Speed-Based Stop Region Detection

Sean Fitch, Andrew Fernandes

[a]*The Swift Group, Reston, VA*

Abstract

This paper introduces an algorithm for detection of stop regions in a spatiotemporal GPS trajectory from mobile devices. Existing work in this area was generally found to be impractical due to high complexity or vulnerability to problems like signal loss. The proposed algorithm, Speed-Based Stop Region Detection (SBSD), labels each point in a trajectory as a stop or move, and then uses a series of spatial and temporal merge metrics to combine them into reasonable clusters. The resultant regions are found to be robust to tight clustering and signal loss.

1. Introduction

Stop region detection has been extensively studied. Most approaches include some form of density-based clustering based on DBSCAN or OPTICS (Mainak, 2020).

The goal of SBSD is to present an algorithm based on easily understood features of trajectories, resulting in simpler and more easily maintained code than comparable algorithms.

Existing approaches were found to be overly sensitive to parameters or vulnerable to changes in sample rate due to signal loss. SBSD is a new algorithm which, while parameterized, is not overly sensitive to parameters and may employ simple heuristics for choosing parameters. Additionally, it is found to have high accuracy for stop region detection and high accuracy for stop region duration.

Notably, this algorithm is domain-specific to use on Augmented GPS Trajectories from mobile devices, as it depends not only on time and location, but also on the device's reported speed and accuracy. This augmentation, while narrowing the scope of SBSD, allows it to remain much simpler than similar algorithms while retaining high accuracy.

2. Definitions

The following definitions are used in the algorithm description.

2.1. Accuracy

The accuracy of a GPS point is an estimation of the error in the latitude and longitude reported in the point. This can be calculated from features such as number of GPS satellites. The accuracy used in this paper is a scalar in meters where $0 <$ accuracy < 400. It is calculated by software on the mobile device.

2.2. Augmented GPS Trajectory

A trajectory is represented by a sequence of spatiotemporal points sampled at certain time intervals, denoted as $TR = \langle p_1, p_2, p_3, ..., p_n \rangle$. Each point is a triple $p_i = (\text{latitude}_i, \text{longitude}_i, t_i)$, where $t_{i+1} > t_i$. This definition is consistent with existing literature (Wang et al., 2022).

An Augmented GPS Trajectory (AGT) is a trajectory $AGT = \langle p_1, p_2, p_3, ..., p_n \rangle$. Differing from the trajectory above, each point includes the reported speed and accuracy with the latitude and longitude, so $p_i = (\text{latitude}_i, \text{longitude}_i, \text{accuracy}_i, \text{speed}_i, t_i)$, where $t_{i+1} > t_i$.

2.3. Stop Region

A stop region (SR) is a subsequence of an AGT $SR_i = \langle p_{j_1}, p_{j_2}, ..., p_{j_n} \rangle$, where $p_{j_k} \in AGT$ and $t_{j_{k+1}} > t_{j_k}$. t_{entry_i} and t_{exit_i} denote the entry and exit times of the region.

Additionally, for any two SRs SR_i and SR_j, $SR_i[-1].time < SR_j[0].time$ or $SR_j[-1].time < SR_i[0].time$, meaning that an SR is defined by one stop at one location rather than multiple stops at one location separated by other stops. They cannot overlap.

3. The Algorithm

SBSD takes as input an AGT and returns a set of SRs. It first labels each point as a *stop* or *move*. It then groups adjacent *stop* points into SRs. It then splits SRs if they have sudden movements, and finally merges SRs based on spatial criteria.

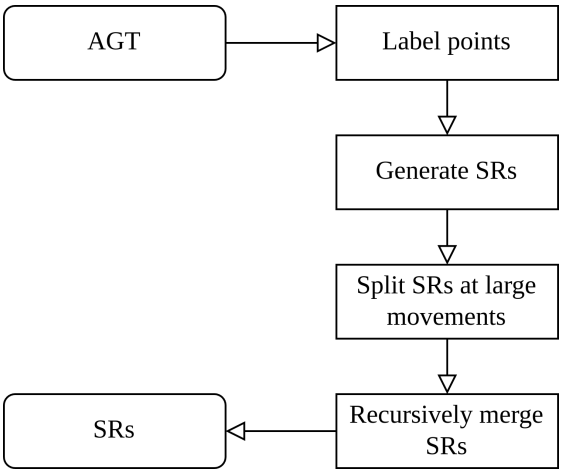

Figure 1: SBSD flow chart

3.1. Label Points

To label the points, both the speed reported by the device in the AGT and the speed calculated by distance over time are compared against thresholds. This is because using either alone results in some SRs being fragmented or missed entirely. For a more detailed discussion of the reported speed, see Section 4.2.

Algorithm 1 Label Points

Require: AGT of length N, speed_thresh, calculated_thresh
Ensure: $L \in \mathbb{Z}^N$
  $L \leftarrow [move_1, ..., move_N]$
  for i in $range(1, N)$ **do**
    $dist \leftarrow distance(AGT_{i-1}, AGT_i)$
    $time \leftarrow t_i - t_{i-1}$
    $calc \leftarrow \frac{dist}{time}$
    if speed_i < speed_thresh **or** $calc$ < calculated_thresh **then**
      $L_i \leftarrow stop$
    end if
  end for

The resultant array L contains binary labels for *move* and *stop*.

3.2. Generate SRs

To generate the SRs, sequential subsequences labeled as *stop* are grouped into a SR.

Algorithm 2 Generate SRs

Require: AGT of length N, $L \in \mathbb{Z}^N$
Ensure: SRs
  $SRs \leftarrow \{\}$
  $SR = []$
  for i in $range(0, N)$ **do**
    if $L_i == stop$ **then**
      SR.append(AGT_i)
    else if $L_i == move$ **and** len(SR) > 0 **then**
      SRs.add(SR)
      $SR = []$
    end if
  end for

3.3. Split SRs

The SRs are then split at any points where two points in the SR move by more than the sum of their accuracy plus an error threshold. This is necessary because there could be periods of signal loss which start and end in two separate SRs. This would lead to the SRs being grouped together since they are a sequential subsequence with the same label.

Algorithm 3 Split SRs

Require: SRs of length M, accuracy_error
Ensure: SRs of length $\geq M$
  $SRs\_new \leftarrow \{\}$
  while len(SRs) > 0 **do**
    $SR = SRs$.pop(0)
    for i in $range(1, \text{len}(SR))$ **do**
      $dist$ = distance(SR_{i-1}, SR_i)
      $thresh$ = accuracy(SR_{i-1}) + accuracy(SR_i) + accuracy_error
      if $dist > thresh$ **then**
        $SR_1 \leftarrow [SR[0 : i]]$
        $SR_2 \leftarrow [SR[i :]]$
        SRs_new.append(SR_1)
        SRs.insert($0, SR_2$)
      end if
    end for
    SRs_new.append(SR)
  end while
  $SRs \leftarrow SRs\_new$

The sum of their accuracy plus an error is an estimate for the maximum measured distance between two points which have identical actual coordinates. Using this as the threshold for splitting assumes that there should be negligible movement during an SR. Further research should be done into whether this assumption is true. If it is not, perhaps there should be a threshold based on allowed movement during an SR as a function of the time difference between the points.

3.4. Merge SRs

The resultant SRs are then merged with temporally adjacent SRs if they pass within a distance threshold of each other.

This prevents one SR from being frequently split due to sudden movements within the SR.

This distance metric also requires a shape to be defined for each SR. To remove outliers, we define the shape as the convex hull around all points with accuracy below a threshold. If this leaves too few points, use the convex hull around all points without filtering.

Algorithm 4 Merge SRs

Require: SRs of length M, distance_thresh
Ensure: SRs of length $\leq M$
  do
    $count \leftarrow \text{len}(SRs)$
    $SRs\_new \leftarrow \{\}$
    for SR in SRs **do**
      if len(SRs_new) == 0 **then**
        SRs_new.append(SR)
        continue
      end if
      $dist$ = min_distance($SR, SR\_new[-1]$)
      if $dist <$ distance_thresh **then**
        $SR\_new[-1] = SR\_new[-1]$.union(SR)
        continue
      end if
      SRs_new.append(SR)
    end for
    $SRs \leftarrow SRs\_new$
  while len(SRs) $<$ $count$

4. Discussion

4.1. Parameters

There are four parameters relevant to SBSD. They are highly important to the quality of the resultant SRs but fortunately are not overly sensitive and represent easily understood quantities.

Parameter	Unit	Recommended Value
speed thresh	m/s	0.5
calculated speed thresh	m/s	0.25
accuracy error	m	30
distance thresh	m	10

There are two thresholds for initial labeling, the speed threshold and the calculated speed threshold. Setting either of these lower will result in more points being labelled as stops, and conversely for setting them higher. The speed threshold should be chosen to be as large as possible without triggering when the target is walking for transportation. The calculated threshold may be a much more conservative number as it primarily exists as a secondary check for signal loss (see Section 4.2).

The accuracy error parameter is used for splitting SRs. This is required to prevent low accuracy points from splitting an SR where it should not. Setting it lower will result in more SRs being split and conversely for setting it higher. The criteria for splitting an SR is wherever the distance traveled exceeds the sum of the points' accuracies and the accuracy error. This does not account for actual movement within a stop region, which may happen if, for example, a person is moving from aisle to aisle in a grocery store. Therefore we add the accuracy error to account for this. This may be a conservatively large number as cases in which a SR should be split are rare and usually obvious.

The distance threshold is used for merging temporally and spatially adjacent SRs. Setting it lower will result in more, smaller final SRs and conversely for setting it higher. It may be relatively small as SRs which should be merged will typically overlap.

4.2. Why Augment the Trajectory?

The use of additional features beyond the latitude, longitude, and time of an AGT is largely unprecedented. Where speed has been used for SR detection, it is typically calculated based on change in position over time. This begs the question of why to make the algorithm dependent on other information.

The main advantage of using the reported speed is that it is more accurate and less noisy than the calculated value. This is because the speed calculated by the device uses a higher sample rate in addition to other sensors. For an example, see Fig. 2. The calculated speed is much larger than it ought to be, since the error in position is high. This is far more drastic than a typical scenario, but these do exist and result in fragmentation of stops. Conversely, the device's reported speed is robust to such error. This removes the need for smoothing and generally allows for a simpler model.

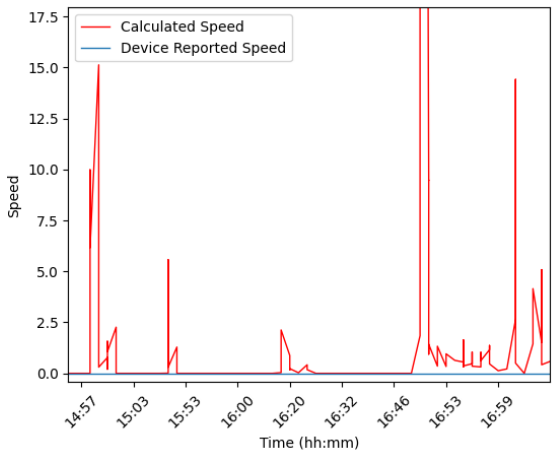

Figure 2: Reported vs calculated speed during a stop with high position error

Note that this does not mean we can use only the reported speed and ignore the calculated speed. This is because there are cases where there is signal loss for the entire duration of a stop. For example, see Figure 3. The time difference is 3 hours and the distance is 1300m resulting in a calculated speed of 0.13. This should obviously create an SR but the reported speed (3 and 19 m/s on either side of the stop) is insufficient to find this.

Figure 3: Example of signal loss during a stop.

As for the accuracy, it is a very convenient piece of information to include, as it allows for very simple outlier rejection in shape definition and a simple heuristic for splitting stop regions. It is also calculated only from information accessible to a typical GPS unit, so it does not significantly diminish the generality of the algorithm.

4.3. Time and Memory Complexity

The algorithm's time complexity is $O(N)$ in the number of points in the trajectory. Algorithms 1 and 2 iterate over every point in the AGT. Algorithm 3 iterates over every point in every SR and a point can only be in one SR. Algorithm 4 iterates over SRs and computes the distance to the prior SR. The distance between two convex sets can be computed in linear time with respect to the number of vertices (Gilbert et al., 1988), and the total number of vertices among comparisons must be less than $2N$.

Notably, algorithms 1-3 could be combined into one loop to reduce overhead. Currently, the distance between adjacent points is calculated in Algorithm 1 and is also required for points in SRs in Algorithm 3. Distance is one of the more expensive calculations required by the algorithm. This would prevent the need to either store or recalculate those distances.

The algorithm's memory complexity is $O(N)$.

4.4. Stay Time Filtering

For post processing of the SRs, it is typically desirable to find their stay times and filter them by a lower threshold. This allows exclusion of stops such as stop lights. In order to increase stay time accuracy, it is recommended to account for possible signal loss time on either side of the SR. This is especially important in the case of one point SRs, as otherwise their stay time would be 0. Below is an algorithm for estimating the stay duration including signal loss time.

Algorithm 5 Get Stay Time

Require: SR, AGT
Ensure: t
$t \leftarrow \text{time}(SR[-1]) - \text{time}(SR[0])$
$last \leftarrow AGT[\text{indexof}(SR[0]) - 1]$
$next \leftarrow AGT[\text{indexof}(SR[-1]) + 1]$
$travel\_t \leftarrow \text{distance}(SR[0], last)/\text{speed}(last)$
$signal\_loss\_t \leftarrow \text{time}(SR[0]) - \text{time}(last) - travel\_t$
$t \leftarrow t + \max(0, signal\_loss\_time)$
$travel\_t \leftarrow \text{distance}(SR[-1], next)/\text{speed}(next)$
$signal\_loss\_t \leftarrow \text{time}(next) - \text{time}(SR[-1]) - travel\_t$
$t \leftarrow t + \max(0, signal\_loss\_time)$

Once the stay time is found for each SR, they should be filtered by minimum stay time. 5 minutes was found to be a reasonable threshold.

References

Gilbert, E., Johnson, D., Keerthi, S., 1988. A fast procedure for computing the distance between complex objects in three-dimensional space. IEEE Journal on Robotics and Automation 4, 193–203. doi:10.1109/56.2083.

Mainak, B., 2020. Leveraging clustering validation index for detecting 'stops' in spatial trajectory data: a semi- automatic approach. Spatial Science doi:10.1080/14498596.2020.1787254.

Wang, S., Niu, X., Fournier-Viger, P., Zhou, D., Min, F., 2022. A graph based approach for mining significant places in trajectory data. Information Sciences 609, 172–194.