Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 52 (2015) 262 – 269

Procedia Computer Science

6th International Conference on Ambient Systems, Networks and Technologies (ANT 2015)

TRIP/STOP Detection in GPS Traces to Feed Prompted Recall Survey

Glenn Cich[a], Luk Knapen[a], Tom Bellemans[a], Davy Janssens[a], Geert Wets[a]

[a]*School of Transportation Sciences, Hasselt University, Wetenschapspark 5 bus 6 3590 Diepenbeek, Belgium*

Abstract

This paper presents two methods to extract trips and stops from GPS traces: the first one focuses on periods of non-movement (stops) and the second one tries to identify the longest periods of movement (trips). In order to assert the quality of both methods, the results are compared to stops and trips identified by the traveler; this was done by means of a visual tool aimed at alignment of manually reported periods in the diary to automatically recorded GPS coordinates. Several quality indicators are presented; they have been evaluated using sensitivity analysis in order to determine the optimal values for the detector's configuration settings. Person traces (as opposed to car traces) were used. Individual specific behavior seems to have a large effect on the optimal values for threshold settings used in both the trip and stop detector algorithms.

Accurate detection of stops and trips in GPS traces is vital to *prompted recall surveys* because those surveys can extend over several weeks. Inaccurate stop detection requires frequent corrections by the users and can cause them to quit.

Keywords: GPS Recording; Trace Segmentation; Stop Detection; Sensitivity Analysis

1. Problem Context

GPS traces are widely used as a data source in mobility science. Such data can be classified as "Big Data" and contains a large amount of interesting information, e.g. researchers extract information about *stops* (locations where people reside for some time in order to perform an intended activity) and *trips* or *moves* linking those stops in time and space. The required accuracy of the results depends on the purpose of the research. This paper focuses on the problem to determine stops from person traces (i.e. containing mixed sequences of walking, public transport and car movements) that will be used in a *prompted recall* survey to capture the traveler's intention. In a prompted recall survey, the stops and trips are presented on a map soon after their accomplishment and the traveler is urged to specify for each stop location the intention why it was visited. Prompted recall surveys often extend over a period of several

* Corresponding author. Tel.: +32-11-269-111 ; fax: +32-11-269-199.
E-mail address: glenn.cich@uhasselt.be

doi:10.1016/j.procs.2015.05.074

weeks. Hence accurate stop detection is crucial in order to avoid missing stop locations or presenting false positives to the respondent. Finally, since the interactive stop location annotation is performed before all data for the complete survey period become available, incremental stop detection is required. The GPS recording device uploads data as they come available and the traveler can decide to annotate new recordings several times a day.

This study compares two tools: (i) a STOP detector based on identification of spatial clusters of GPS coordinates and (ii) a TRIP detector that tries to identify sequences of GPS recordings corresponding to movements. The methods are in some sense complementary and the research aims to support the selection of the method that is best suited to feed the prompted recall survey.

This paper is organised in the following way. In Section 2 we give a brief overview of relevant related work. Sections 3 and 4 focus on algorithms to detect stops and trips. Section 5 presents some techniques to evaluate our TRIP/STOP detection algorithms. In Section 6 we compare these algorithms and discuss our findings. Finally, Section 7 gives a brief summary of the paper.

2. Related Work

In the literature, we find different types of TRIP/STOP detection methods. A first class makes use of *car traces*. Those methods are based on monitoring the engine of the car, namely when the car engine is turned on, the GPS recording starts and when the car engine is turned off, the GPS recording stops. In[1], Schönfelder et al. use such method in order to replace paper based travel data surveying by automatic data collection via GPS recordings. Automatic data collection reduces the burden for the respondents. The paper describes data capturing using a GPS recorder in a car. With this method, they roughly capture all the trips and stops. Nevertheless they encountered some difficulties, namely when a respondent performs a short stop (for example at the bakery). If the car engine is not turned off, this stop is not initially captured by the device. Another problem is caused by recording failures. In many cases the GPS signal was lost or zero speed measurements were reported. The authors suggest some solutions to this problem.

A second class includes points of interests (POI) to examine trips and stops. In[2], Alvares et al. attempt to answer questions about movements of people, e.g. "Which are the places most frequently visited by people attending a conference in a touristic city?". When using raw GPS trajectories, complex queries are required to answer such questions. For this reason, the authors try to add semantic information to the raw GPS data, i.e. they attempt to find "stops" and "moves" in the data. Their algorithm uses "interesting zones" to find stops. For example if a person stays for a minimum period of time in the neighborhood of the "Notre Dame" in Paris, this is flagged as a stop. Otherwise, if he does not stay a minimum of time in this zone, it is a move. Notice that this method requires a set of *interesting zones* and therefore it is application dependent. When conducting research about tourism, touristic POI are required. For example, a traffic light is no interesting zone for tourism but could be in another application context. In[3], Spinsanti et al. attempt to match stops with POI. For their experiment, they use GPS recorders that are located in a vehicle. When a person stops at a parking and then walks two kilometers to a shop, this method will miss this two kilometer trip. To detect stops, the authors use an algorithm that detects sufficiently long periods during which the vehicle doesn't move. After finding a stop, they attempt to match this stop to a POI, taking into account the opening times of the POI. For example when a person arrives at a parking nearby a shop and a museum, the algorithm will analyze whether or not the shop and/or the museum are open at that moment in time. In[4], Furletti et al. describe an improvement/extension of this paper.

In the present paper, we want to conduct a prompted recall survey. For such research we need *person traces*, but we do not know individual POI. Hence, a STOP/TRIP detection algorithm is needed that can operate with minimal information. Our work aligns with the work reported by Yan et al.[5]; they attempt to add semantics to raw GPS trajectories. Their algorithm uses five steps, namely "preparation", "data preprocessing", "trajectory segmentation", "stop identification" and "semantic enrichment". For stop identification, the authors have three methods. The first method is "velocity based", i.e. they calculate the speed between GPS points and when the speed is low enough, the algorithm assumes it is a stop; this corresponds to the trip detection described in Section 4. The second method is "density based" and corresponds to our method mentioned in Section 3. This method also considers, besides speed, the distance the object has traveled during a time period. The third method is "time series based"[6]. This method uses the forecast speed to detect stops. The prompted recall concept is explained in Stopher et al.[7]; in order to conduct this research, they first need to add semantics to their raw GPS traces. They use a data cleaning step and a stop detection

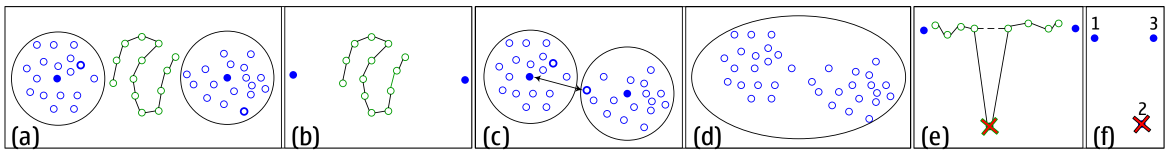

Fig. 1. (a) Stop detection with two stop clusters and one trip. (b) Stop detection with combined stop clusters. (c) Two stop clusters separated by a small distance. (d) A merge of the two stop clusters shown in (c). (e) Trip cleaning. (f) Stop cleaning.

step. After the execution of this algorithm, the researchers analyze the output data manually with `TransCAD` to add or delete unrealistic trip/stops.

3. Stop Detection

A stop detector processing GPS records in two passes is presented. The algorithm inspects the last few recordings received and performs spatial clustering. The first pass attempts to detect stops and trips by classifying every GPS record either as a *stop* member or a *trip* member and by assigning it the number of the stop or trip it belongs to. We can accomplish this by identifying clusters of points that are sufficiently close to each other. Points are represented as a triple $\langle x, y, t\rangle$ with x the longitude coordinate, y the latitude coordinate and t the timestamp. The second pass conducts cleaning of the stops and trips and it replaces the set of points constituting a stop, by a single representative point for which the attributes $\langle x, y, t\rangle$ are averaged over the cluster members.

3.1. Terminology

StopDurationThreshold denotes *the minimum duration for a period* a person needs to stay in a specific area in order to qualify the area as a *stop* location.

StopDistanceThreshold denotes the *size of the area* where a person needs to stay in order to qualify the area as a *stop* location.

A *stop cluster* is a circular area with a radius of *stopDistanceThreshold*. Every stop cluster has a *cluster center*, i.e. the center point of the circle that represents the stop cluster.

Last known stop point is the last point of a stop cluster, i.e. the point with the most recent timestamp in the stop cluster.

A *Stop* location (abbreviated to *Stop*) consists of a specific set of GPS points represented by a single $\langle x, y, t\rangle$ tuple together with the *stopDistanceThreshold* radius and the *stopDurationThreshold* period.

A *Trip* consists of a specific amount of GPS points. Notice that the sets of GPS points corresponding to trips and stops respectively are mutually disjoint. Each recorded GPS point belongs either (i) to a trip or (ii) to a stop or (iii) is discarded as an outlier.

In Figure 1(a), we show a conceptual view explaining the terminology. The larger circles represent stop clusters along with their associated member points. The filled point in the stop cluster is the *cluster center* and the point having a thick line perimeter is the *last known stop point*. The points that are connected with a line are part of a trip.

3.2. Algorithm: first pass

In Algorithm 1, the first pass of the algorithm is presented as pseudo-code.

The input to the stop detection algorithm consists of a GPS trace for a given person. Such trace is a chronologically ordered sequence of $\langle x, y, t\rangle$ tuples. The initial step is to label the first GPS record as the center of a provisional cluster, this can be noticed in Line 1. From this point we continue the algorithm by sequentially processing every GPS point fetched from the input, shown in Line 4.

In Line 5 we analyze whether or not *gpsRecord* spatially belongs to the current stop cluster; if so, the *gpsRecord* is labeled as a potential stop point. If not, we are certain that the current calculated stop cluster cannot be extended, but we do not know yet whether or not it is a valid stop cluster. The test in Line 8, analyzes the duration of the current stop cluster. If the duration of the current stop cluster is less than the *stopDurationThreshold*, we label all potential stop points as trip members, shown in Line 18. If the condition in Line 5 holds, we are certain that the current stop cluster is a valid stop. At this point in the algorithm, we know that a trip, if any was constructed, is complete. If there are potential trip points, we have to analyze whether or not the trip is a valid one (Line 9). Not every sequence of GPS recordings does represent a trip; therefore, the newly added sequence consisting of all $P \in$ *potentialTripPoints* is evaluated to find out whether or not it meets the criteria to constitute a (partial) trip. In a first experiment, following straightforward conditions were used: (i) the duration of the trip shall be strictly larger than zero, (ii) the average speed shall be sufficiently high and (iii) the number of GPS points in the sequence shall be not less than a specified minimum. Using only those conditions, we suffered from "*ghost*" trips that arise from point sequences that seem to contain measurement errors. It is assumed that those sequences occur when a person was in a building or tunnel or when the measurements got disturbed by the *urban canyon* effect. GPS errors caused many outliers. We solved this problem by considering the theoretical recording frequency of the GPS recording device.

When all previously stated conditions are satisfied, all $P \in$ *potentialTripPoints* are written to a file, otherwise they are discarded. In Line 12 the potentialTripPoints are cleared because we handled the trip points in the previous step. After handling the trip, we have to write the current stop to the output file. Remember that this was a valid stop, because the GPS points in the *potentialStopPoints* remained in the stop cluster for a period of *StopDurationThreshold*. At this point, when the algorithm detects two consecutive stops, it will analyze the distance between the two stops in both time and space. Spatial and chronological distance are calculated between the last point found in the first cluster and the center of the second cluster. If the distance is sufficiently small, both clusters are merged into a single one. This scenario is illustrated in Figure 1(c,d).

Finally, the *gpsRecord* being processed becomes the center for an new provisional cluster; it is also added to the *potentialStopPoints* (Line 15 and 16).

Algorithm 1 Stop Detection Algorithm

```
1:  clusterCenter ← first GPS record
2:  potentialStopPoints ← {}
3:  potentialTripPoints ← {}
4:  for all gpsRecord in input source do
5:    if distance(gpsRecord, clusterCenter) < stopDistance then
6:      add gpsRecord to potentialStopPoints
7:    else
8:      if duration(potentialStopPoints) > stopDuration then
9:        if isTripValid(potentialTripPoints) then
10:         writeToFile(potentialTripPoints)
11:       end if
12:       empty potentialTripPoints
13:       checkToMergeClusters()
14:       writeToFile(potentialStopPoints)
15:       clusterCenter ← gpsRecord
16:       add gpsRecord to potentialStopPoints
17:     else
18:       add all potentialStopPoints to potentialTripPoints
19:     end if
20:     empty potentialStopPoints
21:   end if
22: end for
```

3.3. Algorithm: second pass

The second pass consumes the output of the first pass. First, all points belonging to the same cluster are replaced by a single representative point. The result can be seen in Figure 1(b). This is a straightforward process; it will be not discussed here.

The second task consists of cleaning both *trips* and *stops*. For the trip cleaning we execute the following steps: for all but the last point in a trip, we calculate the average speed between the point and its successor (speed is calculated from exactly two GPS recordings). If the average speed between consecutive points A and B is too high, we delete point B. Hence we can solve GPS recording problems where a GPS point B appears at a distance of thousands of kilometers from the chronological predecessor A while the successor of B is again near to A. An illustration of such a situation can be found in Figure 1(e).

The *stop cleaning* part is similar to *trip cleaning*. We process all stops (i.e. the stop cluster representatives) and when the average speed between stop A and stop B is too high, we delete point B, as shown in Figure 1(f). Again, we can solve problems with outlier GPS points.

4. Trip Detection

The trip detector scans the GPS records and maintains a variable size sliding window containing the last records seen. Those records have not yet been finally qualified as *stop*, *trip* or *junk*. Each time a record is read, several quantities are evaluated: instantaneous and smoothed speed and acceleration, window size and period etc. Specific changes in the evaluated quantities lead to event firing. The events are fed to the state machine shown in Figure 2 that controls the qualification of the subsequence contained in the window. Definitely qualified records are dropped from the window.

Event firing is controlled by thresholds settings loaded from a configuration file. Two of them are used in sensitivity analysis: (i) the minimum period during which speed is below the *almostZeroSpeed* to fire *sufficZeroSpeedDurToStopTrip* and (ii) *bDistMinFirstAny* the minimal bird's eye view between the first point and at least one other point in a sequence to qualify the sequence as a tripComponent.

5. Sensitivity Analysis - Quality indicators

The quality of the trip and stop detectors was evaluated using a set of *person* (as opposed to *car*) traces captured using smartphones during 3 to 6 weeks. The surveyed people participated in a pilot project evaluating electric vehicles. The smartphone was used for GPS recording and for interactive diary data entry: recording information of activities conducted at locations where resided, travel mode used etc.

The diaries had been aligned with the GPS traces by means of a dedicated tool. Period start and end times were adapted such that there was no movement during periods labeled as *home*, *work*, *shopping* etc. The resulting cleaned diaries were used as "base truth" in the sensitivity analysis.

The two threshold settings mentioned in section 4 and two equivalent settings for the stop detector described in section 3 were given 10 values each (resulting in 100 runs for each user by each detector). The following sets of values were used:

$$\textit{StopDurationThreshold[sec]} = \{60, 90, 120, 150, 180, 210, 240, 300, 360, 420\}$$

$$\textit{StopDistanceThreshold[m]} = \{25, 50, 75, 100, 125, 150, 200, 250, 300, 400\}$$

Following quality indicators have been computed: (i) the average trip speed, (ii) the average trip duration, (iii) the average trip distance (all three calculated over all trips found in the trace for the respondent), (iv) the number of trips and (v) a temporal trip matching indicator based on the time intervals corresponding to the reported and the detected trips respectively; the indicator is the ratio of the duration of the intervals intersection period to the intervals union period. The ratio equals one if and only if the total trip times are equal and all intervals pairwise exactly overlap. The first three indicators are used for plausibility checking, the latter two are significant to assert the quality in the context of prompted recall surveys where we want to avoid false positives and false negatives during stop detection. Other indicators focusing on the time-shift between reported and recorded trips and their difference in duration, have been

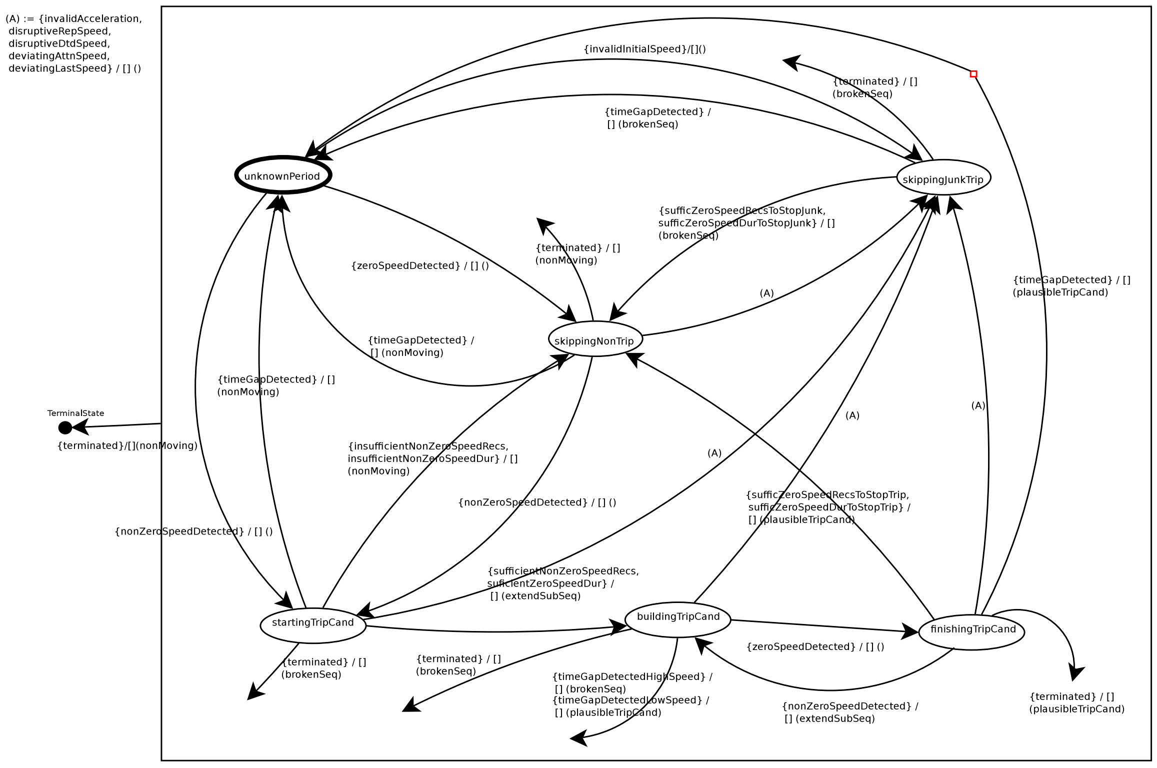
(A) := {invalidAcceleration, disruptiveRepSpeed, disruptiveDtdSpeed, deviatingAttnSpeed, deviatingLastSpeed} / [] ()
{invalidInitialSpeed}/[]()
{terminated} / [] (brokenSeq)
{timeGapDetected} / [] (brokenSeq)
unknownPeriod
skippingJunkTrip
{sufficZeroSpeedRecsToStopJunk, sufficZeroSpeedDurToStopJunk} / [] (brokenSeq)
{terminated} / [] (nonMoving)
{zeroSpeedDetected} / [] ()
(A)
{timeGapDetected} / [] (plausibleTripCand)
{timeGapDetected} / [] (nonMoving)
skippingNonTrip
{timeGapDetected} / [] (nonMoving)
TerminalState
{terminated}/[](nonMoving)
(A)
(A)
{insufficientNonZeroSpeedRecs, insufficientNonZeroSpeedDur} / [] (nonMoving)
(A)
{sufficZeroSpeedRecsToStopTrip, sufficZeroSpeedDurToStopTrip} / [] (plausibleTripCand)
{nonZeroSpeedDetected} / [] ()
{nonZeroSpeedDetected} / [] ()
{sufficientNonZeroSpeedRecs, suficientZeroSpeedDur} / [] (extendSubSeq)
startingTripCand
buildingTripCand
finishingTripCand
{terminated} / [] (brokenSeq)
{terminated} / [] (brokenSeq)
{zeroSpeedDetected} / [] ()
{timeGapDetectedHighSpeed} / [] (brokenSeq)
{timeGapDetectedLowSpeed} / [] (plausibleTripCand)
{nonZeroSpeedDetected} / [] (extendSubSeq)
{terminated} / [] (plausibleTripCand)

Fig. 2. Finite state machine to control the trip detector while scanning GPS records. The *unknownPeriod* state (bold perimeter) is the initial state.

considered too. Those are only useful if the number of reported trips equals the number of recorded trips (which does not frequently occur).

For a representative set of users, the 100 indicator values have been plotted in a 3D diagram using a $10x10$ grid. Examples are shown in Figure 3. The horizontal plane drawn as a grid, represents the *true value* extracted from the interactively aligned diary. The intersection of the surface representing the detector results for the given thresholds with the *truth plane* has been projected on the horizontal lower plane. This projection is the geometric place consisting of pairs of threshold settings that deliver the value reported by the respondent. When error values below a given threshold are allowed, the geometric place consists of a (not necessarily connected) region. Optimal settings have been derived by a least square method. For each parameter combination, the deviation between the reported and calculated (i) number of trips and (ii) temporal indicator are determined. Error values for (i) and (ii) were squared and summed over the set of traces. This resulted in two separate rankings which have been merged to one final list by summing up the rank number for each parameter combination. Finally, the parameter combination with the lowest ranking is the preferred one.

6. Discussion - *Stop* and *Trip* Detection Comparison

6.1. Sensitivity Analysis Results

Sensitivity to threshold settings was analyzed in two ways: graphs and temporal indicator (discussed in Section 5) have been compared. We compared the iso-lines (i.e. the *geometric place* projection mentioned above) and tried to identify similar patterns.

In around 75% of the cases, we found roughly similar iso-line patterns. For the *average distance* determined by the stop detector, a typical L-shaped pattern was found as shown in Figure 3. In a minority of cases other iso-line shapes occurred; in some cases the the distance was always over- or under-estimated and hence no iso-line was discovered.

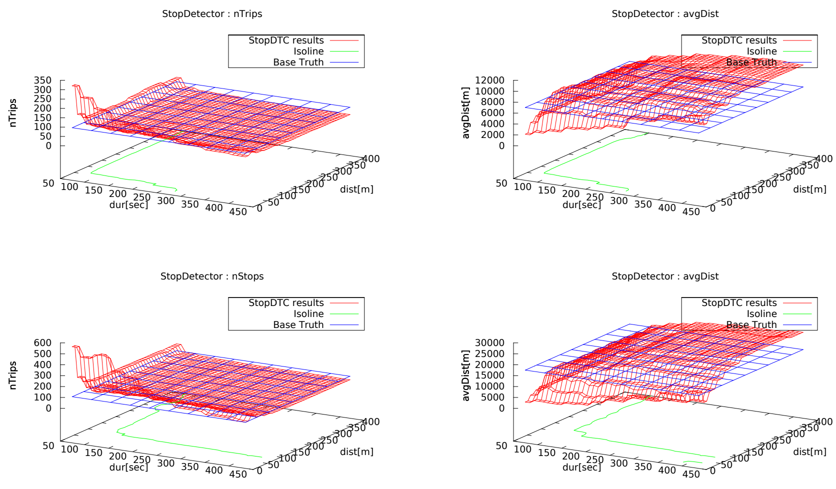

Fig. 3. Sensitivity analysis indicators: the number of trips (left) and the average trip distance (right). The horizontal blue rectangular plane grid represents the value reported by the respondent. The green line is the projection of the intersection of this plane with the detector output.

There seems not to be a single point belonging to each of the geometric places. We attempted to identify a small region that is crossed by all geometric places (iso-lines) for the "*average distance*", "*average duration*", "*average speed*" and "*amount of trips*" indicators. In the majority of cases, we found an intersection in the following intervals:

duration[sec]: [100, 200] and distance[m]: [50, 100]

The visual representation of the different parameters gives us a good initial idea about the correctness of our algorithms. However we need a more quantitative method to compare the results of the TRIP/STOP detector with the "base truth". Hence we calculated the *minimal ranked squared error* method as discussed in Section 5. Based on all the results, it was clear that the STOP detector performed better than the TRIP detector. The best results are shown in in Table 1 (sorted by the column "Rank[StopDTC]" in decreasing order).

Table 1. Some results of the ranked squared error.

StopDuration[sec]	StopDistance[m]	Rank[StopDTC]	Rank[TripDTC]	Temporal[StopDTC]	Temporal[TripDTC]
180	100	4	85	0.6320	0.4514
180	125	7	86	0.6316	0.4514
180	150	10	84	0.6260	0.4516
150	100	20	55	0.6335	0.4610
240	150	21	125	0.6257	0.4274

The best result we found was a ranking of four. This means that this parameter combination scored very well in both squared error lists (nTrips and temporalIndicator). In Table 1, we show the corresponding average temporal indicator as well. On average, we found an overlap of 63%. At a first glance, this does not seem to be ideal but careful interpretation is required. The results might be misleading, due to the length of the trips. When calculating the overlap of several trip intervals, it is possible that the trip periods identified by the stop detector are shifted relative to the "base truth" trips. Such situation is illustrated in Figure 4. The horizontal bars represent the trips. The green bars show high overlap situations. The red bars show situations where the overlap indicator value is low. We see that the overlap in red is very small, but due to the limited trip length, the overlap indicator value amounts to 50% or less. If the trace contains many of these trips, the overlap indicator will be low, although the calculated trips might be acceptable for use in a prompted recall survey. However, we tried to strengthen our comparison by taking into account the amount of trips as well.

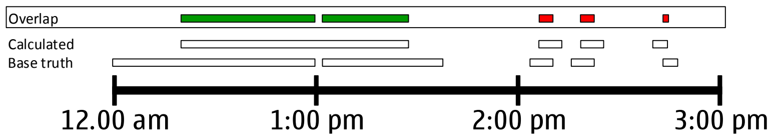

Fig. 4. An illustration of the overlap between calculated trips and "base truth" trips.

Finally, Table 1 confirms the conclusion drawn from the visual analysis of the of the iso-lines. As a result, we choose threshold values *180 seconds* for the duration and *100 meters* for the distance for use in the prompted recall survey.

6.2. Reasons for High Degree of Variability

As discussed in Section 6.1, no single pair of threshold parameters suits all cases. This is assumed to be caused by the use of person traces as opposed to car traces. The slow movement of people (e.g. walking) can be the cause for missing trips, especially in combination with short trips. Examples are social visits or daily shopping at a small distance from home. Due to the limited trip length and duration and the low speed of the trip, there is a large probability that this trip is just discarded or annotated as a stop. In general, trips are discovered more accurately when the trip is made using a fast moving vehicle and over a decent amount of kilometers. Therefore, it is likely that person characteristics such as work, preferences and possession of a car might be reasons for the diversity between the users.

The use of person traces means that each participant carries a GPS recorder all the time, i.e. in the car, when walking, in buildings. Problems may occur when people enter a building. The GPS signal will lose much accuracy and outlier GPS recordings will show up. These outlier GPS points are problematic, because they can pretend to be trips. A solution was discussed in Section 3.2, however this solution might not work for every possible case.

7. Conclusion

In this paper, we attempted to discover the best algorithm (trip or stop detector) to feed a prompted recall survey. Overall, the stop detector performed better than the trip detector. The trip detector makes use of more threshold settings (e.g. maximal acceleration, expected recording frequency, minimum developed length for a trip, etc.) than the stop detector. A single set of values for those settings was used; it is possible that some of those settings were not suited for the particular dataset. The additional parameters turn out to make the trip detector less robust. Apart from algorithm selection, we also attempted to discover optimal settings for the *stopDurationThreshold* and the *stopDistanceThreshold*. Due to the high variability of the results, this is a cumbersome task. The pair delivering the lowest value for the minimal ranked squared error indicator was chosen.

References

1. Schoenfelder, S., Axhausen, K., Antille, N., Bierlaire, M.. Exploring the potentials of automatically colected GPS data for travel behaviour analysis: A swedish data source. Tech. Rep.; Ecole Polytechnique Fédérale de Lausanne; 2002.
2. Alvares, L.O., Bogorny, V., Kuijpers, B., de Macedo, J.A.F., Moelans, B., Vaisman, A.. A model for enriching trajectories with semantic geographical information. In: *Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*; GIS '07. New York, NY, USA: ACM. ISBN 978-1-59593-914-2; 2007, p. 22:1–22:8. URL: http://doi.acm.org/10.1145/1341012.1341041. doi:http://doi.acm.org/10.1145/1341012.1341041.
3. Spinsanti, L., Celli, F., Renso, C.. Where you stop is who you are: visited. 2010.
4. Furletti, B., Cintia, P., Renso, C., Spinsanti, L.. Inferring human activities from GPS tracks. In: *UrbComp 13 Proceedings of the second ACM SIGKDD International Workshop on Urban Computing*. Chicago: ACM; 2013, .
5. Yan, Z., Spaccapietra, S.. Towards semantic trajectory data analysis: A conceptual and computational approach. In: *VLDB2009 (PhD Workshop)*. 2009, .
6. Yan, Z.. Traj-ARIMA: A spatial-time series model for network-constrained trajectory. San Jose, CA, U.S.A.; 2010, .
7. Stopher, P., Prasad, C., Zhang, J.. Can GPS replace conventional travel surveys? some findings. In: *Australasian Transport Research Forum 2010 Proceedings*. Canberra, Australia; 2010, .