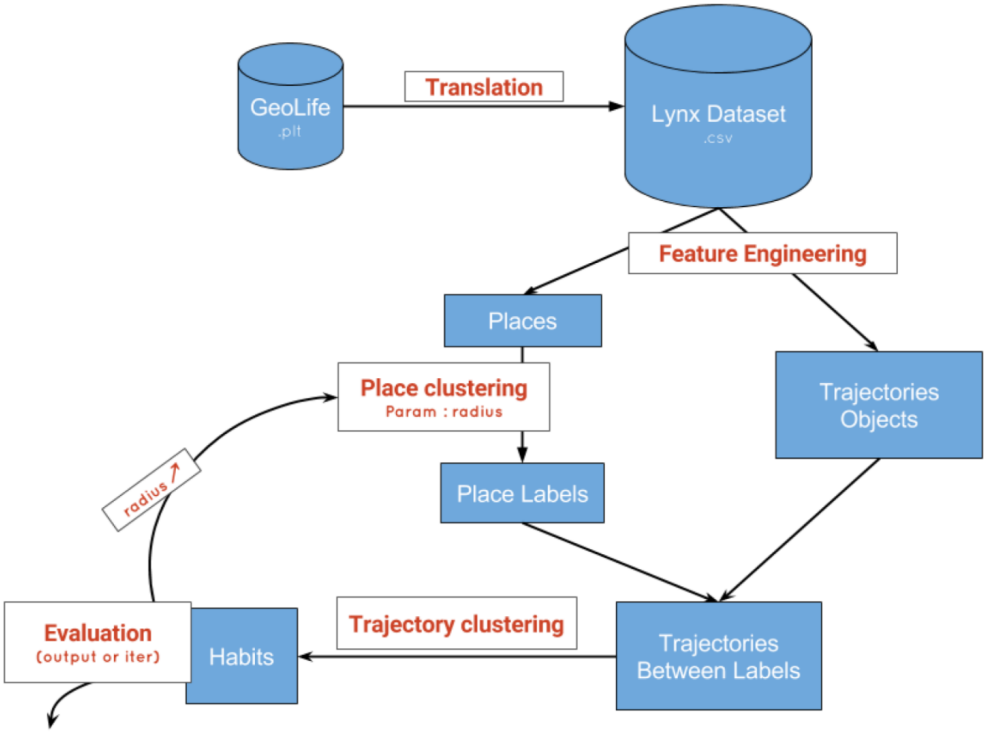


Figure 2: Main idea. NB: place labels = labels = location = points of interest. Place clustering to detect frequently visited areas, trajectory clustering to detect habits.

1. Data (gps-log) cleaning
   - Remove outliers (inconsistent distance), teleporation in space and too high speed
2. Cutting clean gps-log data of a user into trajectories
   - Detecting stay points (≡ start and stop points) [1] to cut gps log in trajectories
     - Could be done perhaps directly by the application (geofence+time) ??
     - Even if signal loss (tunnel), most of them should be ok
     - noise can be reduced with algo that detect cluster of points to detect a precise location
   - Trajectories exceeding 200 km are deleted
   - Users with less than 150 trajectories will not be used to test and tune the algo (irrelevant if we get our own data).
   - algo developped in [1]
3. Feature engineering: slight modification of data of trajectories
   - Start & stop position: trivially useful
   - Time: qualify repetitive behaviour (represented in circular space) [pg19 UCL]

Difficult (or is it location histories ?)!

- Day: qualify repetitive behaviour (represented in circular space) [pg 19 UCL]
- Length:
- Average direction: derived from start and stop position (represented in a circular space) [pg 21 UCL]

4. Detection of important place (≡ location) from cleaned gps-log data of everyone (?)

of everyone ou plutôt pour chaque utilisateur ??

   - goal:
     - Give info about frequently visited places, thus habits
     - Pruning of non-relevant trajectories (not often visited)
     - Only consider here all the start & end points of trajectories (intermediate points of gps-log are not kept because user is moving)
   - Cluster list of places to form location(=points of interest=labels) and give them a unique number
     (a) Centering of the centroïd in 6 steps [algo pg 24 UCL ou [5] Ashbrook] (in [5], they wait until the mean doesn't change anymore).
     (b) radius 20-30m might seem good for building but should be extended to 100-150m for noisy data
     (c) Thanks to this radius, we can gather all the start and end points of what is going to be a location(=point of interest)
   - Identify the most important places among all that were found
     - Most dense group selected by a threshold of minimum number of observations $\longrightarrow$ we get a list of locations that represent the most frequently visited places.

5. Trajectory clustering/filtering/sorting

   - Goal and filtering
     - To combine the locations (frequently visited places) with the trajectories of the user.
     - Highlight 3 kind of trajectories
       * location $\rightarrow$ location: pick this as interesting trajectories ≡ filtering
       * ?/location $\rightarrow$?/location: discard
       * ? $\rightarrow$?: discard
   - First phase of clustering
     (a) Filter (see above)
     (b) Trajectory sorting
       i. Regroup interesting trajectories by pair of labels they connect (has the same meaning as iii)
       ii. Collect interesting statistics
         - if number of label $\longrightarrow$ label trajectories is too small, clustering is too thin, so increase radius.
       iii. Trajectories are sorted by their location in a directed way (A$\longrightarrow$B $\neq$B$\longrightarrow$A)
   - Second phase of clustering
     (a) Inside every pair of label, can be multiple routes corresponding to differents habits (chemin différents)
     (b) Usefull features
       - Start time: ok mais bof
       - day of the week: ok mais bof
       - time: too noisy so discard
       - length: best $\longrightarrow$ on garde que celui-ci, pourrait être amélioré avec [7 Idiary].
     (c) algo [pg 29 UCL]

(d) Good interval value = 6% $\Longleftrightarrow$ majority of trajectories in a habit does not exceed this threshold without using a different route.

- Finally, compute habbits
  - minimum habits is 5 trajectories f.e.
  - quality ratio = $\theta = \frac{\text{Number of trajectories that are part of an habit}}{\text{Number of trajectories connecting two labels}} \approx$ 4-10% [UCL]
  - θ low $\Longleftrightarrow$ majority of routes between labels are not used enough to represent an habit so increase the radius of the clustering.
  - algo [pg 31 UCL]

6. Predict sequences in visited sequences (is it really useful for our project?)

BUT: No online mode. Analysis seems more expensive and no indication on probability of transitions, only habbits in a given day (hour ?). Is it enough to just list the habbits detected for a given day in the app ?

Gap in the gps signal

If the time between two points is greater than a certain threshold ΔT, this means the trajectory must be cut between these points. In the following formalism, they define L as the log of all GPS points, p_i as a GPS point and $p_i.T$ the time of the point p_i, $Traj = p_1 \longrightarrow p_2 \longrightarrow p_3 \longrightarrow \cdots \longrightarrow p_n$
Where

- $p_1 \in L$
- $p_{i+1}.T > p_i.T$
- $p_{i+1}.T - p_i.T < \Delta T$

Stay points

A stay point is a geographic region where the user logged in GPS point for certain time interval without leaving it. The extraction of such points depends on two parameters, the radius of the zone and the time threshold. Such a zone is characterized by a group of consecutive GPS points $P = \{p_m, p_{m+1}, \ldots, p_n\}$
where $\forall m < i \leq n$

- Distance(p_m, p_i) $<$ ΔDist
- the time $|p_{n'}.T - p_{m'}.T| \geq \Delta T$

The stay point is centered on the mean position of every points of P. The first point, p m , is considered as the arrival point and the end of the trajectory. The last point , p n , is the start of a new trajectory.

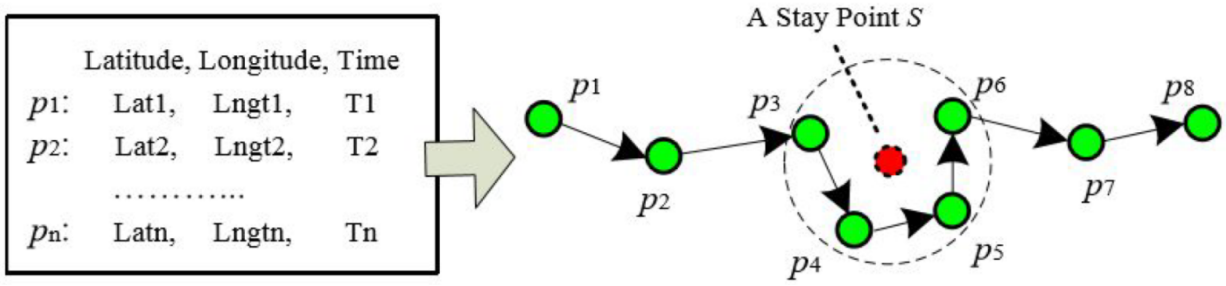


Figure 3: Stay points detection in raw GPS log

To tune the parameters, we need to send all the data to the server and make the stay points there. However, for final release, the stay points could already be computer on the app to alleviate the load on the server.

Radius-based clustering (same for next method)

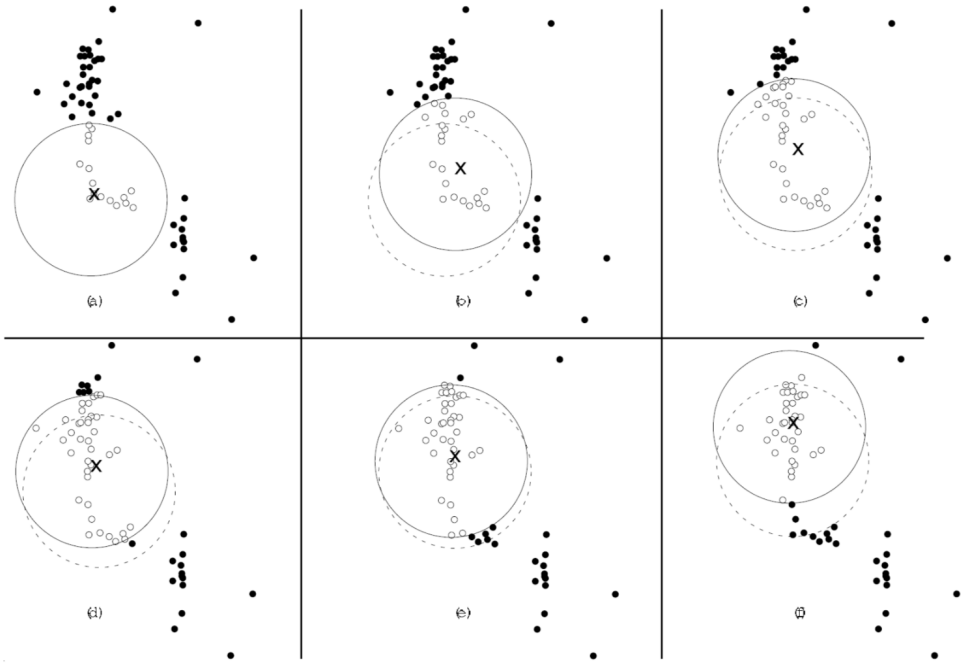


Figure 4: Illustration of location clustering algorithm. The X denotes the center of the cluster. The white dots are the points within the cluster, and the dotted line shows the location of the cluster in the previous step. At step (e), the mean has stopped moving, so all of the white points will be part of this location. 150 meters is a good value instead of the 20-30meters (semantically more correct).

4.3 High-level workflow of a user-habit detection [5]

We cluster the raw gps-log into locations = points of interest (and can extend to sub-locations). Then, by keeping the information on the order of visited locations, we build a MDP to link them and extract probabilities to infer a travel habbit based on probabilities this time. [5] heavily exploit the clustering curve and the change in the knee.

1. Find places (= stay points\ geofence in [UCL and 1])
   - Define a threshold Δt as an amount of stopping time (cfr. no one stop on the highway)
2. Clustering these places (found according to time) into locations (incorporate notion of space)
   - Variant of k-means clustering (radius-based) >< [1 pg 7] Density agglomerative approach that can detect clusters with irregular structures.
   - Algo [5 pg 5]
   - Algo to detect correct radius with change in slope [5 pg 5] (like we have seen in ML, just find the right radius between having one place per location and regrouping unrelated locations. We thus progressively compute the mean on the right of the graph and compare it with the current value, if the difference is too great, this means that we are at the point where the number of clusters has sufficiently converged. Thus, we are at the right radius) Don't understand
3. Learning sub-locations
   - City-wide (Liège $\neq$ Beijing) scale or campus-wide scale

- Algo [5 pg 6]
- Taking into account speed can enhance the prediction (not done)

4. Compress the data
   (a) Assign each location (cluster of previous step processed according to time AND space) an ID
   (b) Substitute for each places (only grouped by time) the id of the location it belongs to
      - This gives us a list of locations the user visited in the order they were visited
5. Create a Markov model for each location with transitions to every other location.
   - Each node in the Markov model is a location, and a transition |e| 2 nodes represent the probability of the user traveling between those 2 locations (if user never travelled between 2 locations, transition probability is worth 0).
6. Test on wether a path has sufficient evidence for prediction
   - Node with too few occurences (seldom travel) should be ignored for prediction
   - Compare the path's relative frequency to the probability that the path was taken by chance
   - See small example [5 pg 7] to compare with random
7. Increase Markov model order
   - Better tackle situation like I go to the Starbucks to drink a Coffee in my car on the way to home from university every day.
   - 4 months of data, few second order transition

BUT:

- No online mode
- No prediction on time (day of the week).
- Limitation of Markov model: changes in schedule (exams,...) may take a long time to be reflected in the model
  - Because in the model, each transition is given equal weight → might take the entire semester to adapt...
  - Solution: weight updates to the model more heavily (while not weighting unduly one-time trip though) or ask for feedback in the form of a button in the app !

.

Can we group raw gps-log data in days and apply separately this algorithm for each day ? Would we have enough data then ? Or should we compute the locations with all the data of the user and then, consider the day in the MDP to adapt probabilities ? Should be more or less fine: if I'm detected near a certain location a certain day, i'll be prompted with the most frequent destination for that day BUT I may have several different destination from A, we should therefore also consider time of the day, .. How ?

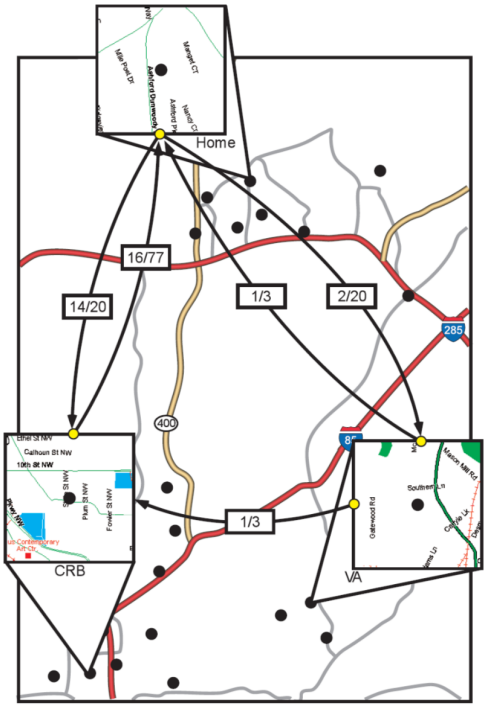
Ways to improve (change in methodology): "When finding places, an important consideration is how time is used in our previous study, we considered a point a place if it had time t between it and the previous point. This basically meant that places would be detected when the user exited a building and the GPS receiver reacquired a lock. Our current method, however, registers a place when the signal is lost, and so is not dependent upon signal acquisition time." ??

Workaround for lack a time analysis in Markov model: Looking for very long (6-10 hours) gaps in the data may clue us in to when people are at home sleeping without explicitly coding knowledge of day/night cycles into the algorithms.

Radius-based clustering

See subsection 4.2, page 8.

Markov model



(a) Partial Markov model of trips made between home, Centennial Research Building (CRB), and Dept. of Veterans Affairs (VA). Because some paths are not shown, the ratios do not sum to 1.

Transition	Relative Frequency	Probability
$A \to B$	14/20	0.7
$A \to B \to A$	3/14	0.2142
$A \to B \to C$	2/14	0.1428
$A \to B \to D$	3/14	0.2142
$A \to B \to E$	1/14	0.0714
$A \to B \to F$	1/14	0.0714
$A \to B \to G$	1/14	0.0714
$A \to B \to H$	1/14	0.0714
$A \to B \to I$	1/14	0.0714
$B \to A$	16/77	0.2077
$B \to A \to B$	13/16	0.8125
$B \to A \to J$	3/16	0.1875
$B \to C$	10/77	0.1298
$B \to C \to A$	6/10	0.6
$B \to C \to K$	4/10	0.4
$D \to B$	5/7	0.7142
$D \to B \to A$	2/5	0.4
$D \to B \to L$	2/5	0.4
$D \to B \to M$	1/5	0.2

(b) Probabilities for transitions in first and second order Markov models from preliminary data. **Key**: A = Home," B = CRB," and D = south of Tech."

Figure 5: Markov model [5]

5 Update of the GUI

- Detection of travel mean discarded (but could try to guess upon speed of travel though)
- Once a user has been detected at a possible location more than a couple times, we can prompt the user for a name. With multiple users, the system could suggest names for that location that other users had previously used. The user could also indicate that the detected location isn't meaningful and it could be ignored for future predictions.

6 Ways to improve

- On-line mode
- Detection of mean of travel

- If person A is found to be in several of the same locations as person B, it may be possible to use person B's collection of locations and predictions as a base set of data for person A. Then as person A continues to collect more data, the locations and predictions can be updated appropriately.
- Better exploitation of time
- When several users spend time at the same location and then increase speed, then this location may be a traffic jam

7 Quid on-line mode ?

[7]-[14]-[13] discuss a way to build coresets with streaming insertion support in $O(log^2 n)$. Too complicated.

Part III
Questions for client (depreciated)

7.1 Related to our initial plan

- Quid of the points of interest of CovoitUliege ? Can we use them ? How are they computed ? See main idea diagram, can we rely on them to select what trajectories we detected ?
- What can be offered by your api ?
- Can we have the root access to the server s.t. we can put our HTTPS certificate with let's encrypt ?
- Peut-on demander à l'utilisateur quels sont les points d'intérêts qui sont intéressants pour lui quand il crée son compte et analyser uniquement que ces points là pour détecter quand il les relie ?

7.2 Data acquisition

- Ok utiliser geolife ou autres dataset déjà existant en attendant ?
- We have difficulties to find a working module that can collect gps data at a sufficiently high frequence in background. What would you advise us?

7.3 Data analysis

- Do we need to predict what the next travel is gonna be according to detected habbits and can we just put in a list all the different habbits so that he can select the relevant ones by hand (like auto-completion of the form in a sense)
- can we use a button in the list of travels to notify to the server that there is a change in habbits ? Like new semester for example ?

7.4 Security

- ...

7.5 Server

- Quels sont les ordres de grandeurs propre à l'utilisation de la plateforme ? (Nombre d'utilisateurs, taille de l'information stockée sur le serveur - bdd, nombre de connection au serveur/minute, taille des packets échangés, durée de la connection au serveur par client, ...)